UNITED ARAB REPUBLIC

THE INSTITUTE OF NATIONAL PLANNING

التخطيط القومي

Memo. No. 406

Notes on Interpolation Formulae

By

Dr. S. Hamid

and

Fayza H. Omar

CAIRO

Part I.

The Lagrangian Interpolation Formula.

Section 1. Introduction:

Assume we are given n+1 values of a dependent variable "f" corresponding to n+1 values of independent variable "a". Denote these variables respectively

$$f(a_j) \;,\quad a_j \qquad \text{for } j=1,2,\ldots,n+1. \qquad \ldots (1)$$

Then it is possible to construct a polynomial of degree n.

$$f(x)=A_0 + A_1 x + A_2 x^2 + \ldots + A_n x^n. \qquad \ldots (2)$$

Such that:

$$f(x)= f(a_j) \quad \text{at} \quad x = a_j. \qquad \ldots (3)$$

For numerical application, a more convenient form for the polynomial f(x) is needed, which is derived in the following section.

Section 2. The Lagrangian Interpolation Formula:

From equations 2 & 3 we get the following n+1 equations

$$f(a_j) = A_0 + A_1 a_j + A_2 a_j^2 + \ldots + A_n a_j^n .$$

$$j = 1,2, \ldots,n+1. \qquad \ldots (4)$$

The system of n+2 equations given by equation (2) & (4) are consistent if & only if

$$\begin{vmatrix} f(x) & 1 & x & x^2 & \cdots & x^n \\ f(a_1) & 1 & a_1 & a_1^2 & \cdots & a_1^n \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ f(a_j) & 1 & a_j & a_j^2 & \cdots & a_j^n \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ f(a_{n+1}) & 1 & a_{n+1} & a_{n+1}^2 & \cdots & a_{n+1}^n \end{vmatrix} = 0. \qquad \ldots(5)$$

Expanding the above determinant columnwise we have

$$f(x).\, \Delta_o = \sum_{j=1}^{n+1} f(a_j).\, \Delta_j(x).\, (-1)^{j+1}. \qquad \ldots(6)$$

Where

$$\Delta_o = \begin{vmatrix} 1 & a_1 & a_1^2 & \cdots & a_1^n \\ 1 & a_2 & a_2^2 & \cdots & a_n^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_j & a_j^2 & \cdots & a_j^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{n+1} & a_{n+1}^2 & \cdots & a_{n+1}^n \end{vmatrix} \qquad \ldots(7)$$

and Δ_j (x) is the [illegible] in the determinant given in the right-hand side of equation 6.

For example:

$$\Delta_3 = \begin{vmatrix} 1 & x & x^2 & \cdots & x^n \\ 1 & a_1 & a_1^2 & \cdots & a_1^n \\ 1 & a_2 & a_2^2 & \cdots & a_2^n \\ 1 & a_4 & a_4^2 & \cdots & a_4^n \\ 1 & a_5 & a_5^2 & \cdots & a_5^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{n+1} & a_{n+1}^2 & \cdots & a_{n+1}^n \end{vmatrix} \qquad \ldots (8)$$

Notice that Δ_j (x) is a polynomial of degree n, and hence can in general be expressed in the form

$$\Delta_j(x) = C_j(x - c_1)\,(x - c_2)\,\ldots\,(x - c_n). \qquad \ldots (9)$$

From the determinantal definition of Δ_j (x), it is seen that

$$\Delta_j(x) = 0 \qquad \ldots (10)$$

at the n values of x given by

$$x = a_i \qquad i \neq j \qquad \ldots (11)$$

From (9) , (10) & (11) we can put

$$c_i = a_i \qquad i < j$$

$$\qquad\qquad \ldots (12)$$

$$c_i = a_{i+1} \qquad i \geqslant j$$

Or in other words

$$\Delta_j (x) = \frac{C_j}{(x-a_j)} \cdot \prod_{j=1}^{n+1} (x - a_j) \qquad \ldots (13)$$

$$= \frac{C_j}{(x-a_j)} \cdot P_n(x) \qquad \ldots (14)$$

where

$$P_n(x) = \prod_{j=1}^{n+1} (x - a_j) \qquad \ldots (15)$$

Consider the determinant Δ_0. By analogy it can be expanded in a similar way as $\Delta_j(x)$ and this can take the following form:

$$\Delta_0 = C_j \prod_{i=1)j(n+1} (a_j - a_i) \qquad \ldots (16)$$

where $i = 1) j (n+1$ stands for $i = 1, 2, \ldots, j-1, j+1, \ldots, n+1$.

From the above definition of $p_n(x)$ it can be seen that

$$\Delta_0 = C_j \left[\frac{d\, P_n(x)}{dx} \right]_{x = a_j} = C_j \; P_n'(a_j) \qquad \ldots (17)$$

From (14) & (17) we can put

$$L_j(x) = \frac{\Delta_j(x)}{\Delta_0} = \frac{1}{(x-a_j)} \cdot \frac{P_n(x)}{P_n'(a_j)} \qquad \ldots (18)$$

Substituting the above equation in equation (6) defining the function f(x), we obtain the following convenient expression for the function f(x).

$$f(x) = \sum_{j=1}^{n+1} L_j(x) \; f(a_j) \qquad \ldots (19)$$

This formula is the wellknown "Lagrangian Interpolation Formula".[1]

Section 3 : Computation procedure:

The above Lagrangian interpolation formula had been programmed on the I.B.M. 1620 using Fortran Language.

Section 3.1 gives the symbols used in the Fortran program and the way the data must be prapared.

Section 3.2 gives the block diagram.

Section 3.3 gives the program itself in the Fortran Language.

Section 3.4 gives a numerical example.

The program available can handle interpolation by Lagrangian formula provided the number of points at which the function is given is less or equal to 100. If we have more points, equal in value to the integer J1, then we must change the first dimension statment in the source program to the following statment

```
DIMENSION  A(J1), F(J1)
```

and of course we have to compile the program again to get the object program.

---

1) See Kopal, Zdenek
Numerical analysis, 2nd. ed.   N.Y., Wiley, 1961.

Section 3.1 : Symbols used in the program and the preparation of data.

The formula is

$$f(x) = \sum_{j=1}^{n+1} L_j(x)\, f(a_j)$$

where

$$L_j(x) = \frac{(x-a_1)\,(x-a_2)\,\ldots\,(x-a_{j-1})(x-a_{j+1})\ldots(x-a_{n+1})}{(a_j-a_1)(a_j-a_2)\ldots(a_j-a_{j-1})(a_j-a_{j+1})\ldots(a_j-a_{n+1})}$$

Symbols in Theory	Corresponding Symbols in Fortran
$n+1$	N1
j	J
a_j	A(J)
$f(a_j)$	F(J)
x	X
$f(x)$	FX

The preparation of data:

The first card must contain the number N1 of values at which the function is given and the code of the process as following:

N1	: Colomn	1→3	xxx	integer form.
Code	: Colomn	4→10	xx.xxxx	floating form.

After the first card we have N1 cards each corresponding to one given value of the function.

The data in these N1 cards are as following:

J	: colomn	1→3	xxx	integer form
A(J)	: colomn	4→17	±x.xxxxxxxE±xx	E form
F(J)	: colomn	18→31	±x.xxxxxxxE±xx	E form

After the N1 cards, we have cards, each corresponding to the value of the argument X at which we want to determine our function.

