MM – Finding Extrema of Functions

Abstract

MM is a program written in 1980 for the HP-41C to find extrema (maxima and/or minima) of an arbitrary user-supplied function y=f(x) by calling program RF (Root Finder) internally as part of the computation. Two worked examples are included.

Keywords: *extrema, maxima, minima, RF, Root Finder, programmable calculator, RPN, HP-41C, HP-41CV, HP-41CX, HP42S*

1. Introduction

MM is a very short *(28 steps) RPN* program that I wrote in 1980 for the *HP-41C* programmable calculator (will also run *as-is* in the *HP-41CV/CX* and the *HP42S)*, which will try to find extrema (maxima and/or minima) of an user-supplied function *y = f(x)* by calling the ***RFP*** program *(Root Finder Programmable*, part of ***RF*** *Root Finder)* to find a root of the function's derivative, which will correspond to the location of the extrema.

The procedure is as follows: given a function ***y = f(x)*** and an initial guess for the location of the maximum or minimum, the program calls ***RFP*** to find a root of the derivative, which is computed by a separate (included herein) program ***DY*** *(Derivative of Y)* which calls the user-specified function and returns the value of ***y'(x)***.

The program does not recognize *inflection ("saddle")* points, it will report them as *maxima* or *minima*. Also, the accuracy depends on the **FIX n** or **SCI n** display setting and usually it won't be higher than about 6-7 correct places due to limitations in the accuracy achievable while computing the derivative (cancellations).

MM was written with the explicit intent of demonstrating how ***RF*** could be used as a subroutine by other programs, which would become much shorter and easier to write. ***RF*** was submitted (and rejected) for inclusion in the ***PPC*** *ROM*, so ***MM*** would have been able to make a direct *ROM* call, saving worthy *RAM* memory.

2. Program Listing

```
01 ♦LBL "MM"
02 "NAME?"
03 AON
04 STOP
05 AOFF
06 ASTO 10
07 "X0?"
08 PROMPT
09 "DY"
10 ASTO 00
11 XEQ "RFP" ►
12 FS? 00
13 RTN
14 R↑
15 STO 00
16 RDN
17 XEQ IND 10 ►
18 "MIN"
19 RCL 00
20 X<Y?
21 "MAX"
22 "⊦: "
23 ARCL 02
24 "⊦, "
25 ARCL Y
26 AVIEW
27 X<>Y
28 END
```

- 28 steps
- requires at least SIZE 014
- uses flag 00 and Alpha register
- y(x) may use R_{04}-R_{09} and R_{14}-R_{nn}
- to key in "⊦" use Append

```
01 ♦LBL "DY"
02 STO 11
03 8E-4
04 STO 12
05 ST+ 12
06 ST- 11
07 +
08 XEQ IND 10 ►
09 STO 13
10 RCL 11
11 XEQ IND 10 ►
12 ST- 13
13 RCL 12
14 ST/ 13
15 RCL 13
16 END
```

- 16 steps
- requires at least SIZE 014
- the function's name is in R_{10}

For completeness' sake, this is the listing of program ***RF*** / ***RFP*** *(see **References** for the paper documenting it):*

```
01 ♦LBL "RF"
02 "NAME?"
03 AON
04 PROMPT
05 AOFF
06 ASTO 00
07 "X0?"
08 PROMPT
09 ♦LBL "RFP"
10 CF 00
11 STO 02
12 50
13 STO 03
14 ♦LBL 00
15 RCL 02
16 1
17 D-R
18 D-R
19 +
20 XEQ IND 00
21 STO 01
22 RCL 02
23 XEQ IND 00
24 X=0?
25 GTO 01 ►
26 ST- 01
27 RCL 01
28 X=0?
29 SIGN
30 /
31 D-R
32 D-R
33 ST- 02
34 RND
35 X=0?
36 GTO 01 ►
37 DSE 03
38 GTO 00 ►
39 SF 00
40 ♦LBL 01
41 RCL 02
42 END
```

3. Usage Instructions

Step 1: Write a program to define *f(x)*. It must be a separate program under its own *global label* (6 char. max.), must assume that the argument *x* is in stack register *X* upon being called, and must compute and leave the value of *f(x)* in stack register *X*. It may use registers R_{04}-R_{09} and R_{14} onwards, and must not use *flag 00*.

The *accuracy* depends on the display setting, **FIX n** / **SCI n**. The greater **n**, the better the accuracy and the longer the time required to achieve it, though usually the computed extremum will be accurate to just 6-7 correct places.

Step 2: Set the display setting (**FIX/SCI 2-4** recommended) and run the program:

FIX n or SCI n	XEQ "MM"	***NAME?***
(enter name of the function)	R/S	***X0?***
(enter guess of location, x_0)	R/S	***MIN: x, y*** *or* ***MAX: x, y***
		where ***x*** is the location of the *MIN/MAX*, and ***y = f(x)***

Notes: - once computed and displayed, ***x*** is in R_{02} and ***y*** is in the display (stack register *X*).
- if the function doesn't have extrema or the procedure does not converge to one, it will automatically stop after 50 iterations and *flag 00* will be *set*. To try another guess go to *Step 2* above. For another function, go to *Step 1*.

