1970

Computer Programming

Gary Rice
Ouachita Baptist University

COMPUTER PROGRAMMING

Presented to

Dr. A. R. Nisbet

In Partial Fulfillment

of the Requirements of

Honors H491

---

by

Gary Rice

May 22, 1970

The chief advantage of the digital computer is that it can be instructed to perform complex or repetitive arithmetical operations in a very short period of time. Any sequence of operations which can be fully analyzed can theoretically be done by a computer. The method of instruction takes the form of various precisely defined computer languages. The programs to be discussed here were written in a Fortran language, Fortran being a contraction of Formula Translation. There are at least four variations of Fortran, but the differences are relatively minor. Fortran is basically intended for scientific and engineering purposes. The programs discussed here were in one case the calculation of electron density in an atomic orbital, and in the other three cases methods of solving systems of equations. No programs are attached to this paper, as they are rather long and not self-explanatory, and were turned in separately.

The atomic orbital calculated was a 2p orbital in a hydrogen atom; the probability of finding an electron at a point in a plane is given by ψ^2, which is a function of an angle from a base line and the distance from the nucleus:

$$\psi^2 = \frac{1}{32\pi a_0^5} e^{-r/a_0} \cos^2\theta$$

To properly define the shape of the orbital, it is necessary to know the probability at a large number of points. The digital computer is admirably suited for calculating these values, as it can do so rapidly and accurately and output the answers in tabulated form.

The three programs for solutions of simultaneous equations deal with systems such as:

$$\begin{aligned} a_{11}\,x + a_{12}\,y + a_{13}\,z &= c_1 \\ a_{21}\,x + a_{22}\,y + a_{23}\,z &= c_2 \\ a_{31}\,x + a_{32}\,y + a_{33}\,z &= c_3 \end{aligned} \qquad \text{(I)}$$

One method of solving such a system is to rearrange the equations in the form:

$$\begin{aligned} x &= 1/a_{11}\,(c_1 - a_{12}\,y - a_{13}\,z) \\ y &= 1/a_{22}\,(c_2 - a_{21}\,x - a_{23}\,z) \\ z &= 1/a_{33}\,(c_3 - a_{31}\,x - a_{32}\,y) \end{aligned} \qquad \text{(II)}$$

Then, an arbitrary value is assigned to each variable, and a new value of x is calculated from the first equation. This is inserted in the second equation and a new y calculated. Both x and y are used to find a new z, which is put back in the first equation and x recalculated. The process can be continued until the variables are within any desired range of the true value. The method works for any system which has a singular solution so long as no main diagonal coefficient in (I) is zero. It will sometimes happen that the calculated variables will diverge from the true values, but this difficulty, as well as the one involving the zero diagonal element, can be remedied by rearranging the order of the equations.

The other two methods depend on matrix algebra. System (I) can be rewritten in matrix form as:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad \text{(III)}$$

which can be abbreviated as:

$$AX = C \qquad \text{(IV)}$$

Now, any nonsingular matrix A has an inverse A^{-1}, such that:

$$AA^{-1} = I \qquad (V)$$

where I is the identity matrix, with the property IY = Y for any matrix Y which has the same number of rows as I has columns. The inverse matrix A^{-1} can be found by the Gauss-Jordan inversion technique, which involves augmenting the matrix A with an identity matrix, then performing elementary row operations on the entire matrix until the left-hand side is converted into an identity matrix, whereupon the right-hand half of the matrix will be A^{-1}. Then, multiplying both sides of (IV) by A^{-1}, and using (V):

$$A^{-1}AX = IX = X = AC$$

The product AC is a column matrix whose members are just the solution of the system.

