

## HP Forum Archive 14

[ [Return to Index](#) | [Top of Index](#) ]

### Solution times for matrix problems

Message #1 Posted by [Palmer O. Hanson, Jr.](#) on 7 July 2004, 11:51 p.m.

Valentin's submission "Re: Your 15C results are \*wrong\*!" of 6:39 on July 1 ended with the following statement: "A final advice, if you would: as I understand you're working in some project re the friendly HP-TI competitions, it would be most proper to thoroughly test and validate each claimed program running times, results, and the like, made by both TI and HP sides, lest you'll find yourself embarrassed by non-reproducible, provably erroneous results. Unlikely perhaps, but a rigorous approach is mandatory for scholar-type projects." Shortly after that submission he realized that he had not reproduced my results because he was not working the same problem. As a result of his comment I belatedly realized that I have not addressed running times in my discussions. So, here goes, and if readers don't agree with my results or conclusions I am confident that they will let me know.

In my mind we need to focus on the total solution time including both the time to enter the test matrix, the running time, where running time includes the time to calculate the determinant if that is required as with the HP-41 Math Pac and the TI-59 Master Library, and the readout time. As a place to begin I used the 4x4 inversion from Kahan's paper "Mathematics Written in Sand". With the test matrix in place the approximate times for data entry and run times in seconds until indication that the inversion was complete are

Machine/Program	Entry	Run	Total
HP-15C (from Kahan's paper)	40	11	51
HP-41 Math Pac MATRIX	65	40	105
HP-41 Advanced Solutions MATRX	65	7	72
HP-28S (/ or INV)	45	2	47
TI-59 Master Library (ML-02)	35	90	125

where the long entry times for the HP-41 programs are due to the need to wait for the display to settle after one entry before making the next entry. If the user goes too fast he will get false entries. For the HP-28S and the TI-59 I simply can't enter the data so quickly that I can induce false entries. I am not sure what the HP-41 programs are doing during data entry. Years ago there were some TI-59 matrix inversion programs that did some processing in response to each input before accepting the next input. That does not appear to be the case with the HP-41 programs since I can recall the entered input data from memory after completing the input. Perhaps the delay is due to generation and display of prompting messages. If you know, let me know. My method for the HP-28S is to establish a 4x4 matrix by the 4 ENTER IDN sequence and enter the elements of the matrix with PUTI commands.

I did not include the times required to read out the solution in the table above. There is some short, fraction-of-a-second running time between display of individual elements for the HP-41 and TI-59 solutions. For the TI-59 there is an additional two second delay as the output switches from the end of one column to the next. That delay is used in decoding the pivoting index to identify which column to display next. For the HP-41 Math Pac program the delay when moving from one column to the next is over fifteen seconds. I think that time is actually used to calculate the elements in the next column. So far as I can tell the program delivers one column of solution elements at a time to R35 through R38. If you disagree, let me know.

There are also differences in the availability of the solution elements for further calculations. The HP-41 Advanced Solutions (MATRX) program leaves them in order, row by row, starting at R01. The TI-59 ML-02 leaves them in columns starting at R08, but the order of the columns may be scrambled if pivoting was required during the solution. The HP-28S leaves them in a matrix in level 1 where they can be operated on as desired. As noted above I'm not sure exactly what the HP-41 Math Pac solution leaves in memory

## Re: Solution times for matrix problems

Message #2 Posted by [Valentin Albillo](#) on 8 July 2004, 4:51 a.m.,  
in response to message #1 by Palmer O. Hanson, Jr.

Hi, Palmer:

Palmer wrote:

*"In my mind we need to focus on the total solution time including both the time to enter the test matrix, the running time, [...] and the readout time."*

I don't concur. I think the only significant time here is the running time, not the input/output time. Taking them into account introduces a strongly subjective factor, because times then will depend on the ability of the particular user doing the tests, and the methodology he/she uses to do the input/output, both of which are highly subjective.

Just for instance, you give 40 seconds to input Kahan's 4x4 matrix into the HP-15C while I've timed myself to do it in 30 seconds, routinely, instead of your listed 40. That makes a significant difference to the totals. Also, you take quite different times if you're reading the numbers from some paper or screen one at a time, or else if you memorize a row of them and then enter them without further looking. Finally, any user is likely to be more used to a particular machine, and be more comfortable with it than with others, so further biasing the times.

There's also the fact that normally the operation under test (a matrix inversion in this particular case) means little by itself, being usually called from another program or routine. In that usual case the one and only relevant info is the time for the inversion, as the calling program will have stored or computed the input elements and will take care of using the output values for whatever it needs them, all of them or perhaps just some. This being so, including human input/output times is useless at best and misleading at worst, and can seriously bias results.