The data in these cards are as following:

X : colomn 1→14 ±x.xxxxxxxE±xx E form

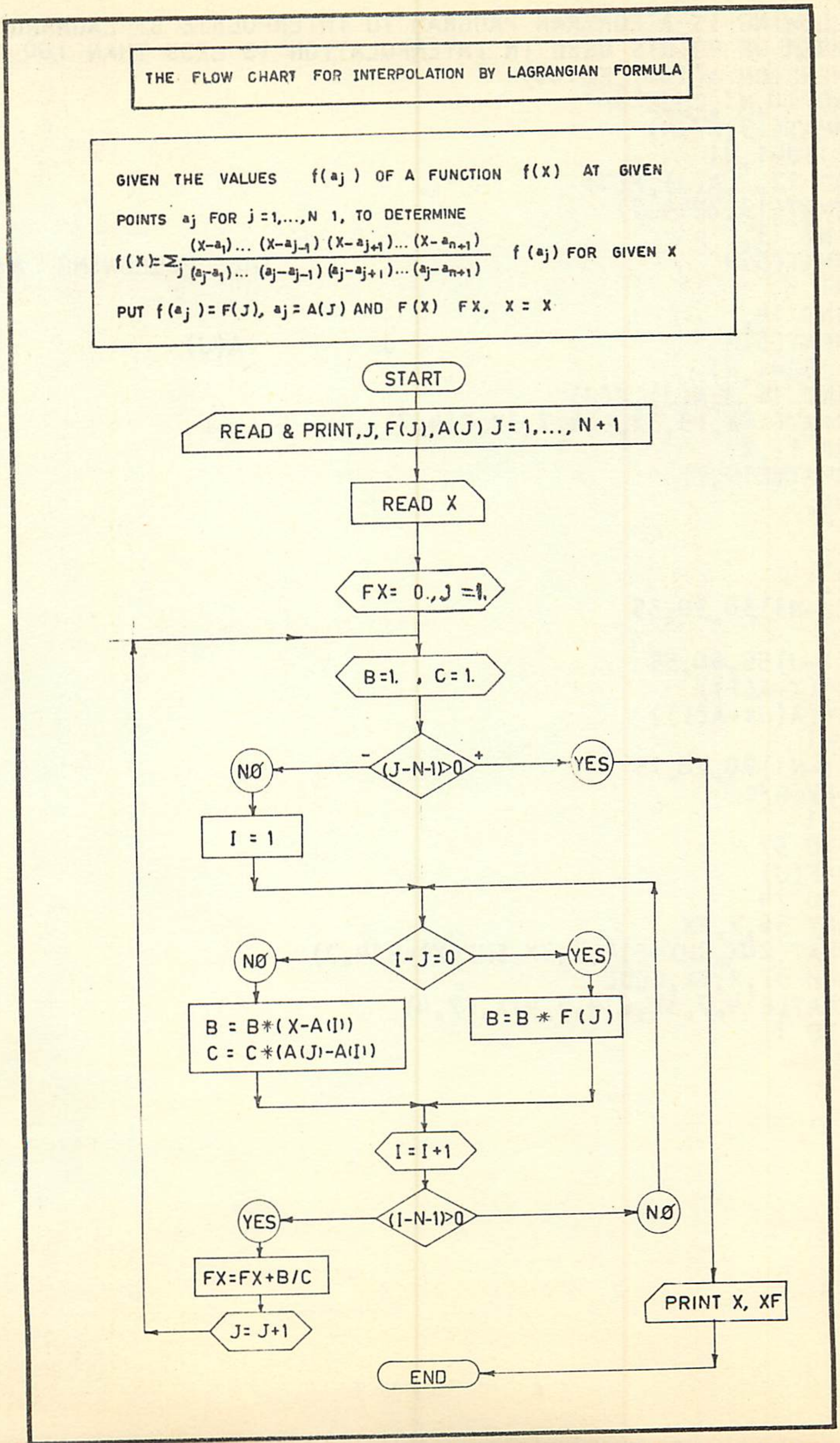
So if our problem is to interpolation for M1 values, given N1 values of the function, then our data cards will be

1 + N1 + M1 .

Section 3.2

Block diagram

of Lagrangian.



```
C THE FOLLOWING IS A FORTRAN PROGRAM TO INTERPOLATE BY LAGRANGIAN FORMUL
C THE NUMBER OF POINTS USED IN INTERPOLATION IS LESS THAN 100
      DIMENSION A(100),F(100)
      READ 10,N1,CODE
   10 FORMAT(I3,F7.4)
      DO 11J=1,N1
   11 READ 12,J,A(J),F(J)
   12 FORMAT(I3,2E14.7)
      PRINT 13,
   13 FORMAT(59H                                            THE  FOLLOWING  ARE  THE  D
     XATA)
      PRINT 14,
   14 FORMAT(52H                             J           A(J)               F(J))
      DO 15J=1,N1
   15 PRINT 16,J,A(J),F(J)
   16 FORMAT(20X,I3,3X,E14.7,3X,E14.7)
    1 READ 17,X
   17 FORMAT(E14.7)
      FX=0.
      J=1
   57 B=1.
      C=1.
      IF(J-N1)30,30,35
   30 I=1
   70 IF(I-J)55,60,55
   55 B=B*(X-A(I))
      C=C*(A(J)-A(I))
   74 I=I+1
      IF(I-N1)70,70,75
   75 FX=FX+B/C
      J=J+1
      GO TO 57
   60 B=B*F(J)
      GO TO 74
   35 PRINT 36,X,FX
   36 FORMAT(20X,2HX=E14.7,3X,5HF(X)=E14.7)
      PUNCH 37,X,FX,CODE
   37 FORMAT(E14.7,3X,E14.7,41X,F7.4)
      GO TO 1
      END
```

THE FOLLOWING ARE THE DATA

J	A(J)	F(J)
1	1.0000000E+00	0.0000000E-99
2	9.0380000E-01	2.2030000E-01
3	8.0920000E-01	4.2130000E-01
4	7.2870000E-01	5.7930000E-01
5	6.6790000E-01	6.7560000E-01
6	5.8470000E-01	7.6730000E-01
7	4.8290000E-01	8.5650000E-01
8	3.7100000E-01	9.2660000E-01
9	2.4800000E-01	9.7180000E-01
10	7.6500000E-02	9.9450000E-01

X= 5.0000000E-01 F(X)= 8.4171143E-01

Part II.

Aitken's Interpolation Formula.