The third method of solving equations can be considered as derived from the Gauss-Jordan method and from the echelon form of a system used in linear algebraic work. For the present, it might as well be referred to as the echelon method. It is more generally applicable than the two methods discusses above. In particular, note that the inversion method fails when A has no inverse, and when $c_i = 0$ for all c_i. Also, neither of the above methods can be used for a system with an unequal number of equations and unknowns. The echelon method begins by taking the matrix A and augmenting it with the column matrix C. The augmented matrix is then carried into an identity matrix by operations just like those used in the Gauss-Jordan method, with the exception that the last column on the

right is not changed to a column of zeroes. If the left-hand portion is converted perfectly to a diagonal unit matrix, then the solution of the system is the column of constants which remains at the right. However, should it be found impossible to prevent having a zero appear on the main diagonal, it is left there, and the numbers remaining in the column above it are also left (the elements below it are already zero, or it would not be necessary to leave the zero in the diagonal). Suppose that the matrix on which the operations are begun is the matrix obtained by augmenting A by C (from (III)), and that the final matrix obtained is:

$$\begin{bmatrix} 1 & 3 & 0 & b_1 \\ 0 & 0 & 1 & b_2 \\ 0 & 0 & 0 & b_3 \end{bmatrix} \qquad \text{(VI)}$$

The system may be seen not to have a specific solution, but it does have a general solution if, and only if, $b_3 = 0$. If it is zero, it simply means that the original system contained a redundant equation. In this case, part of the solution is definite: $z = b_2$. To find the rest of the solution, which is general, some arbitrary choice must be made for one variable. Suppose the choice made is $y = r$, where r is any real number. Then, $x = b_1 - 3r$. The method can also be used when the number of equations and unknowns is unequal. If there are more equations than unknowns, the "extra" equations will appear as additional lines at the bottom of the matrix as written. These rows must be all zeroes in the final matrix if the system is to have any solution. When there are more unknowns than equations, it will be necessary to make one

or more arbitrary choices, as was done for (VI).. The computer is of course incapable of making these choices, but it can perform all the operations on the matrix and output it in a form where very little work remains to be done on it. This last method should be applicable to any system of linear equations.

