Lagrangian Interpolation for the HP41C

by

Namir Shammas

This article presents an HP-41C program for Langrangian Interpolation of N points, where N > 1

Program XLGINT

Usage

XEQ XLGINT (same as pressing A)

A Program prompts you to enter N, N points, and the interpolated value of x

B Program prompts you to enter the interpolated value of x

Program calculates and displays the value of the interpolated Y.

Example

Consider the data in the following table:

X	Y
1	1
5	25
10	100

Using the above data, calculate Y for X = 4. The Steps involved are:

Step	Task	Command/Input	Output
1	Start the program.	XEQ "XLGINT"	N?
2	Enter the number of points.	3 [R/S]	Y1↗X1?
3	Enter the first data point.	1 [ENTER↑] 1 [R/S]	Y2↗X2?

Step	Task	Command/Input	Output
4	Enter the second data point.	25 [ENTER↑] 5 [R/S]	Y3↗X3?
5	Enter the third data point.	100 [ENTER↑] 10[R/S]	READY
6	Start the interpolation.	[B]	X?
7	Enter the interpolated value of X	4 [R/S]	Y=16.00000

Algorithm

```
INPUT N, array X(1..N), Y(1..N), and Xint
Yint = 0
FOR I = 1 TO N
  Product = Y(I)
  FOR J = 1 to N
    IF I <> J THEN
      Product = Product * (Xint – X(J)) / (X(I) – X(J))
    ENDIF
  NEXT J
  Yint = Yint + Product
NEXT I
Show Yint
```

Memory Map

R00 = N
R01 = Xint
R02 = Yint
R03 = Product
R05 = J for X(J)
R06 = I for X(I)
R07 = I for Y(I)
R10 = X(1)
R11 = X(2)
...
R10+N-1 = X(N)
R10+N = Y(1)
R11+N = Y(2)
...
R10+2N-1 = Y(N)

Source Code

The source code for the HP-41C program appears below. Please note the following:

Text appearing in a pair of double quotes represents characters in the Alpha register.

The |- characters represent the single *append character* for the Alpha register.

The blank lines are intentionally inserted to separate logical blocks of commands:

```
LBL XLGINT
LBL A
"N?"
PROMPT
INT
STO 00
2
X>Y?                    # is N < 2? then re-prompt N
GTO A
X<>Y
1E3
/
1
+
STO 05                  # Initializes indices for storage
10
STO 06                  # I for X(I)

RCL 00
+
STO 07                  # I for Y(I)

LBL 00                  # start input loop
FIX 0
CF 29
"Y"
ARCL 05
"|-^X"
ARCL 05
"|-?"
FIX 5
SF 29
PROMPT
STO IND 06
X<>Y
```

```
STO IND 07
1
ST+ 06
ST+ 07
ISG 05
GTO 00                  # end input loop

LBL B
"X?"
PROMPT
STO 01

XEQ 09
STO 06                  # set I for X(I)

XEQ 10
STO 07                  # set I for Y(I)

0
STO 02                  # y = 0

LBL 01                  # outer loop starts

RCL IND 07
STO 03

XEQ 09
STO 05                  # set J for X(J)

LBL 02                  # inner loop starts

RCL 06
RCL 05
X=Y?                    # if I = J the skip iteration
GTO 03

RCL 01
RCL IND 05
-
RCL IND 06
RCL IND 05
-
/
ST* 03                  # update product
```

```
LBL 03
ISG 05
GTO 02                  # inner loop ends

RCL 03
ST+ 02

1
ST+ 07                  # update index of Y() by simply adding 1
ISG 06
GTO 01                  # outer loop ends

"Y = "
ARCL 02
PROMPT
GTO B

LBL 09                  # subroutine to calculate index range for
10                      # accessing array X()
STO Y
RCL 00
+
1
-
1E3
/
+
RTN

LBL 10                  # subroutine to calculate starting index # for accessing array
                        Y()
10
RCL 00
+
RTN
```

Program XLGIN2

Usage

XEQ XLGINT2	(same as pressing A)
A	Program prompts you to enter N, N points, and the interpolated value of x
B	Program prompts you to enter the interpolated value of x

Program calculates and displays the value of the interpolated Y.

Example

Consider the data in the following table:

X	Y
1	1
5	25
10	100

Using the above data, calculate Y for X = 4. The Steps involved are:

Step	Task	Command/Input	Output
1	Start the program.	XEQ "XLGIN2"	N?
2	Enter the number of points.	3 [R/S]	Y1↗X1?
3	Enter the first data point.	1 [ENTER↑] 1 [R/S]	Y2↗X2?
4	Enter the second data point.	25 [ENTER↑] 5 [R/S]	Y3↗X3?
5	Enter the third data point.	100 [ENTER↑] 10[R/S]	READY
6	Start the interpolation.	[B]	X?
7	Enter the interpolated value of X	4 [R/S]	Y=16.00000