Section 1. Introduction:

Aitken's Algorithim for the Lagrangian Interpolation Formula:

There is an algorithim, due to Aitken, for computing Lagrangian interpolation formula, which is suitable for digital computers & even desk calculators. Given the function $f(a_j)$ at the n+1 points a_j $(j=1,2,\ldots,n+1)$, Aitken's algorithim for computing the lagrangian interpolation formula f(x) whose numerical values at that is satisfied at the point $x=a_j(j=1,\ldots, n+1)$ coincides with the values $f(a_j)$, proceeds as following:

The Lagrangian interpolation formula of degree r :

$$f(x \mid a_1, a_2, \ldots, a_r, a_t) \qquad n+1 \geqslant t \geqslant r+1 \qquad \ldots (1)$$

which numerically is equal to the function f(x) at the points:

$$x=a_1, a_2, \ldots, a_r, a_t. \qquad \ldots (2)$$

can be obtained from the two Lagrangian interpolation formulae of degree r-1:

$$f(x \mid a_1, a_2, \ldots, a_{r-1}, a_r) \qquad \ldots (3)$$

$$\& \qquad f(x \mid a_1, a_2, \ldots, a_{r-1}, a_t) \qquad \ldots (4)$$

that passes respectively through the two following sets of points:

$$a_1, a_2, \ldots, a_{r-1}, a_r . \qquad \ldots (5)$$

$$\& \qquad a_1, a_2, \ldots, a_{r-1}, a_t . \qquad \ldots (6)$$

Again any one of the above Lagrangian formulae expression (3) or expression (4), can be obtained from another two Lagrangian interpolation formulae of degree r-2.

For example the interpolation formula:

$$f(x \mid a_1, a_2, \ldots, a_{r-1}, a_t) \quad \ldots (7)$$

can be obtained from the following two formulae:

$$f(x \mid a_1, a_2, \ldots, a_{r-2}, a_{r-1}) \quad \ldots (8)$$

$$\& \quad f(x \mid a_1, a_2, \ldots, a_{r-2}, a_t) \quad \ldots (9)$$

& so on.

It is obvious how this is extended foreward till we construct the required Lagrangian interpolation formula

$$f(x \mid a_1, a_2, \ldots, a_{n+1}) \quad \ldots (10)$$

Again, it can be seen how this algorithm is extended backward till we can start it, when given

$$f(a_j) \ \& \ a_j \quad \text{for } j=1, 2, \ldots, n+1. \quad \ldots (11)$$

Section 2. Proof of the Algorithm:

To drive the formula relating expression (1) with expressions (3) & (4) we proceed as following:

By definition we have

$$f(x \mid a_1, a_2, \ldots, a_{r-1}, a_t) = \sum_{j=1 \to r} L_j^{(1)}(x)\, f(a_j) \quad \ldots (12)$$

$$\& \quad f(x \mid a_1, a_2, \ldots, a_{r-1}, a_t) = \sum_{j=1 \to r-1, t} L_j^{(2)}(x)\, f(a_j) \quad \ldots (13)$$

where

$$L_j^{(1)}(x)= \frac{(x-a_1)(x-a_2)\ \ldots\ (x-a_{j-1})(x-a_{j+1})\ldots(x-a_{r-1})(x-a_r)}{(a_j-a_1)(a_j-a_2)\ldots(a_j-a_{j-1})(a_j-a_{j+1})\ldots(a_j-a_{r-1})(a_j-a_r)} \quad \ldots (14)$$

while

$$L_j^{(2)}(x)= \frac{(x-a_1)(x-a_2)\ldots(x-a_{j-1})(x-a_{j+1})\ldots(x-a_{r-1})(x-a_t)}{(a_j-a_1)(a_j-a_2)\ldots(a_j-a_{j-1})(a_j-a_{j+1})\ldots(a_j-a_{r-1})(a_j-a_t)} \quad \ldots (15)$$

Multiply equation (13) by $(x-a_r)$ & equations (12) by $(x-a_t)$ & substracting we have

$$f(x \mid a_1, a_2, \ldots, a_{r-1}, a_t)(x-a_r) - f(x \mid a_1, \ldots, a_{r-1}, a_r)(x-a_t)$$

$$= \sum_{j=1\to r,t} L_j(x)\ f(a_j) \left[(a_j-a_r) - (a_j-a_t) \right] \quad \ldots (16)$$

$$= (a_t-a_r)\ f\ (x \mid a_1, a_2, \ldots, a_{r-1}, a_r, a_t) \quad \ldots (17)$$

where

$$L_j(x)= \frac{(x-a_1)(x-a_2)\ldots(x-a_{j-1})(x-a_{j-1})\ldots(x-a_r)(x-a_t)}{(a_j-a_1)(a_j-a_2)\ldots(a_j-a_{j-1})(a_j-a_{j+1})\ldots(a_j-a_r)(a_j-a_t)}$$

Equation (17) can be rewritten in the following form:

$$f(x \mid a_1, a_2, \ldots, a_{r-1}, a_r, a_t) = \begin{vmatrix} f(x|a_1, a_2, \ldots, a_{r-1}, a_r) & a_r - x \\ f(x|a_1, a_2, \ldots, a_{r-1}, a_t) & a_t - x \end{vmatrix} \div (a_t - a_r)$$

... (18)

Section 3. Summary:

To sum up, the following table show the scheme of the computation:

col. No. j / row No. i	0	1	2	3	4 ...	n+1
1	a_1	$f(a_1)$				
2	a_2	$f(a_2)$	$f(x\|a_1,a_2)$			
3	a_3	$f(a_3)$	$f(x\|a_1,a_3)$	$f(x\|a_1,a_2,a_3)$	... ...	
4	a_4	$f(a_4)$	$f(x\|a_1,a_4)$	$f(x\|a_1,a_2,a_4)$	... .	
5	a_5	$f(a_5)$	$f(x\|a_1,a_5)$	$f(x\|a_1,a_2,a_5)$	... .	
⋮	⋮	⋮	⋮	⋮	⋮ ⋮	
n+1	a_{n+1}	$f(a_{n+1})$	$f(x\|a_1,a_{n+1})$	$f(x\|a_1,a_2,a_{n+1})$...	...	$f(x\|a_1,..,a_{n+1})$

For a given row i & colum j the element E_{ij} defines symbolically the following Lagrangian interpolation formula:

$$E_{ij} = f(x|a_1, a_2, a_3, \ldots, a_{j-1}, a_i)$$ ... (19)

The element E_{ij} is obtained from the elements lying from the preceding row & column through the following formula

$$E_{ij} = \begin{vmatrix} E_{j-1,j-1} & a_{j-1}-x \\ & \\ E_{i,j-1} & a_i - x \end{vmatrix} \div (a_i - a_{j-1})$$

... (20)

Section 4: Computation procedure:

The Aitken's algorithim for the Lagrangian interpolation formula had been programmed on the I.B.M. 1620 using Fortran Language.

Section 4.1 gives the symbols used in the Fortran program and the way the data must be prepared.

Section 4.2 gives the block diagram.

Section 4.3 gives the program itself in the Fortran Language.

Section 4.4 gives a numerical example.

The program available can handle interpolation by Aitken's formula provided the number of points at which the function is given is less or equal to 100. If we have more points, equal in value to the integer K, then we must change the first dimension statment in the source program to the following statment

```
DIMENSION  A (K), F(K), F1(K), F1X(K)
```

and of course we have to compile the program again to get the object program.

Section 4.1: Symbols used in the program and the preparation of data.

The formula is.

$$f(x \mid a_1, a_2, \ldots, a_r, a_j) = \begin{vmatrix} f(x \mid a_1, \ldots, a_{r-1}, a_r) & a_r - x \\ f(x \mid a_1, \ldots, a_{r-1}, a_j) & a_j - x \end{vmatrix} \div (a_j - a_r)$$

Symbols in Theory	Corresponding Symbols in Fortran
n+1	M
j	J
a_j	A(J)
$f(a_j)$	F(J)
x	X
$f(x \quad a_1, \ldots, a_r, a_j)$	FIX(J)
r	JO

The preparation of data:

The first card must contain the number M of values at which the function is given and the code of the process as following:

M : colomn 1 → 3 xxx integer form

Code : colomn 4 → 10 xx.xxxx floating form

After the first card we have M cards each corresponding to one given value of the function.

