HP Forum Archive 13

[Return to Index | Top of Index]

Quiz !: Evaluating polynomials [LONG]

*Message #1 Posted by **Valentin Albillo** on 8 July 2003, 1:19 p.m.*

Hi all,

Though noone posted a solution to my latest HP-15C Quiz/Challenge, "Matrix Trilogy", and though it's pretty clear to me by now that most people participating in this forum are hardware/collector-oriented, being the unrepentant software guy that I am, here is yet another HP-15C Quiz/Challenge for you to try, the last one for a number of months to come.

By the way, let me point it out once more that though the challenge's conditions and requirements are meant for an HP-15C, the techniques used in the solution are always equally efficient and thus applicable to other machines, most specially the 42S, 41C/Advantage, and even the 71B/Math, among others. You can attempt the challenge in said machines as well, and the published solution can give you a particularly efficient and novel implementation on your machine as well, which you might easily find useful for your own programs.

The Quiz/Challenge

Let's suppose that we've got some experimental or otherwise dataset and we have have fitted some suitable mathematical function to it, a rational function to be precise:

```
f(x) = P(x) / Q(x)
```

where P(x) and Q(x) are both polynomials of arbitrary degree **n**, i.e:

```
P(x) = a0 + a1 x + a2 x^2 + ... + an x^n
Q(x) = b0 + b1 x + b2 x^2 + ... + bn x^n
```

where the degree, n, is nominally the same for both, but as any of the ai, bi coefficients can be zero, the actual degrees could be distinct.

Now, for the challenge. You must write a subroutine (LBL, ..., RTN) which must use as fixed data the coefficients ai, bi, which will be stored in a (2 x [n+1]) matrix, in the exact way and order shown below:

```
A =  | a0  a1  a2  ...  an  |
     | b0  b1  b2  ...  bn  |
```

This matrix isn't to be considered a parameter passed to the subroutine, but a fixed data repository at a fixed place, matrix **A**. The degree, n, isn't passed either, the subroutine must deduce it from the number of columns in matrix A.

The only parameter which will be passed, in the X register, is a value **x** for which we want the above expression **f(x)** to be evaluated. The routine will evaluate f(x) = P(x)/Q(x) for such an x, and will return the result in the X register, i.e:

```
x, GSB A  ->  f(x)
```

An example will make it clear: If we have:

```
          -2 + 3x - 5x^2 + 4x^3
f(x) =  ----------------------
         5 - 6x + 3x^2 - 7x^3
```

then our routine must use matrix A, which holds the coefficients:

```
A =  |  -2   3  -5   4  |
     |   5  -6   3  -7  |
```

and **x** = 2.3, to return **f(x)** = -0.3472259...

As the subroutine is meant to be used a number of times to evaluate the function, matrix A shouldn't be altered at all between calls. That given, the primary and main requirement is to **optimize for speed**, i.e., f(x) must be evaluated *as fast as possible*. Once time is minimized, the shorter the subroutine, the better.

Subject to that, there exists a general solution for the HP-15C in *18 steps or less* (including both LBL and RTN) which will evaluate f(x) in just *4 seconds* for polinomials of the 4th degree, and in just *15 seconds* for 19th-degree polynomials.

If you succeed, try to generalize the routine in one or both of these ways:

- instead of a single x value, the routine would accept a vector of x values and would return a vector with the corresponding f(x) values
- instead of evaluating two polynomials, the routine would accept the coefficients of m polynomials and would return the evaluated values at x for all of them.

That's all. Give it a thought or two and let's see your ideas ... :-)

Edited: 9 July 2003, 9:55 a.m. after one or more responses were posted

Re: Quiz !: Evaluating polynomials [LONG]

*Message #2 Posted by **Vieira, Luiz C. (Brazil)** on 9 July 2003, 12:34 a.m.,*
in response to message #1 by Valentin Albillo

Hi, Valentin;

I started to write a reply, but I saw it was heading to a direction not related to your post's interesting, primary subject. I was afraid a not-related thread could arise and that would be an upsetting fact.

So I decided to post my reply here, O.K.? If there is anything to be added to that particular reply, it will not interfère that much in your original post/thread.