4. Examples

The following examples can be useful to check that the program is correctly entered and to understand its usage.

4.1 Example 1

Find the minimum of: $y = x^2 - 4x + 8$

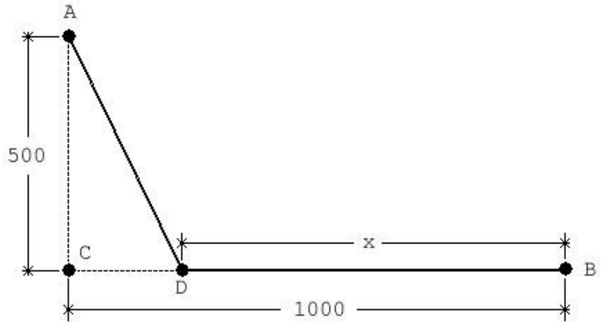
In **PRGM** Mode, enter the following *9-step* program to define *f(x)*:

```
01 ♦LBL "EX1"   02 X↑2   03 LASTX   04 4   05 *   06 -   07 8   08 +   09 END
```

In **RUN** Mode,	FIX 2	XEQ "MM"	***NAME?***
	"EX1"	R/S	***X0?***
	0	R/S	***MIN: 2.00, 4.00*** *so there's a minimum at point* ***(2, 4)***.

4.2 Example 2

To get from point ***A*** to point ***B*** *(see figure below)* some travellers must take both a boat, whose speed is 100 km/h, and a plane, whose speed is 300 km/h. Given that the distance from ***A*** to ***C*** is 500 km and the distance from **B** to **C** is 1000 km, to what point **D** in the coast should they travel by boat (and there take the plane to ***B***) in order to minimize the total travel time from ***A*** to ***B*** ?



(for convenience, use distances/speeds divided by 100)

- *boat time:* $\frac{AD}{1} = \sqrt{AC^2 + CD^2} = \sqrt{5^2 + (10-x)^2}$
$= \sqrt{x^2 - 20x + 125}$

- *plane time:* $\frac{DB}{3} = \frac{x}{3}$

- *total time:* $\sqrt{x^2 - 20x + 125} + \frac{x}{3}$

We need to minimize *total time*, so: $f(x) = \sqrt{x^2 - 20x + 125} + \frac{x}{3}$, which is defined like this:

01 **♦LBL "EX2"**	04 LASTX	07 -	10 SQRT	13 /
02 STO 04	05 20	08 125	11 RCL 04	14 +
03 X↑2	06 *	09 +	12 3	15 END

and now, to compute the minimum (using for initial guess the midpoint of ***BC*** = ***5****(00 km):*

In **RUN** Mode,	FIX 3	XEQ "MM"	***NAME?***
	"EX2"	R/S	***X0?***
	5	R/S	***MIN: 8.232, 8.047***

so point ***D*** is at ***823.2 km*** of ***B*** (*176.8 km* from ***C***) and the minimum time will be ***8.047 h*** = *8h 2'49"*

Notes

1. To see the results more accurately once computed, simply set FIX 6, say, which will show the *value* of the extremum ***(y)*** in the display, and then VIEW 02 will show the corresponding *location* of the extremum ***(x)*** without disturbing the stack.

2. As the accuracy of the extremum location calculated by ***RFP*** depends on the display setting, too low a FIX or SCI setting may result in a location not accurate enough, which in its turn may result in mislabeling a *maximum* as a *minimum* or vice versa. In that case, increase the display setting (from FIX 2 to FIX 3 , say) and try again.

3. Also, the program uses a fast, simple approach to identify whether the computed extremum is a maximum or a minimum, which involves evaluating *f(x)* for a value very near the computed location and comparing both values. This may fail if severe cancellation occurs, and a possible remedy is given *in Note 2* above.

4. The correct way to identify the extremum requires considering the value of the 2nd derivative, ***f"(x)***, at the extremum but the 1st-derivative computation program ***DY*** can't be nested so this would require yet another program to compute the 2nd derivative, at least three additional evaluations of *f(x)*, accuracy would worsen, and this being just a demonstration program for uses of ***RFP*** the additional complexity is not warranted.

5. The program ***(RF***, ***RFP)*** which ***MM*** calls was duly submitted for inclusion in the **PPC ROM** but it wasn't accepted.

References

Francis Scheid (1988). *Schaum's Outline of Theory and Problems of Numerical Analysis, 2nd Edition*.

Valentín Albillo (1980). *HP Program VA411 - HP-41C Finding Roots of Equations*.

Copyrights