The data in these M cards are as following:

J : colomn 1 → 3 xxx integer form

A(J) : colomn 4 → 17 ± x.xxxxxxxE±xx E form

F(J) : colomn 18 → 31 ± x.xxxxxxxE±xx E form

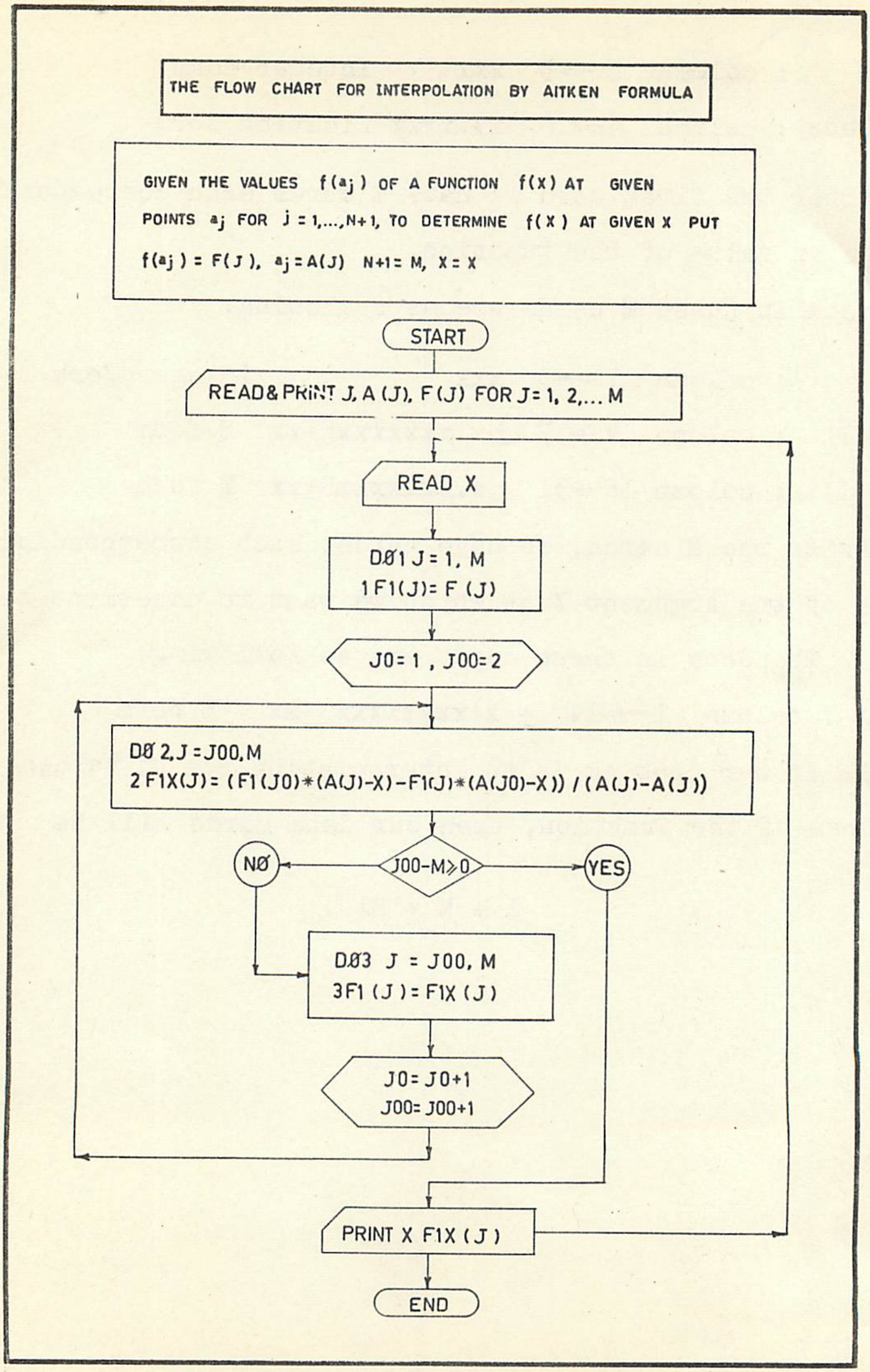
After the M cards, we have cards, each corresponding to the value of the argument X at which we want to determine our function. The data in these cards are as following:

X : colomn 1 → 14 ± x.xxxxxxxE ±xx E form

So if our problem is to interpolation for M1 values, given M values of the function, then our data cards will be cards

1 + M + M1 .

Block diagram
of Aitken



```
C THE FOLLOWING IS A FORTRAN PROGRAM FOR AITKEN INTERPOLATION FORMULA
C THE NUMBER OF POINTS USED IN INTERPOLATION IS LESS THAN 100
      DIMENSION A(100),F(100),F1(100),F1X(100)
      READ 100,M,CODE
  100 FORMAT(I3,F7.4)
      DO 101J=1,M
  101 READ 102,J,A(J),F(J)
  102 FORMAT(I3,2E14.7)
      PRINT 103,
  103 FORMAT(59H                               THE  FOLLOWING  ARE  THE  D
     XATA)
      PRINT 104,
  104 FORMAT(52H                    J          A(J)                F(J))
      DO 105J=1,M
  105 PRINT 106,J,A(J),F(J)
  106 FORMAT(20X,I3,3X,E14.7,3X,E14.7)
  107 READ 108,X
  108 FORMAT(E14.7)
      DO 109J=1,M
  109 F1(J)=F(J)
      J0=1
      J00=2
  110 DO 111J=J00,M
  111 F1X(J)=(F1(J0)*(A(J)-X)-F1(J)*(A(J0)-X))/(A(J)-A(J0))
      IF(SENSE SWITCH 1)116,216
  116 DO 118J=J00,M
  118 PRINT 117,J,J0,J00,F1X(J)
  117 FORMAT(3I3,E14.7)
  216 IF(J00-M)112,114,114
  112 DO 113J=J00,M
  113 F1(J)=F1X(J)
      J0=J0+1
      J00=J00+1
      GO TO 110
  114 PRINT 115,X,F1X(M)
  115 FORMAT(20X,2HX=E14.7,3X,5HF(X)=E14.7)
      GO TO 107
      END
```

THE FOLLOWING ARE THE DATA

J	A(J)	F(J)
1	1.0000000E+00	0.0000000E-99
2	9.0380000E-01	2.2030000E-01
3	8.0920000E-01	4.2130000E-01
4	7.2870000E-01	5.7930000E-01
5	6.6790000E-01	6.7560000E-01
6	5.8470000E-01	7.6730000E-01
7	4.8290000E-01	8.5650000E-01
8	3.7100000E-01	9.2660000E-01
9	2.4800000E-01	9.7180000E-01
10	7.6500000E-02	9.9450000E-01

X= 5.0000000E-01   F(X)= 8.4171142E-01

Part III.

Hermit's Interpolation Formula.