I did this to avoid disturbance and misdirection, and I hope not creating others...

Best regards.

Luiz (Brazil)

Edited: 9 July 2003, 12:56 a.m.

Re: Quiz !: Evaluating polynomials [LONG]

*Message #3 Posted by **Tizedes Csaba** on 9 July 2003, 1:09 p.m.,*
in response to message #1 by Valentin Albillo

I do not see Vieira~s solution, yet... This is my version:

LBL A STO 9 'store x RCL f DIM A 1 STO f DIM B f MATRIX 1 u STO B LBL 9 RCL mul 9 u STO B GTO 9 RCL f MATRIX A RCL f MATRIX B mul u RCL C RCL C div RTN

'mul' mean 'multiply', 'div' mean 'divide'.

I do not measure running times, I do not know how fast this program. But it is works...

Csaba

Re: Quiz !: Evaluating polynomials [LONG]

*Message #4 Posted by **Patrick** on 9 July 2003, 7:14 p.m.,*
in response to message #3 by Tizedes Csaba

Hello All

It will come as no surprise to you all that I have been trying this quiz myself. I have a long-ish solution that is close to the idea of Csaba's above:

```
001 LBL A
002 RCL DIM A
003 EEX
004 DIM B
005 RUP              // Get x back from T register
006 ENTER
007 MATRIX 1         // Prepare for loop
008 EEX
009 ENTER            // Stack is now [1,1,x,x]
010 LBL 0
011 *                // Get next power of x, starting at 1
012 u STO B
013 GTO 0
014 RCL MATRIX A
015 RCL MATRIX B
016 RESULT E
017 *                // E = [ P(x) Q(x) ]^T
018 u RCL E          // P(x), R0, R1 update
019 RCL E            // Q(x), R0, R1 do not update
020 /                // f(x) = P(x) / Q(x)
021 RTN
```

The general idea is to create a column vector (matrix B) consisting of the powers of x:

$B = [\ 1\ x\ x^2\ \ldots\ x^n\ ]^T$

When you multiply A by this column vector, the result, E, is a column vector consisting of the values of the polynomials P(x) and Q(x).

The difference between my version and Csaba's is mostly in the way we generate the matrix B. There is a problem in coding the loop (his LBL 9, my LBL 0) in that the first element of the matrix requires special handling. Csaba handles this by a duplicate STO B instruction, one inside his loop and the other before it. I handle it by creating a stack with values X=1, Y=1, Z=x, T=x. The first time you do a multiply inside the loop, you get 1. Successive times you get the higher powers of x.

While longer, my routine will operate in two conditions where Csaba's fails:

1. When n=0.
2. When the result matrix is not its default value of C

The first is a reasonable limitation, I think, but the second is questionable. If we accept the first limitation but not the second, then Csaba's solution only increases by 1 step.

There is one point in my solution where something of a trick is played. If you look at my step 007, it seems to be out of place. Right in the middle of some stack manipulations we insert an instruction which is aimed at preparing a loop some steps later. Why not do it later? The answer is that the MATRIX 1 instruction has the useful side effect of enabling stack lift! Without that instruction there, we would have needed an additional ENTER.

Re: Quiz !: Evaluating polynomials [LONG]

*Message #5 Posted by **Patrick** on 9 July 2003, 7:20 p.m.,*
in response to message #4 by Patrick

Another thought occurs to me that you might speed this routine up some more at the cost of more steps by using rapid branching through the I register rather than using LBL 0. I've never used it myself, so I don't know how much of a speedup it would bring. Of course, it also costs you the I register.

Re: Quiz!: Evaluating polynomials

*Message #6 Posted by **Karl Schneider** on 10 July 2003, 4:09 a.m.,*
in response to message #4 by Patrick

Here's a less-elegant and slower program that does not really utilize the matrix-math capabilities, but instead uses a loop-counter (DSE) and subroutine calls (GSB). With a few modifications, it would be more portable to the 11C and 34C, although those machines could accept only 20 coefficients not in a matrix structure.