Algorithm

```
INPUT N, array X(1..N), Y(1..N), and Xint

' Initialize product difference virtual matrix
For I = 1 To N
  DiffProducts(I) = 1
Next I

' Calculate virtual matrix of differences for x(i) - x(j)
For I = 1 To N - 1
  For J = I + 1 To N
    Diff = Xarr(I) - Xarr(J)
    DiffProducts(I) = DiffProducts(I) * Diff
    DiffProducts(J) = DiffProducts(J) * (-Diff)
  Next J
Next I

' calculate Product of (x - xarr(i)) and
Prod_X_minus_Xarr = 1
For I = 1 To N
  Prod_X_minus_Xarr = Prod_X_minus_Xarr * (X - Xarr(I))
Next I

' now perform the interpolation
Yint = 0
```

```
For I = 1 To N
  Term = Yarr(I) / (X - Xarr(I)) / DiffProducts(I)
  Yint = Yint + Term
Next I
Yint = Yint * Prod_X_minus_Xarr
Show Yint
```

Memory Map

R00 = N

R01 = Xint

R02 = Yint

R03 = Diff

R05 = J for X(J) or I for DiffProduct(I)

R06 = I for X(I)

R07 = I for Y(I) or DiffProduct(I)

R08 = I

R09 = J, Prod_X_minus_Xarr

R10 = X(1)

R11 = X(2)

...

R10+N-1 = X(N)

R10+N = Y(1)

R11+N = Y(2)

...

R10+2N-1 = Y(N)

R10+2N = Diff(1)

R11+2N = Diff(2)

...

R11+3N-1 = Diff(N)

Source Code

The source code for the HP-41C program appears below. Please note the following:

Text appearing in a pair of double quotes represents characters in the Alpha register.

The |- characters represent the single *append character* for the Alpha register.

The blank lines are intentionally inserted to separate logical blocks of commands:

```
LBL XLGIN2
LBL A
"N?"
PROMPT
INT
```

```
STO 00
2
X>Y?            # is N < 2? then re-prompt N
GTO A

1               # Initializes indices for storage
STO 08          # I = 1
XEQ 09
STO 06          # I for X(I)
1
XEQ 10
STO 07          # I for Y(I)

LBL 00          # start input loop
FIX 0
CF 29
"Y"
ARCL 08
"|-^X"
ARCL 08
"|-?"
FIX 5
SF 29
PROMPT
STO IND 06
X<>Y
STO IND 07
1
ST+ 06
ST+ 07
ST+ 08          # I = I + 1
RCL 00
RCL 08
X<=Y?
GTO 00          # end input loop

1
STO 08          # I = 1
XEQ 08
STO 07          # Set index for DiffProduct()
LBL 01          # Initialize DiffProduct()
1
STO IND 07
ST+ 07
```

```
ST+ 08
RCL 00
RCL 08
X<=Y?
GTO 01              # end input loop

1
STO 08              # I = 1
XEQ 09
STO 06              # Set first index for X(I)
LBL 02              # Start outer loop
RCL 08
1
+
STO 09              # J = I + 1
XEQ 09
STO 05              # Set first index for X(J)
LBL 03              # Start inner loop
RCL IND 06
RCL IND 05
-
STO 03              # Diff = X(I) - X(J)
RCL 08
XEQ 08              # Get index for DiffProduct(I)
RCL 03
ST* IND Y           # DiffProducts(I) = DiffProducts(I) * Diff
RCL 09
XEQ 08              # Get index for DiffProduct(J)
X<>Y
CHS
ST* IND Y           # DiffProducts(J) = DiffProducts(J) * (-Diff)
1
ST+ 05
ST+ 09              # J = J + 1
RCL 00
RCL 09
X<=Y?
GTO 03              # End of inner loop
RCL 00
1
ST+ 06
ST+ 08
-
RCL 08
```

```
X<=Y?           # I <= N-1?
GTO 02          # End of outer loop
BEEP
"READY"
PROMPT

LBL B
"X?"
PROMPT
STO 01

1
STO 08          # I = 1
STO 09          # Prod_X_minus_Xarr = 1
XEQ 09
STO 06          # Set index range for X(I)
LBL 04          # Start loop
RCL 01
RCL IND 06
-               # calculate Xint - X(I)
ST* 09          # Prod_X_minus_Xarr = Prod_X_minus_Xarr * (X - Xarr(I))
1
ST+ 08
ST+ 06
RCL 00
RCL 08
X<=Y?
GTO 04          # end of loop

0
STO 02          # Yint = 0
1
STO 08          # I = 1
XEQ 09
STO 06          # set index for X(I)
1
XEQ 10
STO 07          # Set index for Y(I)
1
XEQ 08
STO 05          # Set index for DiffProduct
LBL 05          # Start of loop
RCL IND 07
RCL 01
```

```
RCL IND 06
-
/
RCL IND 05
/                   # Term = Yarr(I) / (X - Xarr(I)) / DiffProducts(I)
ST+ 02              # Yint = Yint + Term
1
ST+ 05              # increment indices
ST+ 06
ST+ 07
ST+ 08
RCL 00
RCL 08
X<=Y?
GTO 05              # End of loop
RCL 09
ST* 02              # Yint = Yint * Prod_X_minus_Xarr
"Y = "
ARCL 02
PROMPT
GTO B

LBL 09              # subroutine to calculate index for
9                   # accessing array X()
+
RTN

LBL 10              # subroutine to calculate index for accessing array Y()
9
+
RCL 00
+
RTN

LBL 08              # subroutine to calculate index for accessing array DiffProd()
9
+
RCL 00
ST+ X
+
RTN
```