Section 1. - Introduction:

Given the following 2n+2 numerical values

$$f(a_j)\ ,\ f'(a_j) \qquad j = 1,2,\dots,n+1. \qquad \dots (1)$$

of a dependent variable f and its derivative f' corresponding to the n+1 values of the independent variables

$$a_j \qquad j=1,2,\dots,n+1. \qquad \dots (2)$$

then it is possible to construct an interpolation polynomial f(x) of degree 2n+1, such that

$$\begin{aligned} f(x) &= f(a_j) \\ f'(x) &= f'(a_j) \end{aligned} \qquad \dots (3)$$

at $x = a_j \quad j=1,2,\dots,n+1.$

This polynomial is called "Hermite Interpolation polynomial". In the following section we shall derive this polynomial in a formula convenient for numerical application.

Section 2. The Hermite Interpolation Formula.

From the above definition we have

$$f(x) = \sum_{i=o}^{2n+1} A_i\, x^i$$

$$\& \quad f(a_j) = \sum_{i=o}^{2n+1} A_i\, (a_j)^i \qquad j=1,\dots,n+1 \qquad \dots (4)$$

$$\& \quad f'(a_j) = \sum_{i=1}^{2n+1} iA_i \, (a_j)^{i-1} \qquad j=1,\ldots,n+1. \qquad \ldots (4)$$

The set of equations (4) will be consistent, if we have the following condition

$$\begin{vmatrix} f(x) & 1 & x & x^2 & \cdots & x^{2n+1} \\ f(a_1) & 1 & a_1 & a_1^2 & \cdots & a_1^{2n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f(a_{n+1}) & 1 & a_{n+1} & a_{n+1}^2 & \cdots & a_{n+1}^{2n+1} \\ f'(a_1) & 0 & 1 & 2a_1 & \cdots & (2n+1)a_1^{2n} \\ f'(a_2) & 0 & 1 & 2a_2 & \cdots & (2n+1)a_2^{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f'(a_{n+1}) & 0 & 1 & 2a_{n+1} & \cdots & (2n+1)a_{n+1}^{2n} \end{vmatrix} = 0 \qquad \ldots (5)$$

Expanding the above determinant columnwise we have

$$\Delta_o f(x) = \sum_{j=1}^{2n+1} \Delta_j(x) \; f(a_j) \; + \; \sum_{j=1}^{2n+1} \bar{\Delta}_j(x) \; f'(a_j)$$

where $\ldots (6)$

$$
\Delta_o = \begin{vmatrix}
- & a_1 & a_1^2 & \cdots & a_1^{2n+1} \\
1 & a_2 & a_2^2 & \cdots & a_2^{2n+1} \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
1 & a_{n+1} & a_{n+1}^2 & \cdots & a_{n+1}^{2n+1} \\
0 & 1 & 2a_1 & \cdots & (2n+1)a_1^{2n} \\
0 & 1 & 2a_2 & \cdots & (2n+1)a_2^{2n} \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
0 & 1 & 2a_{n+1} & \cdots & (2n+1)a_{n+1}^{2n}
\end{vmatrix} \quad \ldots (7),
$$

$$
\Delta_j(x)=(-1)^{j+1} \begin{vmatrix}
1 & x & x^2 & \cdots & x^{2n+1} \\
1 & a_1 & a_1^2 & \cdots & a_1^{2n+1} \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
1 & a_{j-1} & a_{j-1}^2 & \cdots & a_{j-1}^{2n+1} \\
1 & a_{j+1} & a_{j+1} & \cdots & a_{j+1}^{2n+1} \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
1 & a_{n+1} & a_{n+1}^2 & \cdots & a_{n+1}^{2n+1} \\
0 & 1 & 2a_1 & \cdots & (2n+1)a_1^{2n} \\
0 & 1 & 2a_2 & \cdots & (2n+1)2a_2^{2n} \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
0 & 1 & 2a_{n+1} & \cdots & (2n+1)a_{n+1}^{2n}
\end{vmatrix} \quad \ldots (8),
$$

And

$$\bar{\Delta}_j(x)=(-1)^{j+1}\cdot\begin{vmatrix} 1 & x & x^2 & \cdots & x^{2n+1} \\ 1 & a_1 & a_1^2 & \cdots & a_1^{2n+1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & a_{n+1} & a_{n+1}^2 & \cdots & a_{n+1}^{2n+1} \\ 0 & 1 & 2a_1 & \cdots & (2n+1)a_1^{2n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 1 & 2a_{j-1} & \cdots & (2n+1)a_{j-1}^{2n} \\ 0 & 1 & 2a_{j+1} & \cdots & (2n+1)a_{j+1}^{2n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 1 & 2a_{n+1} & \cdots & (2n+1)a_{n+1}^{2n} \end{vmatrix} \quad \ldots (9)$$

<u>Evaluation of $\bar{\Delta}_j(x)$</u>

The determinant $\bar{\Delta}_j(x)$ is a polynomial in x of degree 2n+1.

From equation (9) it is seen that

$$\bar{\Delta}_j(x) = 0 \quad \ldots (10)$$

at $x = a_j$ $j=1,2,\ldots,n+1$.

Equation (9) can , in fact, take the following form

$$\Delta_j(x) = (-1)^{j+1} \cdot \begin{vmatrix} a_1 - x & a_1^2 - x^2 & \cdots & a_1^{2n+1} - x^{2n+1} \\ a_2 - x & a_2^2 - x^2 & \cdots & a_2^{2n+1} - x^{2n+1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n+1} - x & a_{n+1}^2 - x^2 & \cdots & a_{n+1}^{2n+1} - x^{2n+1} \\ 1 & 2a_1 & \cdots & (2n+1)a_1^{2n} \\ 1 & 2a_2 & \cdots & (2n+1)a_2^{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 2a_{j-1} & \cdots & (2n+1)a_{j-1}^{2n} \\ 1 & 2a_{j+1} & \cdots & (2n+1)a_{j+1}^{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 2a_{n+1} & \cdots & (2n+1)a_{n+1}^{2n} \end{vmatrix} \quad \ldots (11)$$

$$= (x-a_1)(x-a_2)\ldots.(x-a_{n+1}) \; D_j'(x) \; (-1)^{j+1} \quad \ldots(12)$$

where $D_j'(x)$ is a polynomial of degree

$(2n+1) - (n+1) = n$ which takes the following form:

$$D_j'(x)=\begin{vmatrix} -1 & -(a_1^2-x^2)/(a_1-x) & \cdots & -(a_1^{2n+1}-x^{2n+1})/(a_1-x) \\ -1 & -(a_2^2-x^2)/(a_2-x) & \cdots & -(a_2^{2n+1}-x^{2n+1})/(a_2-x) \\ \vdots & \vdots & \cdots & \vdots \\ -1 & -(a_{n+1}^2-x^2)/(a_{n+1}-x) & \cdots & -(a_{n+1}^{2n+1}-x^{2n+1})/(a_{n+1}-x) \\ 1 & 2a_1 & \cdots & -(2n+1)\ a_1^{2n} \\ 1 & 2a_2 & \cdots & (2n+1)\ a_2^{2n} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & 2a_{j-1} & \cdots & (2n+1)\ a_{j-1}^{2n} \\ 1 & 2a_{j+1} & \cdots & (2n+1)\ a_{j+1}^{2n} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & 2a_{n+1} & \cdots & (2n+1)\ a_{n+1}^{2n} \end{vmatrix} \qquad \ldots(13)$$