REFINEMENT 1

H519.4
R1c
Suppl.

```
***  C  SOLUTION OF 5 EQUATIONS AND UNKNOWNS BY
     2  GUESS AND REFINE

        DIMENSION A(5,5), C(5), D(5), E(5), F(5), X(5)
100     FORMAT ('1', 34X, 'REARRANGE DATA CARDS.  '
     2  'DIAGONAL ELEMENT', I2, ' IS ZERO.')
101     FORMAT (6F10.2)
102     FORMAT ('1', 39X, ' X(1)', 6X, 'X(2)', 6X,
     2  'X(3)', 6X, 'X(4)', 6X, 'X(5)'/)
103     FORMAT (35X, 5F10.2)
104     FORMAT ('0', 21X, 'NO SOLUTION BY THIS'
     2  ' METHOD.  EQUATIONS ANALOGOUS TO '
     3  'MUTUAL PERPENDICULARITY.')
105     FORMAT ('0', 31X, 'REARRANGE DATA CARDS.  '
     2  'EQUATIONS DIVERGE IN THIS ORDER.')

        READ (2, 101) ((A(I,J), J=1,5), C(I), I=1,5)
        I=0
1       I=I+1
        IF (A(I,I)) 2,3,2
2       IF (I-5) 1,4,4
3       WRITE (3,101) I
        CALL EXIT
4       DO 5 I=1,5
        X(I)=1.
        D(I) = A(I,1) + A(I,2)+A(I,3)+A(I,4)+A(I,5)
        E(I) = ABS (((D(I) - C(I))/C(I)) * 100.)
5       CONTINUE
        WRITE (3, 102)
        WRITE (3, 103) (X(I), I=1,5)
```

```
      N=0
    6 DØ 7 K=1,5
      N=N+1
      X(K) = (1./A(K,K)) * (C(K)- A(K,1)* X(1)- A(K,2)* X(2)
     2    - A(K,3) * X(3) - A(K,4)* X(4) -A(K,5) * X(5)
     3    + A(K,K) * X(K))
    7 CØNTINUE
      WRITE (3,103) (X(I), I=1,5)
      DØ 8 I=1,5
      D(I) = A(I,1) * X(1) + A(I,2) * X(2) + A(I,3) * X(3)
     2    + A(I,4) * X(4) + A(I,5) * X(5)
      F(I) = ABS ((D(I) - C(I))/C(I)) * 100.)
    8 CØNTINUE
      I=0
    9 I= I+1
      IF (I-5) 10, 10, 20
   10 IF (F(I) - E(I)) 11, 13, 14
   11 IF (F(I) -1.0) 9, 9, 12
   12 IF (N-50) 6, 6, 20
   13 WRITE (3,104)
      GØ TØ 20
   14 WRITE (3,105)
   20 CALL EXIT
      END
```

```
** C   GAUSS-JORDAN  MATRIX  INVERSION
     DIMENSION   A(8,16), C(16)
1    READ (2,100) N
     WRITE (3,99)
     WRITE (3,102)
     GO TO (150, 150, 150, 2, 3, 4, 5, 6) N
2    READ (2,101)      ((A(I,J), J=1,4), I=1,4)
     WRITE (3,103)     ((A(I,J), J=1,4), I=1,4)
     GO TO 7
3    READ (2,101)      ((A(I,J), J=1,5), I=1,5)
     WRITE (3,104)     ((A(I,J), J=1,5), I=1,5)
     GO TO 7
4    READ (2,101)      ((A(I,J), J=1,6), I=1,6)
     WRITE (3,105)     ((A(I,J), J=1,6), I=1,6)
     GO TO 7
5    READ (2,101)      ((A(I,J), J=1,7), I=1,7)
     WRITE (3,106)     ((A(I,J), J=1,7), I=1,7)
     GO TO 7
6    READ (2,101)      ((A(I,J), J=1,8), I=1,8)
     WRITE (3,107)     ((A(I,J), J=1,8), I=1,8)
7    L = N+1
     M = 2*N
     DO 8   I = 1, N
     DO 8   J = L, M
     A(I,J) = 0.0
8    CONTINUE
     DO 9   I = 1, N
     J = N+I
     A(I,J) = 1.0
9    CONTINUE
```

```
      I=0
10    I=I+1
      IF (N-I) 11, 12, 17
11    WRITE (3,108)
      GØ TØ (150, 150, 150, 12, 13, 14, 15, 16)
12    WRITE (3,103) ((A(I,J), J=5,8), I=1,4)
      GØ TØ 1
13    WRITE (3,104) ((A(I,J), J=6,10), I=1,5)
      GØ TØ 1
14    WRITE (3,105) ((A(I,J), J=7,12), I=1,6)
      GØ TØ 1
15    WRITE (3,106) ((A(I,J), J=8,14), I=1,7)
      GØ TØ 1
16    WRITE (3,107) ((A(I,J), J=9,16), I=1,8)
      GØ TØ 1
17    IF (A(I,I)) 18,25,18
18    B=1./A(I,I)
      DØ 19 J=1,M
      A(I,J)=B*A(I,J)
19    CONTINUE
      K=0
20    K=K+1
      IF (I-K) 21,20,21
21    IF (N-K) 10,22,22
22    IF (A(K,I)) 23,20,23
23    B=A(K,I)
      DØ 24 J=1,M
      A(K,J)=A(K,J)-B*A(I,J)
24    CØNTINUE
      GØ TØ 20
```

```
25      L2 = I
26      L2 = L2+1
        IF (N-L2) 30, 27, 27