So, in my very humble, particular opinion, unless you time only the computation itself, excluding input/output times, your results aren't going to be as significant, objective, and useful as they should.

As a final comment, you don't include results for the HP-71B. Mine does invert this matrix in under 2 seconds. And relevant to what I've just stated, input time *\*absolutely\** depends on your methodology, i.e:

- You can input data using a loop, like this:

```
10 DIM A(4,4) @ FOR I=1 TO 4 @ FOR J=1 TO 4 @ INPUT A(I,J) @ NEXT J @ NEXT I
```

- Or, you can input the whole matrix at once, elements separated using commas:

```
10 DIM A(4,4) @ MAT INPUT A
```

- Or, you can put the elements in a DATA statement and read them all at once:

```
10 DIM A(4,4) @ READ A()
20 DATA element1, element2, ..., element 16
```

- Or, you can enter the elements into a file with some program or editor, then have the matrix inversion program read them all at once, like this:

```
10 DIM A(4,4) @ ASSIGN #1 TO "MYDATA" @ READ A() @ ASSIGN #1 TO *
```

As you can see, all are valid input methods and you'll use one or the other depending on the particular need or your personal taste. I, for instance, tend to use a lot the DATA method, because you can very easily make corrections, additions or deletions by simply editing the DATA statements, instead of relying in the command stack or re-typing.

And of course, all of them will give vastly different input times, even not considering the (considerable) variation in physical dexterity to key in the values among the diverse human testers. Not to mention output times, which I'd rather not touch here.

Best regards from V.

*Edited: 8 July 2004, 5:16 a.m.*

### **Re: Solution times for matrix problems**

*Message #3 Posted by **Palmer O. Hanson, Jr.** on 9 July 2004, 12:13 a.m.,  
in response to message #2 by Valentin Albillo*

Thank you, Valentin, for your comments and for the run time information on the HP-71B. I agree that we need to be able to call module programs as subroutines of our own programs, but I do not want to give up the ability to use the module programs efficiently as stand alone problem solvers. I also want to be able to operate directly with the output of the module programs; for example, I want to be able to subtract the exact value of an element of an inverted matrix from the displayed value, and then divide the result by the exact value, see the relative error in the display, press R/S and see the next element of the solution. I can do that with the output of the MATRX program of the HP-41 Advanced Solutions Pac, and with the output of the ML-02 program of the TI-59 Master Library, but not with the MATRIX program of the HP-41 Math Pac. There may be a way to do what I want with the output of the MATRIX program but I haven't figured out how to do it. For TI-59 users being able to do arithmetic on the displayed solution was particularly useful since it was the way the user looked at the guard digits.

I suspect that my preference for the fastest possible data entry is tied to the good, old days when a mainframe ran for many minutes to get solutions which we now obtain in seconds with hand-held devices.

While working through the solution times and reviewing the matrix inversion results I have presented it suddenly occurred to me that I may have become more than a little spoiled by the capability that has been developed over the years. I am impatient when the inversion of an 8x8 matrix takes more than a few seconds. I think of a program as marginal when I find that it can't get exact answers to the inversion of an 8x8 Hilbert. I thought back to another time in the late 1970's when the HP-67 and the SR-52 were the flagship portable, programmable calculators for their manufacturers. Last year I had purchased an HP-67 and I recently had it repaired. A Standard Pac of magnetic cards was included in my purchase. I noted that matrix operations were limited to 3x3. I also remembered that back in 1977 programmers such as Barbara Osofsky on the SR-52 and Hal Brown on the HP-67 had managed to overcome the severe memory limitations of those machines and achieve 4 or 5 place accuracy for inversion of the 5x5 Hilbert matrix. In a Biblical sense those individuals were truly "Mighty men (and women) ... ..of renown."

Here's an interesting item. When I invert a modified 3x3 Hilbert (a standard 3x3 Hilbert multiplied by 60 to yield all integers in the test matrix) with the 3x3 program in the HP-67 I get an exact answer. I haven't been able to get an exact answer with either HP-41 program, with the HP-28S or with the TI-59. I suspect that the HP-67 succeeds because it uses a much more straightforward solution for the inverse as defined on page 10-01 of the manual. I don't remember seeing that form before. I'll try to refresh my failing memory with a look at my old copy of Fadeeva when I am at our winter home this weekend.