Since the $\lim\limits_{x\to a} \dfrac{a^i - x^i}{a-x} = i\ a^{i-1}$ ...(14)

then it is obvious that

$$D_j'(x) = 0 \qquad \ldots(15)$$

$$\text{at } x = a_1,\ a_2,\ \ldots,\ a_{j-1},\ a_{j+1},\ \ldots,\ a_{n+1}. \qquad \ldots(16)$$

i.e. $D_j'(x) = C\ (x-a_1)(x-a_2)\ \ldots(x-a_{j-1})\ (x-a_{j+1})\ \ldots\ (x-a_{n+1})$.(17)

where

$$C = \text{constant}. \qquad \ldots(18)$$

From (12) & (17) we have

$$\bar{\Delta}_j(x)=C\,(x-a_1)^2\,(x-a_2)^2\,\ldots(x-a_{j-1})^2(x-a_{j+1})^2\ldots(x-a_{n+1})^2.$$

$$(x-a_j)\,.\,(-1)^{j+1}. \qquad \ldots (19)$$

To eliminate the constant C , we consider the determinant Δ_o. Comparing the two determinants defining Δ_o & $\Delta_j(x)$ we can write by analogy

$$\Delta_o = (-1)^{j+1}\,(a_j-a_1)(a_j-a_2)\,\ldots\,(a_j-a_{j-1})(a_j-a_{j+1})\,\ldots$$

$$(a_j-a_{n+1})\,.\,D'_{oj}\,. \qquad \ldots (20)$$

where

$$D'_{oj} = C\,(a_j-a_1)(a_j-a_2)\,\ldots\,(a_j-a_{j-1})(a_j-a_{j+1})\ldots(a_j-a_{n+1}) \quad \ldots (21)$$

That means

$$\Delta_o = C\,(-1)^{j+1}\,(a_j-a_1)^2(a_j-a_2)^2\,\ldots\,(a_j-a_{j-1})^2(a_j-a_{j+1})^2$$

$$\ldots\,(a_j-a_{n+1})^2. \qquad \ldots (22)$$

From (19) & (22) we find that

$$\frac{\Delta_j(x)}{\Delta_o} = (x-a_j)\,.\,\left[L_j(x)\right]^2 \qquad \ldots (23)$$

where

$$L_j(x) = \frac{(x-a_1)(x-a_2)\ldots(x-a_{j-1})(x-a_{j+1})\ldots(x-a_{n+1})}{(a_j-a_1)(a_j-a_2)\ldots(a_j-a_{j-1})(a_j-a_{j+1})\ldots(a_j-a_{n+1})} \qquad \ldots (24)$$

$$L_j(x) = \frac{P_n(x)}{(x-a_j)\left[P'_n(x)\right]_{x=a_j}} \qquad \ldots (25)$$

$$P_n(x) = \prod_{i=1}^{i=n+1} (x-a_i) \qquad \ldots (26)$$

<u>The evaluation of $\Delta_j(x)$</u>

Since Δ_j (x) is a polynomial of degree 2n+1 we can put it in the form

$$\Delta_j(x) = \left[L_j(x)\right]^2 \ (A(x-a_j) + B \ .(-1)^{j+1} \qquad \ldots (27)$$

To determine B , put $x=a_j$ in the above equation.

$$\therefore \quad \Delta_j(a_j) = \left[L_j(a_j)\right]^2 \ B \ .(-1)^{j+1} \qquad \ldots (28)$$

Putting $x = a_j$ in (8) & comparing with (7) we have

$$\Delta_j(a_j) = (-1)^{j+1} \Delta_0 . \qquad \ldots (29)$$

Again from equation (24) defining L_j (x) we have

$$L_j(a_j) = 1 \qquad \ldots (30)$$

From (28) & (29) & (30) we have

$$B = \Delta_0 \qquad \ldots (31)$$

To determine A, we defferentiate equation 27 with respect to x, to get the following

$$\frac{d}{dx} \Delta_j(x) = \left[L_j(x)\right]^2 \ A \ . \ (-1)^{j+1} + 2 \, L_j(x) \frac{d \, L_j(x)}{dx} \left[A \ (x-a_j) + \Delta_0\right] .(-1)^{j+1} \qquad \ldots (32)$$

Putting $x=a_j$ we have

$$\left[\frac{d}{dx}\Delta_j(x)\right]_{x=a_j} = (-1)^{j+1}\left[A + 2\Delta_0\frac{dL_j(x)}{dx}\right]_{x=a_j} \quad \ldots(33)$$

From (8) it is obvious that

$$\frac{d}{dx}\Delta_j(x)_{x=a_j} = 0 \quad \ldots(34)$$

$$\therefore A = -2\left[\frac{dL_j(x)}{dx}\right]_{x=a_j}\cdot\Delta_0 \quad \ldots(35)$$

From the definition of $L_j(x)$ (equation 24) we have

$$\frac{1}{L_j(x)}\frac{d\,L_j(x)}{dx} = \frac{1}{x-a_1} + \frac{1}{x-a_2} + \ldots + \frac{1}{x-a_{j-1}} + \frac{1}{x-a_{j+1}} + \ldots \frac{1}{x-a_{n+1}}. \quad \ldots(36)$$

$$\therefore \frac{d\,L_j(x)}{dx} = \frac{1}{a_j-a_1} + \frac{1}{a_j-a_2} + \ldots + \frac{1}{a_j-a_{j-1}} + \frac{1}{a_j-a_{j+1}} + \ldots + \frac{1}{a_j-a_{n+1}}. \quad \ldots(37)$$

It can be proved that the above expression for

$$\left[\frac{d\ L_j(x)}{dx}\right]_{x=a_j} \text{ is equal to } \frac{1}{2}\ \frac{P_n''(a_j)}{P_n'(a_j)} \quad \ldots(38)$$

$$\text{where } P_n'(a_j) = \left[\frac{d\ P_n(x)}{dx}\right]_{x=a_j} \quad \ldots(39)$$

$$\&\ P_n''(a_j) = \left[\frac{d^2\ P_n(x)}{dx^2}\right]_{x=a_j} \quad \ldots(40)$$

From equations (3) , (35) & (27) we get

$$\frac{\Delta_j\ (x)}{\Delta_o} = \left[L_j(x)\right]^2 . \left[1 - \frac{P_n''(a_j)}{P_n'(a_j)} \cdot (x-a_j)\right] \quad \ldots(41)$$

Section 3 . Summary :

To sum up the above analysis, put

$$\frac{\Delta_j(x)}{\Delta_o} = h_j\ (x)$$

$$\frac{\bar{\Delta}_j(x)}{\Delta_o} = \bar{h}_j\ (x)$$

where we have now

$$h_j(x) = \left[1 - \frac{P_n''(a_j)}{P_n'(a_j)} \cdot (x-a_j)\right] \cdot \left[L_j(x)\right]^2$$

& $\bar{h}_j(x) = (x-a_j)\cdot\left[L_j(x)\right]^2$

then

$$F(x) = \sum_{j=1}^{n+1} h_j(x)\, f(a_j) + \sum_{j=1}^{n+1} \bar{h}_j(x)\, f'(a_j)$$

which is the required form of the Hermite Interpolation formula.

