# SOLVING GENERAL LINEAR BANDED SYSTEMS

In the following, we will assume that we are solving a system of $n$ equations in $n$ unknowns so that the number of rows and columns in our matrix $\mathbf{A}$ are equal.

A general banded system is one in which the non-zero elements of $\mathbf{A}$ are confined to a certain number of diagonals of the matrix, usually clustered about the main diagonal. Tridiagonal systems clearly fall into this category. Systems with more than 3 non-zero diagonals also often arise in finite difference calculations (and other applications) and can also be efficiently LU-decomposed by taking advantage of the non-zero structure of the array.

## EXAMPLE: PENTADIAGONAL (5-DIAGONAL) SYSTEM

Recall the BVP we have considered previously

$$u''(x) = f(x) \qquad 0 \le x \le 1 \qquad u(0), u(1) \ \text{ specified}$$

Let us approximate $u''(x)$ using a fourth-order ($O(h^4)$) finite-difference approximation:

$$\frac{-u_{i-2} + 16u_{i-1} - 30u_i + 16u_{i+1} - u_{i+2}}{12h^2} = u''(x_j) + O(h^4) \qquad i = 3, \ldots, n-2$$

(Part of Homework 4 involves using a Richardson-extrapolation technique to derive the above formula from the usual $O(h^2)$ approximation of the second derivative). Because this approximation couples the $i$th unknown, $u_i$ to its nearest, and next-nearest neighbors, $u_{i-1}, u_{i+1}, u_{i-2}$ and $u_{i+2}$, the linear system which results from applying the approximation at mesh points $x_i, \quad i = 3, \ldots, n-2$ is (ignoring momentarily the equations for $u_1, u_2, u_{n-1}$ and $u_n$) is clearly pentadiagonal:

We define the *total bandwidth*, $w$, of a matrix as the total number of diagonals which span the non-zero portion of the matrix. In our current example, the bandwidth is $w = 5$, and all 5 diagonals are generally comprised of non-0 elements. However, particularly for finite-difference approximations to problems in multiple dimensions, some of the diagonals within the bandwidth of the matrix may have elements which are all 0.

As already stated, LU decomposition can be optimized for banded systems, both in terms of execution time:

$$O(c_w n) \quad \text{vs.} \quad O(n^3)$$

and storage requirements

$$O(n) \approx (2\texttt{kl} + \texttt{ku} + 1)n \quad \text{vs.} \quad O(n^2)$$

where

$$c_w \;\equiv\; w\text{-dependent constant}$$

$$\texttt{kl} \;\equiv\; \text{number of lower diagonals}$$

$$\texttt{ku} \;\equiv\; \text{number of upper diagonals}$$

Note that although $c_w$ is a fairly rapidly varying function of $w$ (must go to $n^2$ as $w$ goes to $n$), for any fixed bandwidth, the number of operations needed to solve the banded system is proportional to the size, $n$, of the system.

## LAPACK STORAGE SCHEME FOR BANDED MATRICES

First note that the length of the $k$th sub- or super-diagonal is $n_k = n - k$. Thus, particularly for $n \gg w$, $n_k \approx n$, and we can efficiently store the matrix in a 2-d array in which the *diagonals* of the matrix are stored in either rows or columns of the array. `LAPACK` adopts the former strategy:

- Columns of matrix are stored in corresponding columns of array (but elements are shifted)
- Diagonals of matrix are stored in rows of array

**Example:** $n = 5$, $\mathtt{kl} = 2$, $\mathtt{ku} = 1$, $w = 4$

$$\begin{bmatrix} a_{11} & a_{12} & & & \\ a_{21} & a_{22} & a_{23} & & \\ a_{31} & a_{32} & a_{33} & a_{34} & \\ & a_{42} & a_{43} & a_{44} & a_{45} \\ & & a_{53} & a_{54} & a_{55} \end{bmatrix} \quad \text{is stored as} \quad \begin{bmatrix} \star & a_{12} & a_{23} & a_{34} & a_{45} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \\ a_{21} & a_{32} & a_{43} & a_{54} & \star \\ a_{31} & a_{42} & a_{53} & \star & \star \end{bmatrix}$$

Here, $\star$, denotes an "undefined" element which will not be referenced by `LAPACK` routines.

