

```

50      CONTINUE
C
C      ... COMPUTE THE NEW APPROXIMATION TO VARIABLE X(I)
C      TEMP = (A(I, NPLUS1) - SUM) / A(I,I)
C
C      ... AT THE END OF A SWEEP OF ALL EQUATIONS, THE FOLLOWING
C      ... STATEMENT WILL HAVE PUT LARGEST RESIDUAL IN RESID
C      IF (ABS(TEMP - X(I)) .GT. RESID ) RESID = ABS(TEMP - X(I))
C
C      ... STORE NEW APPROXIMATION TO VARIABLE X(I)
C      X(I) = TEMP
60      CONTINUE
C
C      ... ONE SWEEP HAS NOW BEEN COMPLETED -- PRINT VARIABLES
C      WRITE (6, 140) (X(K), K = 1, N)
C
C      ... IF LARGEST RESIDUAL LESS THAN EPSILON, PROCESS HAS CONVERGED
C      IF ( RESID .LT. EPSILON ) STOP
70      CONTINUE
C
C IF THIS OUTER DO IS EVER SATISFIED, MORE THAN MAXIT ITERATIONS WOULD
C BE NEEDED FOR CONVERGENCE -- WRITE ERROR COMMENT AND GIVE UP
C      WRITE (6, 150) MAXIT
C      STOP
C
C
100     FORMAT (2I2, 2F10.0)
110     FORMAT (1X, 'ERROR IN CARD WITH I = ', I2, ', J = ', I2,
1      ' ', VALUE = ', 1PE14.6)
120     FORMAT ('0', 'DECK CONTAINED TOO MANY CARDS')
130     FORMAT ('0', 'ERRORS FOUND IN ', I4, ' DATA CARDS - JOB ABORTED')
140     FORMAT ('0', 8F12.5)
150     FORMAT ('0', 'PROCESS DID NOT CONVERGE IN', I4, ' ITERATIONS')
      END

```

In almost every way, this is an excellent program. It validates its input data. It uses multiple loop exits to detect errors. In several places, the authors have sacrificed a tiny amount of computation time by re-testing a condition, to avoid extra labels and GOTO's.

And it is thoroughly commented and neatly formatted. Notice that even the data is commented. One of the most effective ways to document a program is simply to describe the data layout in detail. If you can specify for each important variable what values it can assume and how it gets changed, you have gone a long way to describing the program. (The checker-playing subroutine we looked at in Chapter 4 is another good example.)

Document your data layouts.

All in all, the code above is a model of programming style.

But there are a few difficulties. To begin with, what about modularity? When a single routine sprawls over several pages, it is hard to follow. Since well over half of the actual code is concerned with validating the data, this could profitably be made a separate input function, as we suggested in Chapters 4 and 5. Then the main program could read

```
IF (INPUT(N, A, MAXIT, EPSILON) .EQ. ERROR) STOP
```

The modularization would have the advantage that each part of the program would fit comfortably on one page. (In fairness, we should observe that the textbook from