Appendix

Derivation of formula for $\dfrac{P_n''(a_j)}{P_n'(a_j)}$

By definition of $P_n(x)$ we have

$$P_n(x) = (x-a_1)(x-a_2)\ldots(x-a_{n+1})$$

Defferantiating logarithmically we have

$$\frac{P_n'(x)}{P_n(x)} = \frac{1}{x-a_1} + \frac{1}{x-a_2} + \ldots + \frac{1}{x-a_{n+1}}$$

Defferantiating again we have

$$\frac{P_n''(x)}{P_n(x)} = \frac{P_n'^2(x)}{P^2(x)} - \left[\frac{1}{(x-a_1)^2} + \frac{1}{(x-a_2)^2} + \ldots + \frac{1}{(x-a_{n+1})^2}\right]$$

$$= \left[\frac{1}{x-a_1} + \frac{1}{x-a_2} + \ldots + \frac{1}{x-a_{n+1}}\right]^2$$

$$-\left[\left(\frac{1}{x-a_1}\right)^2 + \left(\frac{1}{x-a_2}\right)^2 + \ldots + \left(\frac{1}{x-a_{n+1}}\right)^2\right]$$

$$\frac{1}{2} \quad \frac{P_n''(x)}{P_n(x)} = \frac{1}{x-a_1}\left[\frac{1}{x-a_2} + \frac{1}{x-a_3} + \dots + \frac{1}{x-a_j} + \frac{1}{x-a_{j+1}}\right.$$

$$\left. + \dots + \frac{1}{x-a_{n+1}}\right] + \frac{1}{x-a_2}\left[\frac{1}{x-a_3} + \frac{1}{x-a_{j+1}}\right.$$

$$\left. + \dots + \frac{1}{x-a_{n+1}}\right] + \dots + \dots + \dots +$$

$$+ \frac{1}{x-a_j}\left[\frac{1}{x-a_{j+1}} + \dots + \frac{1}{x-a_{n+1}}\right]$$

$$+ \dots + \dots + \dots + \frac{1}{x-a_n} \; \frac{1}{x-a_{n+1}} .$$

$$= \text{Finate quantities at } x = a_j$$

$$+ \frac{1}{x-a_j}\left[\frac{1}{x-a_1} + \frac{1}{x-a_2} + \dots + \frac{1}{x-a_{j+1}} + \frac{1}{x-a_{j+1}}\right.$$

$$\left. + \dots \; \frac{1}{x-a_{n+1}}\right]$$

$$\frac{1}{2} P_n''(x) = P_n(x). \text{ Quantities which are finite at } x = a_j$$

$$+ \frac{P_n(x)}{x-a_j}\left[\frac{1}{x-a_1} + \frac{1}{x-a_2} + \dots + \frac{1}{x-a_{j-1}} + \frac{1}{x-a_{j+1}}\right.$$

$$\left. + \dots + \frac{1}{x-a_{n+1}}\right]$$

But $$\frac{P_n(x)}{x-a_j} = \left[P_n'(x)\right]_{x=a_j}$$

$$\frac{1}{2} \frac{P_n''(a_j)}{P_n'(a_j)} = \frac{1}{a_j-a_1} + \frac{1}{a_j-a_2} + \ldots + \frac{1}{a_j-a_{j-1}} + \frac{1}{a_j-a_{j+1}} + \ldots + \frac{1}{a_j-a_{n+1}} .$$

Section 4. Computation procedure.

The Hermit's interpolation formula had been programmed on the I.B.M. 1620 using Fortran Language.

Section 4.1 gives the symbols used in the Fortran program and the way the data must be prepared.

Section 4.2 gives the block diagram.

Section 4.3 gives the program itself in the Fortran Language.

Section 4.4 gives a numerical example.

The program available can handle interpolation by Hermit's formula provided the number of points at which the function is given is less or equal to 100.
If we have more points, equal in value to the integer J1, then we must change the first dimension statment in the source program to the following statment

```
DIMENSION  A(J1), F(J1), F1(J1)
```

and of course we have to compile the program again to get the object program.

Section 4.1: Symbols used in the program and the preparation of data.

The formula is

$$F(x) = \sum_{j=1}^{n+1} h_j(x) f(a_j) + \sum_{j=1}^{n+1} \bar{h}_j(x) f'(a_j)$$

where

$$h_j(x) = \left[1 - \frac{P_n''(a_j)}{P_n'(a_j)} \cdot (x-a_j)\right] \cdot \left[L_j(x)\right]^2$$

$$\bar{h}_j(x) = (x-a_j) \left[L_j(x)\right]^2 .$$

$$\frac{1}{2} \frac{P_n''(a_j)}{P_n'(a_j)} = \frac{1}{a_j-a_1} + \frac{1}{a_j-a_2} + \ldots + \frac{1}{a_j-a_{j-1}} + \frac{1}{a_j-a_{j+1}} + \ldots + \frac{1}{a_j-a_{n+1}} .$$

$$L_j(x) = \frac{(x-a_1)(x-a_2)\ldots(x-a_{j-1})(x-a_{j+1})\ldots(x-a_{n+1})}{(a_j-a_1)(a_j-a_2)\ldots(a_j-a_{j-1})(a_j-a_{j+1})\ldots(a_j-a_{n+1})}$$

Symbols in Theory	Corresponding Symbols in Fortran
$n+1$	N1
j	J
a_j	A(J)
$f(a_j)$	F(J)
$f'(a_j)$	F1(J)
x	X
$\left[L_j(x)\right]^2$	ELJX2
$h_j(x)$	HJX
$\bar{h}_j(x)$	H1JX
$F(x)$	FX

<u>The preparation of data:</u>

The first card must contain the number N1 of values at which the function is given and the code of the process as following

N1	: colomn	1 → 3	xxx	integer form
Code	: colomn	4 → 10	xx.xxxx	floating form

After the first card we have N1 cards each corresponding to one given value of the function. The data in these N1 cards are as following:

J	: colomn	1 → 3	xxx	integer form
A(J)	: colomn	4 → 17	±x.xxxxxxxE±xx	E form
F(J)	: colomn	18 → 31	±x.xxxxxxxE±xx	E form
F1(J)	: colomn	32 → 45	±x.xxxxxxxE±xx	E form