## ADDITIONAL WRINKLE

To perform the LU decomposition with partial pivoting and row interchanges (for numerical stability) `LAPACK` routines require `kl` additional rows of storage. (Elements of these rows need not be assigned values prior to calling the solver.) Thus, in the above example, we would actually need to set-up our banded array `ab` as follows:

$$\begin{bmatrix} - & - & - & - & - \\ - & - & - & - & - \\ \star & a_{12} & a_{23} & a_{34} & a_{45} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \\ a_{21} & a_{32} & a_{43} & a_{54} & \star \\ a_{31} & a_{42} & a_{53} & \star & \star \end{bmatrix}$$

Note that the leading dimension, `ldab`, of this array is thus

$$\texttt{ldab} \quad = \quad 2\,\texttt{kl} + \texttt{ku} + 1$$

**FORTRAN IMPLEMENTATION: (illustrative purposes only)**

```
integer        maxn
parameter    ( maxn = 1 000 )

real*8         a(maxn,maxn)

integer        kl,       ku,       ldab
parameter    ( kl = 2,   ku = 1,   ldab = 6 )
real*8         ab(ldab,maxn)
```

Assume that we have defined the (full) matrix `a` (i.e. set `a(i,j) for i, j` $= 1, \ldots, n$), then the following code will pack the coefficients using the `LAPACK` band-storage scheme.

```
do j = 1 , n
   do i = max( 1 , j - ku ) , min( n , j + kl )

      ab( kl + ku + 1 + i - j , j ) = a( i , j )

   end do
end do
```

Note that the above indexing guarantees that (a) elements remain in their original columns and (b) diagonals (`i - j` $=$ `constant`) are stored in rows of `ab`. As we will see, in a real application, we will generally compute $A_{ij}$ and load the value into `ab(kl+ku+1+i-j,j)` without explicitly storing `a(i,j)`. However, the "index translation" remains the same.

## LAPACK GENERAL BANDED SOLVER: DGBSV

```
subroutine dgbsv( n, kl, ku, nrhs, ab, ldab, ipiv, b, ldb, info )

   integer  n             ! size of system
   integer  kl, ku        ! # of lower and upper diagonals
   integer  nrhs          ! Number of right hand sides (nominally 1)
   integer  ldab          ! Leading dimension of ab
                          ! Note: ldab must be .ge. 2 * kl + ku + 1
   real*8   ab(ldab,*)    ! Matrix in LAPACK band-storage form
   integer  ldb           ! Leading dimension of rhs array
   real*8   b(ldb,*)      ! Right hand side(s) on input
                          ! Solution(s) on output
   integer  ipiv(n)       ! Storage for pivot vector
   integer  info          ! Usual LAPACK return code
```

## EXAMPLE: 1D BVP

We again consider

$$u''(x) = f(x) \qquad 0 \le x \le 1 \qquad u(0), u(1) \text{ specified}$$

and adopt the same uniform discrete domain as previously, but now use the $O(h^4)$ (five-term) approximation to $u''(x)$ where possible, and the $O(h^2)$ (three-term) approximation elsewhere. Then we have the following system of equations for the $n$ unknowns, $u_i$:

$$\begin{aligned}
u_1 &= u(0) & i &= 1 & (1)\\
h^{-2}(u_{i-1} - 2u_i + u_{i+1}) &= f_i & i &= 2 & (2)\\
(12h^2)^{-1}(-u_{i-2} + 16u_{i-1} - 30u_i + 16u_{i+1} - u_{i+2}) &= f_i & i &= 3, \ldots n-2 & (3)\\
h^{-2}(u_{i-1} - 2u_i + u_{i+1}) &= f_i & i &= n-1 & (4)\\
u_n &= u(1) & i &= n & (5)
\end{aligned}$$

Clearly, this set of equations is of the form

$$\mathbf{Au} = \mathbf{b}$$

where $\mathbf{A}$ is an $n \times n$ pentadiagonal matrix.

Having derived the linear system we must solve, the only slightly challenging part of the exercise is correctly setting up the array `ab` in `LAPACK` banded form. Organizationally, this process can be expedited by noting that all of the equations (1-5) can be written in the form:

$$\sum_{k=-2}^{k=2} c_k u_{i+k} = b_i$$

where, for each of the 5 sub-cases, the $c_k$ are constant coefficients, and it is to be understood that the limits on the $k$-sum are to be changed if they result in references to $u_{i+k}$ such that $i + k < 1$ or $i + k > n$.

Specifically, we have

$$c = [\ 0\ ,\ 0\ ,\ 1\ ,\ 0\ ,\ 0\ ] \qquad (1,5)$$

$$c = h^{-2}\ [\ 0\ ,\ 1\ ,\ -2\ ,\ 1\ ,\ 0\ ] \qquad (2,4)$$

$$c = (12h^2)^{-1}\ [\ -1\ ,\ 16\ ,\ -30\ ,\ 16\ ,\ -1\ ] \qquad (3)$$

Refer to the program source for `bvp1d4.f` (available on-line) for full implementation details.