The "key to acceptable speed" in these calculator routines is the use of Horner's Method to decompose the exponential calculations of polynomials into arithmetic operations. This is implemented in the "looping" code in both Patrick's routine and mine. Horner's Method is explained in the manuals for the 34C and 15C (and presumably the 11C); the 49G has a "HORNER" function.

The "clever trick" in Patrick's routine is the looping code of of lines 010-013, in which the HP-15C is smart enough to stop filling the array and jump out of the loop (by skipping line 013?) when it got to the end. I believe that Valentin's program from "Long Live the HP-15C" exploited this feature, but I'll have to check if it's documented.

So, here's another program:

```
001  LBL A
002  ENTER
003  ENTER
004  RCL DIM A
005  MATRIX 1
```

```
006  STO 1
007  STO 2
008  X
009  CLx               // stack has  0 x x x
010  GSB 0             // calculate numerator
011  X<>2              // store numerator and retrieve # of coefs
012  STO 1
013  CLx
014  2
015  STO 2
016  CLx               // stack has  0 x x x
017  GSB 0             // calculate denominator
018  STO/2
019  RCL 2
020  RTN               // return answer
021  LBL 0             // "Horner's Method" loop
022  RCL A
023  +                 // add coefficient
024  DSE 1             // decrement index of coef; check if = 0
025  GTO 1
026  RTN               // if = 0, done; return
027  LBL 1
028  X                 // if not = 0, multiply by x
029  GTO 0
```

Now, here's a solution using Matlab (if you are fortunate to have it on a PC or Unix workstation at your desk). A private license for one copy is only $1900 -- 10x the price of a 49G!

```
C:> matlab
>> p = [4 -5 3 -2; -7 3 -6 5]   ; note order of coefs
>> x = 2.3
>> f = polyval (p(1,:),x) / polyval (p(2,:),x)
```

Thought required for programming: mimimal
Speed of results: virtually instantaneous
Using the right tool for the job: priceless!

(I would expect only those in America who have seen the TV ads for MasterCard to get the reference. My apologies...)

Of course, the "right tool" is not always available or affordable.

Loop exit trick

*Message #7 Posted by **Patrick** on 10 July 2003, 4:09 p.m.,*

in response to message #6 by Karl Schneider

Hey Karl

Nice to see some new faces in the quiz threads. Maybe Valentin will be encouraged by this and continue to post new ones! (hint, hint)

The loop trick you refer to is documented on page 176 of the HP-15C Owner's Handbook, in the section "Using Matrix Operations in a Program". If you look back at the solutions of some of Valentin's previous quizzes, you'll see this technique used there as well -- it is very handy.

The other useful part of that same feature is that, when the row and column numbers (R0 and R1) get exhausted for a particular series of matrix operations, not only is there a program line skip, but both counters are also reset to 1, obviating the need for another call to MATRIX 1. Steps 018 and 019 in my solution depend on this as well. Note in particular that step 019 is done with user mode **off**. Otherwise the division in step 020 would be skipped!

Patrick has right and a new question: Nested loops

*Message #8 Posted by **Tizedes Csaba** on 10 July 2003, 4:59 p.m.,*
in response to message #4 by Patrick

(Sorry for my poor english, I never learn it...)

Hi Patrick!

You are right in the two limitations! And must be insert 'RESULT C' instruction!

Csaba

Ps.: New quiz question for everybody: How are you programming nested loops on HP15C? I will post in my next response my version. Its not a challenge, just interest me, how it to do with another brain... :)

Csaba

Nested loops

*Message #9 Posted by **Tizedes Csaba** on 11 July 2003, 11:00 a.m.,*
in response to message #8 by Tizedes Csaba

Hi All!

I wrote this program about four years ago.

General method:

```
--------
LBL 0
1
STO I
--------
LBL 1
RCL RR2
STO 2
1
STO + I
--------
LBL 2
RCL RR3
STO 3
1
STO + I
--------
  .
  .
  .
--------
LBL n-1
RCL RRn
STO n
1
STO + I
--------
LBL n
######################
#                    #
# Loop's instructions #
#                    #
######################
ISG (i)
GTO I
--------
LBL decrement
DSE I
GTO jump
RTN
--------
LBL jump
```

```
ISG (i)
GTO I
GTO decrement
--------
```

RR2, RR3, ..., RRn contains original value of loop counters (2nd, 3rd, ..., nth registers of calculator) Before running it must be set.