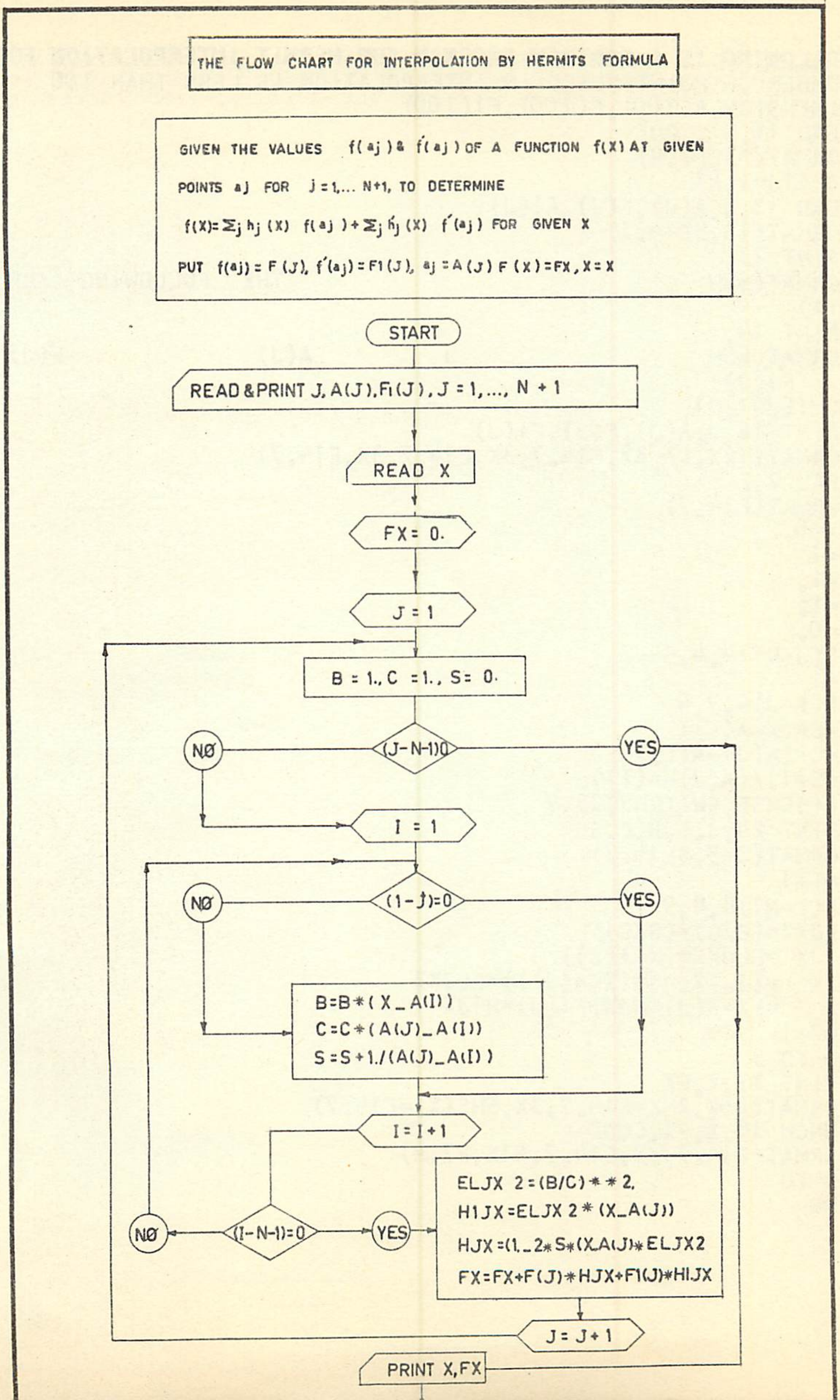
After the N1 cards, we have cards, each corresponding to the value of the argument X at which we want to determine our function.

The data in these cards are as following:

X : colomn 1 → 14 ±x.xxxxxxxE±xx E form

So if our problem is to interpolation for M1 values, given N1 values of the function, then our data cards will be

$$1 + N1 + M1 .$$


THE FLOW CHART FOR INTERPOLATION BY HERMITS FORMULA
GIVEN THE VALUES f(aj) & f'(aj) OF A FUNCTION f(X) AT GIVEN POINTS aj FOR j = 1,... N+1, TO DETERMINE
f(X)= Σj hj (X) f(aj) + Σj h'j (X) f'(aj) FOR GIVEN X
PUT f(aj) = F(J), f'(aj) = F1(J), aj = A(J) F(X) = FX, X = X
START
READ & PRINT J, A(J), F1(J), J = 1,..., N + 1
READ X
FX = 0.
J = 1
B = 1., C = 1., S = 0.
(J-N-1)0
NØ
YES
I = 1
(1-J)=0
NØ
YES
B=B*(X_A(I))
C=C*(A(J)_A(I))
S=S+1./(A(J)_A(I))
I = I+1
(I-N-1)=0
NØ
YES
ELJX 2 = (B/C) * * 2,
H1JX = ELJX 2 * (X_A(J))
HJX = (1._2* S*(X.A(J))* ELJX2
FX = FX+F(J)*HJX+F1(J)*HIJX
J = J+1
PRINT X, FX

```
C THE FOLLOWING IS A FORTRAN PROGRAM FOR HERMIT INTERPOLATION FORMULA
C THE NUMBER OF POINTS USED IN INTERPOLATION IS LESS THAN 100
      DIMENSION A(100),F(100),F1(100)
      READ 10,N1,CODE
   10 FORMAT(I3,F7.4)
      DO 11J=1,N1
   11 READ 12,J,A(J),F(J),F1(J)
   12 FORMAT(I3,3E14.7)
      PRINT 13,
   13 FORMAT(59H                                          THE  FOLLOWING  ARE  THE  D
     XATA)
      PRINT 14,
   14 FORMAT(65H                  J          A(J)                F(J)
     X     F1(J))
      DO 15J=1,N1
   15 PRINT 16,J,A(J),F(J),F1(J)
   16 FORMAT(16X,I3,3X,E14.7,3X,E14.7,3X,E14.7)
    1 READ 2,X
    2 FORMAT(E14.7)
      FX=0.
      J=1
    3 B=1.
      C=1.
      S=0.
      IF(J-N1)4,4,5
    4 I=1
    8 IF(I-J)6,7,6
    6 B=B*(X-A(I))
      C=C*(A(J)-A(I))
      S=S+1./(A(J)-A(I))
      IF(SENSE SWITCH1)23,7
   23 PRINT 24,J,I,B,C,S
   24 FORMAT(2I3,3E14.7)
    7 I=I+1
      IF(I-N1)8,8,9
    9 ELJX2=(B/C)*(B/C)
      H1JX =ELJX2*(X-A(J))
      HJX  =(1.-2.*S*(X-A(J)))*ELJX2
      FX   =FX+F(J)*HJX+F1(J)*H1JX
      J=J+1
      GO TO 3
    5 PRINT 17,X,FX
   17 FORMAT(20X,2HX=E14.7,3X,5HF(X)=E14.7)
      PUNCH 18,X,FX,CODE
   18 FORMAT(E14.7,3X,E14.7,41X,F7.4)
      GO TO 1
      END
```

THE FOLLOWING ARE THE DATA

J	A(J)	F(J)	F1(J)
1	1.0000000E+00	0.0000000E-99	-2.3891756E+00
2	9.0380000E-01	2.2030000E-01	-2.2090654E+00
3	8.0920000E-01	4.2130000E-01	-2.0302224E+00
4	7.2870000E-01	5.7930000E-01	-1.8217870E+00
5	6.6790000E-01	6.7560000E-01	-1.2616252E+00
6	5.8470000E-01	7.6730000E-01	-9.9629300E-01
7	4.8290000E-01	8.5650000E-01	-7.6122790E-01
8	3.7100000E-01	9.2660000E-01	-4.9904420E-01
9	2.4800000E-01	9.7180000E-01	-2.5338610E-01
10	7.6500000E-02	9.9450000E-01	-4.5962200E-02

X= 5.0000000E-01 F(X)= 8.4194621E-01