27      IF (A(L2, I)) 28, 26, 28
28      DO 29 J = I, M
        C(J) = A(I,J)
        A(I,J) = A(L2, J)
        A(L2,J) = C(J)
29      CONTINUE
        GO TO 18
30      WRITE (3, 109)
        GO TO (150, 150, 150, 31, 32, 33, 34, 35) N
31      WRITE (3, 110) ((A (I,J), J = 1, 8), I = 1, 4)
        GO TO 1
32      WRITE (3, 111) ((A (I,J), J = 1, 10), I = 1, 5)
        GO TO 1
33      WRITE (3, 112) ((A (I,J), J = 1, 12), I = 1, 6)
        GO TO 1
34      WRITE (3, 113) ((A (I,J), J = 1, 14), I = 1, 7)
        GO TO 1
35      WRITE (3, 114) ((A (I,J), J = 1, 16), I = 1, 8)
        GO TO 1
** C    LAST DATA CARD SETS N=1, TERMINATES PROGRAM
150     CALL EXIT
        END
```

```
   99    FORMAT ('1')
  100    FORMAT (I2)
  101    FORMAT (8F 10.2)
  102    FORMAT (50X, 'MATRIX TO BE INVERTED'//)
  103    FORMAT (36X, 4F 12,4 )
  104    FORMAT (30X, 5F 12.4 )
  105    FORMAT (24X, 6F 12.4)
  106    FORMAT (18X, 7F 12.4)
  107    FORMAT (12X, 8F 12.4)
  108    FORMAT ('0', 51X, 'INVERSE OF MATRIX'//)
  109    FORMAT ('0', 47X, 'METHOD FAILS AT THIS POINT'//)
  110    FORMAT (8X, 8E 12.4)
  111    FORMAT (1X, 10F 12.4)
  112    FORMAT (1X, 12F 10.4)
  113    FORMAT (2X, 7F 7.2, 7F10.4)
  114    FORMAT (1X, 8F 6.2, 8F 9.4)
**  C    INSERT FORMATS AFTER DIMENSION
```

```
** C   MODIFIED ECHELON FORM
      DIMENSION A(8,9) C(9)
1     READ (2,100) N
      WRITE (3,99)
      WRITE (3,102)
      GO TO (150,150,150,2,3,4,5,6) N
2     READ (2,101) ((A(I,J), J=1,5), I=1,4)
      WRITE (3,103) ((A(I,J), J=1,5), I=1,4)
      GO TO 7
3     READ (2,101) ((A(I,J), J=1,6), I=1,5)
      WRITE (3,104) ((A(I,J), J=1,6), I=1,5)
      GO TO 7
4     READ (2,101) ((A(I,J), J=1,7), I=1,6)
      WRITE (3,105) ((A(I,J), J=1,7), I=1,6)
      GO TO 7
5     READ (2,101) ((A(I,J), J=1,8), I=1,7)
      WRITE (3,106) ((A(I,J), J=1,8), I=1,7)
      GO TO 7
6     READ (2,101) ((A(I,J), J=1,9), I=1,8)
      WRITE (3,107) ((A(I,J), J=1,9), I=1,8)
7     I=0 <--- M=N+1
8     I=I+1
      IF (N-I) 22,9,9
9     IF (A(I,I)) 10,17,10
10    B=1./A(I,I)
      DO 11 J=1,M
      A(I,J)=B*A(I,J)
11    CONTINUE
      K=0
12    K=K+1
```

```
      IF (I-K) 13, 12, 13
13    IF (N-K) 8, 14, 14
14    IF (A(K,I)) 15, 12, 15
15    B = A(K,I)
      DO 16 J = I, M
      A(K,J) = A(K,J) - B * A(I,J)
16    CONTINUE
      GO TO 12
17    L = I
18    L = L+1
      IF (N-L) 8, 19, 19
19    IF (A(L,I)) 20, 18, 20
20    DO 21 J = I, M
      C(J) = A(I,J)
      A(I,J) = A(L,J)
      A(L,J) = C(J)
21    CONTINUE
      GO TO 10
22    WRITE (3, 108)
      GO TO (150, 150, 150, 23, 24, 25, 26, 27) N
23    WRITE (3, 103) ((A(I,J), J = 1, 5), I = 1, 4)
      GO TO 1
24    WRITE (3, 104) ((A(I,J), J = 1, 6), I = 1, 5)
      GO TO 1
25    WRITE (3, 105) ((A(I,J), J = 1, 7), I = 1, 6)
      GO TO 1
26    WRITE (3, 106) ((A(I,J), J = 1, 8), I = 1, 7)
      GO TO 1
27    WRITE (3, 107) ((A(I,J), J = 1, 9), I = 1, 8)
      GO TO 1
150   CALL EXIT
      END
```

```
 99    FORMAT ('1')
100    FORMAT (I2)
101    FORMAT (8 F 10.2)
102    FORMAT (50X,'MATRIX FORM OF SYSTEM'//)
103    FORMAT ( 30X,  5F 12.4 )
104    FORMAT (24 X,  6F 12.4 )
105    FORMAT (18X,  7F 12.4 )
106    FORMAT (12X,  8F 12.4 )
107    FORMAT ( 6X,  9F 12.4 )
108    FORMAT ('0', 44X, 'SIMPLIFIED MATRIX FORM OF SYSTEM'//)
XX  C  INSERT FORMATS AFTER DIMENSION
```