### **Re: Solution times for matrix problems**

*Message #4 Posted by **Valentin Albillo** on 9 July 2004, 7:30 a.m.,  
in response to message #3 by Palmer O. Hanson, Jr.*

Hi, Palmer:

Palmer posted:

*"I noted that matrix operations were limited to 3x3"*

Standard Pac's programs were meant as simple examples for the user to get to know his/her new calculator, and they absolutely lack any degree of sophistication or advanced programming techniques. With proper programming and using just a one-card program and no data cards, you could still solve a 7x7 system of linear equations in an HP-67, a 12x12 tridiagonal system of

equations, invert a 5x5 matrix in place, or do 4x4 matrix operations within a single program, to name a few.

*"When I invert a modified 3x3 Hilbert (a standard 3x3 Hilbert multiplied by 60 to yield all integers in the test matrix) with the 3x3 program in the HP-67 I get an exact answer. I haven't been able to get an exact answer with either HP-41 program, with the HP-28S or with the TI-59."*

Inverting a 3x3 matrix is so easy you can do it on your head, I've done it a number of times when sleep wouldn't come or just for fun. The HP-67 program doesn't try for any general, NxN algorithms (such as diagonalization, Gauss-Legendre, exchanges, etc) but simply uses the exact solution for the 3x3 case, inverting the matrix by minors. This entails only multiplications, which are exact if the matrix elements are small enough integers, and a final scalar division of all elements by the value of the determinant. This division also comes out exact, so you're never working with approximate real numbers or rounding. On the other hand, the programs for the machines you mention (and the HP-15C, 71B, etc) do use general NxN algorithms, that entail divisions all the time as part of the reduction process. This is bound to generate inexact results, which becomes important if the original matrix is ill-conditioned. The HP-67 approach doesn't suffer from this, but would be prohibitively slow and cumbersome if used for larger matrices.

Anyway, Palmer, I think you're far too worried (obsessed one might say) with top accuracy, and precision and all that jazz, struggling to get those elusive and exclusive TI guard digits into the light, etc, etc. Perhaps a little example will help you take it easier. Here, take your favourite calculator(s) among the 500+ you own, and evaluate this simple polynomial:

$$1335/4*y^6 + x^2*(11*x^2*y^2-y^6-121*y^4-2) + 11/2*y^8 + x/(2*y)$$

for  $x=77617$  and  $y=33096$ . Compare your results among the diverse calculators and computers, and tell me your best verdict as to what the result is.

Best regards from V.

### Re: Solution times for matrix problems

Message #5 Posted by [bill platt](#) on 9 July 2004, 12:55 p.m.,  
in response to message #4 by Valentin Albillo

This is a trick question.

Quote:

$$1335/4*y^6 + x^2*(11*x^2*y^2-y^6-121*y^4-2) + 11/2*y^8 + x/(2*y)$$

for  $x=77617$  and  $y=33096$ . Compare your results among the diverse calculators and computers, and tell me your best verdict as to what the result is.

It depends on the order in which you sum the terms. All but one of the terms are very large:

term 1 is about 4.39 E 29  
term 2 is about -7.917112 E36  
term 3 is about 7.917111 E36  
term 4 is about 1.1726

The combination of terms (2) and (3) is about -4.39 E29

So, if you add terms (2) and (3), then term (1), you may get zero, which is added to 1.1726, which is a red herring.....

Or, depending on round-off, you might find that (term 2) + (term 3) + (term 1) = 3.58 E25

Or, if you add (term 1) + (term 2) + (term 3) you will get exactly 4 E25

So, the point is that addition is not truly commutative on a calculator---one must pay attention to the precision and the floating points!

I suspect that if I did a manual solution, I would find that terms 1,2,3 should cancel perfectly. At least that would be a fitting result, considering the author of this quiz :-)

By the way, the numbers I generated up above are from the 32s, which I programmed, using registers "x" and "y" and calling them, and storing each term into a register (a b c d)

When I put the whole mess into the command line on my 30s (the cheap hp-kinpo-compaq thingie) I got 1.17260394---in other words, the first three terms seemed to have cancelled perfectly.

so, perhaps the 30s is "better" than the old RPN?

Can someone please test this on the 33s?

Best regards,

Bill

Regards,

Bill

*Edited: 9 July 2004, 12:59 p.m.*

### **Re: Solution times for matrix problems**

*Message #6 Posted by **bill platt** on 9 July 2004, 1:22 p.m.,  
in response to message #5 by bill platt*

More:

If you solve it in Excel, you have commutative issues, too:

Make variables x and y.