An example: Calculate how many 3*4*5:

```
LBL 1
RCL 5
STO 2
1
STO + I
LBL 2
RCL 6
STO 3
1
STO + I
LBL 3
1
STO + 0
ISG (i)
GTO I
LBL 8
DSE I
GTO 9
RCL 0
RTN
LBL 9
ISG (i)
GTO I
GTO 8
```

For running:

```
1.003 STO 1
1.004 STO 5
1.005 STO 6
1 STO I
GSB 1
```

I'll continue on sunday...! I'm running now!

Csaba

Re: Quiz !: Evaluating polynomials

*Message #10 Posted by **Fernando del Rey** on 11 July 2003, 12:20 p.m.,*
in response to message #1 by Valentin Albillo

I don't have a 15C and I'm not familiar with it, but here's a solution to the original problem on the 42S:

```
01 LBL "FB"
02 STO ST L
03 INDEX "MATA"
04 I-
05 RCLIJ
06 1
07 DIM "MATB"
08 INDEX "MATB"
09 LBL 00
10 STOEL
11 LASTX
12 x
13 I+
14 FC? 77
15 GTO 00
16 RCL "MATA"
17 RCL "MATB"
18 x
19 STO "MATC"
20 INDEX "MATC"
21 RCLEL
22 I+
23 RCLEL
24 /      (divide)
25 RTN
```

This one takes about 3 or 4 seconds for polinomials of degree 19.

I have another solution in 17 steps, but it is slower (6 to 7 seconds for the same case).

Regards to all!

My original solution

*Message #11 Posted by **Valentin Albillo** on 11 July 2003, 12:57 p.m.,*
in response to message #1 by Valentin Albillo

Hi all,

Thanks to all of you who posted your solutions and ideas on this 15C Quiz/Challenge. I've read all of them with utmost interest, you're quite a clever audience, indeed ! . My original solution for the HP-15C, which works for polynomials of degree 1 and upwards (a polynomial of degree 0 is a *constant*, and thus hardly needs to be evaluated for arbitrary x arguments), was as follows:

```
01   LBL A
02   STO I
03   RCL DIM A
04     1
05   DIM B
06   MATRIX 1
07 u STO B
08   LBL 0
09   RCL* I
10 u STO B
11   GTO 0
12   RCL MATRIX A
13   RCL MATRIX B
14   RESULT C
15     *
16 u RCL C
17   RCL/ C
18   RTN
```

and as you can see, Csaba arrived at a nearly identical solution, the main differences being:

- he omitted the *necessary* RESULT C. Csaba's solution does use matrix C for the result, but he forgot to add RESULT C, which is needed, as the default result matrix upon initial conditions (memory reset), is *matrix A*, unless otherwise changed.
- he ended the routine with uRCL C, RCL C, /, but my original uRCL C, RCL/ C, using *recall arithmetic*, saves one program step and runs slightly faster.

Apart from that and minor changes in the names of labels, etc, Csaba's routine is basically identical to my original, which reinforces the impression that the solution is very probably *unique* and can't be significantly shortened or altered.

A couple of comments on the solution:

- though this routine runs twice as fast as a direct approach evaluating the polynomials using Horner's rule, it actually performs *more* arithmetic operations, because you must construct the x-powers vector, which needs n extra multiplications. The speed is achieved because the HP-15C is optimized for numeric computations, but not for branching, step-decoding, etc, so the more time we can spend within its microcode program, the better. In this case, once the x-

powers vector is created, all the work of actually evaluating both polynomials is done by the single "*" at step 15, thus saving so much time that it more than pays to create the vector in the first place.

- In view of the previous point, we would save even more time if we could avoid the powers loop. However, I've found no way to create all the powers non-iteratively, using some sequence of matrix operations, without external looping. It would be very interesting if some way to do it was discovered.

Thanks again to everyone who did participate, see you in a next challenge.

Best regards from V.

Edited: 11 July 2003, 1:09 p.m.

[Return to Index | Top of Index]

Go back to the main exhibit hall