Then, solve each term on its own (you can copy Valentin's formulas \*exactly\* into the cells)

Now, do it so that you have a column, with:

```
term 1 ; {4.39 e39} cell t1
term 2 ; {-7.917...E36} cell t2
term 3 ; {7.917...E36} etc
term 4 ; {1.1726}
```

Now, if you use the "Sum" command on this column, you get:

```
-1.18059E+21
```

If you sum term1 + term2, and add that result to term3, you get:

```
t1 + t2 = -7.917...E+36,
and the final result of (t1 + t2) + t3 = 0
```

On the other hand, if you do:  
(t2 + t3) + t1, you will get, -1.390061E+21

If you write: =t1 + t2 + t3, you will get exactly 0.

*Edited: 9 July 2004, 1:24 p.m.*

### **Be Careful when Mixing floating point numbers**

*Message #7 Posted by [bill platt](#) on 9 July 2004, 1:51 p.m.,  
in response to message #6 by bill platt*

The basic problem is most easily seen in term 2:

$$x^2*(11*x^2*y^2-y^6-121*y^4-2)$$

$y^6$  is on the order of  $10^{27}$   $121*y^4$  is on the order of  $10^{20}$  Adding them together on a calculator, you really only feel the very first couple of digits of the  $10^{20}$  portion being added to the  $10^{27}$  portion.

So, when you subtract "2" it is not even felt---it doesn't really happen.

The moral of the story here: watch what happens with the numbers.

It is also instructive to note that it is very much easier to realize what is happening with something like this, if you actually go through it step by step---because you will see the magnitudes and have a chance to realize what is going on.

Whereas, if you use the fancy programmability, or you type into a formula window and ask for the answer, you may not see or realize the nuances---unless you bothered to look initially.

This is a lesson which should be taught to all schoolchildren, when they get to the stage where they are using machines for computation. It is a classic problem in (machine) numerical analysis.

And, it should be reminded to the rest of us, too! (thanks, Valentin).

Best regards,

Bill

### Yes, yes. And the result is ? [NT]

Message #8 Posted by [Valentin Albillo](#) on 12 July 2004, 5:11 a.m.,  
in response to message #7 by bill platt

Best regards from V.

### Re: Yes, yes. And the result is ? [NT]

Message #9 Posted by [Werner Huysegoms](#) on 12 July 2004, 10:51 a.m.,  
in response to message #8 by Valentin Albillo

the exact result is  $-54767/66192 = -0.827396059947$   
but if I key the exact formula in and evaluate it in approx mode, it returns 3.076e25

This is how I can get the result in approx mode:

order by powers:

$$\begin{aligned} & 11/2*y^8 - x^2*y^6 \\ & + 1335/4*y^6 - 121*x^2*y^4 + 11*x^4*y^2 \\ & - 2*x^2 \\ & + x/(2*y) \end{aligned}$$

Take  $t=11*y^2$

$$\begin{aligned} & y^6*(t - 2*x^2)/2 \\ & + 1331/4*y^6 + y^6 - 121*x^2*y^4 + 11*x^4*y^2 \\ & - 2*x^2 \\ & + x/(2*y) \end{aligned}$$

or

$$\begin{aligned} & y^6*((t - 2*x^2)/2 + 1) \\ & + t^3/4 - x^2*t^2 + x^4*t \\ & - 2*x^2 + x/(2*y) \end{aligned}$$

$$\begin{aligned} & y^6*((t - 2*x^2)/2 + 1) \\ & + t/4*(t - 2*x^2)^2 \\ & - 2*x^2 + x/(2*y) \end{aligned}$$

In this form, it can be evaluated as follows:

```
x := 77617;
y := 33096;
```

```
y2 := y*y;
x2 := x*x;
```

```
t := 11*y2;
u := t - 2*x2;
```

```
f := (1+u/2)*y2^3 + t/4*u^2 - 2*x2 + x/(2*y)
```

```
yields -0.82739605995
```

### Re: Solution times for matrix problems

Message #10 Posted by [Martin cohen](#) on 9 July 2004, 2:41 p.m.,  
in response to message #5 by bill platt

On my 33s (which I am coming to really like, aside from the decimal point visibility/point-comma confusion), I get:

(Note: All results are the full results displayed with the "ALL" Display option)

term 1 = 4.3860575085E29 term 2 = -7.9171117992E36 term 3 = 7.9171113406E36 term 4 = 1.17260394005

If I do (1)+(2)+(3)+(4) I get -2.E25

If I do (2)+(3) I get -4.3863E29

If I then add (1) I get -2.4249155E25

So (1)+(2)+(3) is not the same as (2)+(3)+(1)

I don't have my 49G+ with me, so I can't evaluate it exactly.

For those who are interested, Section 4.2 of Knuth's "The Art of Computer Programming" and Chapter 1 of Wilkinson's classic "Rounding Errors in Algebraic Processes" (published in 1963 and still useful) have good discussions of these sorts of things. Kahan has good stuff also.

Martin Cohen

### Very interesting. And the result is ? [NT]

Message #11 Posted by [Valentin Albillo](#) on 12 July 2004, 5:16 a.m.,  
in response to message #10 by Martin cohen



Best regards from V.

**Re: Solution times for matrix problems,**

Message #12 Posted by [Veli-Pekka Nousiainen](#) on 12 July 2004, 8:43 a.m.,  
in response to message #4 by Valentin Albillo

-54767/66192 (if the numbers are base #10d)  
(VPN)

**Re: Solution times for matrix problems,**

Message #13 Posted by [Valentin Albillo](#) on 12 July 2004, 9:27 a.m.,  
in response to message #12 by Veli-Pekka Nousiainen

Hi, VPN:

Care to do it in floating point, please ? Exact integer arithmetic is useless for discussing floating point implementations and their problems. Not to mention a single square root,  $e^x$  or trig function renders it useless. Or perhaps your machine can't cope unless 'cheating' by doing it the integer way ?

Speaking of integers, I'm still waiting for you to find out the product of the 3 prime numbers I suggested you to try. Or will you give a floating point answer instead ? ;-)

Best regards from V.

**Re: Solution times for matrix problems,**

Message #14 Posted by [Veli-Pekka Nousiainen](#) on 12 July 2004, 12:20 p.m.,  
in response to message #13 by Valentin Albillo

if ENTERed as stated it goes to zero  
even with 100 'DIGITS' STO and using LongFloat library

"Speaking of integers,  
I'm still waiting for you to find out the product of the 3 prime numbers I suggested you to try.  
Or will you give a floating point answer instead ? ;-)"

I have totally forgotten that during my vocation  
Care to repeat the question?  
{VPN}

**Re: Solution times for matrix problems,**

Message #15 Posted by [Valentin Albillo](#) on 13 July 2004, 7:15 a.m.,  
in response to message #14 by Veli-Pekka Nousiainen

Hi, VPN:

VPN posted:

*"if ENTERed as stated it goes to zero even with 100 'DIGITS' STO and using LongFloat library"*

Then something's surely wrong with your "LongFloat" implementation because 40 digits are more than enough to give an accurate result, let alone 100. If 100 'DIGITS' STO (that is, DIGITS = 100 for us straight people) produce a zero result like that, I wouldn't trust that 'LongFloat' library at all.

*"I have totally forgotten that during my vocation Care to repeat the question?"*

Not really. I deleted the text file with the question. Anyway, it wasn't that important, just a chance for you to test your machine's long integer precision as well as delighting in the beautiful result. By the way, I didn't know that 'vocations' would have such negative side effect on memory. Are you applying for priesthood or such ?

Best regards from V.

### **Re: Solution times for matrix problems,**

*Message #16 Posted by [Veli-Pekka Nousiainen](#) on 13 July 2004, 10:16 a.m.,  
in response to message #15 by Valentin Albillo*

My bad - the solution with LoangFloat was correct, but I read it wrong.  
The exponent was -100, but it uses integers and since digits was 100, the result only indicated it was less than 1 (instead of being  $1^{-100}$  eg. ~zero).  
The answer is  
"-8273960599468213681411650954798162919990331157843848199178148416727096930142615421803239062122310850.E-100"  
I think I used FMULT instead of FY^X  
but the answer looked the same.

The holidays always make me to forget things.  
Hmmm.maybe I AM applying for priesthood?  
[Veli-Pekka = Brother-Peter]

### **Re: Solution times for matrix problems**

*Message #17 Posted by [Palmer O. Hanson, Jr.](#) on 16 July 2004, 9:35 a.m.,  
in response to message #4 by Valentin Albillo*

You wrote:

Standard Pac's programs were meant as simple examples for the user to get to know his/her new calculator, and they absolutely lack any degree of sophistication or advanced programming techniques. With proper programming and using just a one-card program and no data cards, you could still solve a 7x7 system of linear equations in an HP-67, a 12x12 tridiagonal system of equations, invert a 5x5 matrix in place, or do 4x4 matrix operations within a single program, to name a few.

Where can I find this program?

### **Re: Solution times for matrix problems**

*Message #18 Posted by [Valentin Albillo](#) on 16 July 2004, 10:43 a.m.,  
in response to message #17 by Palmer O. Hanson, Jr.*

Hi, Palmer:

Palmer posted:


*"Where can I find this program?"*

Actually, they're four different programs, and they'll get published in [Datafile](#) within a few months.

Best regards from V.

---

[ [Return to Index](#) | [Top of Index](#) ]

 [Go back to the main exhibit hall](#)