HP Forum Archive 15

[Return to Index | Top of Index]

Even More Results with Matrices

*Message #1 Posted by **Palmer O. Hanson, Jr.** on 29 May 2005, 11:10 p.m.*

During the thread "More matrix results on HP and TI machines" back in April Gene Wright posted the determinant and the inverse for Valentin Albillo's matrix number 1 as obtained with the TI CC-40 Mathematics module. Valentin requested some additional results. One of the major deficiencies of the CC-40 is that when a Solid State Cartridge program is operated from the keyboard the results are brought to the display in a form that does not permit additional calculations on the result without reentering the value. Thus, the results that Valentin requested are not easy to obtain when using the CC-40 Mathematics module from the keyboard. The results are readily obtainable when calling the routines in the Mathematics module from a program. I was busy for most of May with moving to our summer home and with the marriage of my son. I have finally written programs that access the Mathematics module routines. Results obtained for Albillo 1 follow.

Determinant of Albillo 1 to more significant figures: 1.0028267103

Determinant of the computed inverse of Albillo 1: -0.9971504614

Determinant of the product of Albillo 1 and its computed inverse: 0.997159035

The incorrect sign for the determinant of the computed inverse is an error that occasionally occurs with the Mathematics module in the CC-40. I noted a single occurence of that error in the early days of the CC-40 and reported the problem to TI. The error was eliminated in the Mathematics module for the TI-74.

Product of Albillo 1 and its inverse:

```
 1.0001  -0.000002   0          0         0        0.0001   0
 0        1          0         -0.0001    0        0.0001  -0.0001
-0.0001   0          0.999999   0.0001    0.0001   0        0
-0.0001   0          0          1.000007 -0.0001   0.0001   0.0002
-0.0002  -0.000001   0.000001   0.0001    0.9999   0.0001   0.0002
 0        0          0.000001   0.0001    0.0001   1        0.0001
 0.003    0.000009  -0.000016   0.0001   -0.0003   0.001    0.9971


Row norm =           0.007325
```

```
Column norm =        0.0035
Frobenius norm =     0.0043359421
```

where, as suggested by Rodger Rosenbaum, the identity matrix was subtracted from the product of the matrix and its inverse before calculatiing the norms. Many of the elements, the ones and zeroes, seem to be to good to be true, and I have always paid homage to the old adage that says "If it seems too good to be true, it probably is." The calculation of the (1,1) element of the product of the matrix A and its inverse C is A(1,1)x C(1,1) + A(1,2)xC(2,1) ... ... A(1,7)xC(7,1) and listing the products of the actual matrix elements in order yields

```
58  x   96088630.278672         =     5573140556.1630
71  x       4467.3718497165     =         317183.40132987
67  x     -39324.8401.9987      =       -2634764.293391
36  x     272469.8067209        =        9808913.041952
35  x   -1840970.110491         =      -64433953.867185
19  x   13113279.556936         =      249152311.5818
60  x  -96089170.750456         =    -5765350245.0274


     Pencil and Paper  Sum      =              1.00010587


     CC-40 Math Module Sum      =              1.0001
```

where the program completes the sum of the products in sequence and in essence throws away digits beyond the fourth digit to the right of the decimal point because of the larger magnitude of the first product. One can obtain the "Pencil and Paper" sums in a program by summing the integer and fractional portions of each product separately and and combining the two sums when storing the calculated element in the solution matrix. I admit that in the real world one must be careful when playing around with numbers in this way but I did it anyway out of curiosity. The revised product of Albillo 1 and its inverse is

```
 1.00010587   -0.0000005434   0.0000001352   -0.0000158544    -0.0001134     0.0001122    0.00000113
 0.00001387    0.9999994058  -0.0000003548   -0.0000050844    -0.00005402    0.00005084  -0.00001387
-0.00009634   -0.0000003764   0.9999994508    0.000040054      0.00000726    0.00002209   0.0000548
-0.00009494    0.0000004173   0.0000002775    1.000007241     -0.00009058    0.00004022   0.00010816
-0.00020652   -0.0000014466   0.0000002842    0.0000195828     0.99997651    0.00000264   0.00021756
-0.00004683   -0.0000004497   0.0000006379    0.000049836      0.000046956   1.00005925   0.0000562
 0.00294259    0.0000088494  -0.0000161785    0.0001265592    -0.0035169     0.00094542   0.99710385


Determinant =       0.9972511401
Row norm =          0.0072874371
Column norm =       0.00350696
Frobenius norm =    0.0042746581
```

Not surprisingly, the norms did not change very much.

Some arithmetic on another machine:

Back in the 1980's one of the benchmark tests that we used was the solution of a set of linear equations where the matrix was the seventh order sub-Hilbert A(i,j) = 1/(i + j) and the vector was a column of seven ones. The exact solution and the solution obtained using the HP-41 Math Pac are

Exact	Math Pac
56	56.666735
-1512	-1527.383190
12600	12712.24137
-46200	-46566.49600
83160	83755.01020
-72072	-72541.81401
24024	24167.84911

which yields an RMS relative error for the Math Pac solution of 0.859E-02. It was suggested that we should use another way of assessing the quality of the result, namely, multiplying the matrix by the solution and comparing the result to a column of ones. When I multiplied the first row of the matrix by the solution and added in order from the top, i.e., .3333333333 x 56.666735 + ... I received exactly one as the answer. Of course, that set off my ingrained suspicion of answer that seem to be too good to be true. I added the products in the reverse order and received 0.999999600 as the answer. I calculated the sums for the remaining elements as

Forward	Back
1.000000000	0.999999600
0.999997000	0.999997160
1.000003000	1.000002750
0.999999000	0.999998900
0.999998000	0.999998235
0.999997000	0.999997060
0.999998000	0.999997575

where for this problem it appears that the arithmetic is not consistent with the commutative law of addition. Examination of the details of the calculation shows that the effect are similar to those seen with the product of Albillo 1 and its inverse on the CC-40. For the fifth element above the calculations are

```
1/6  x     56.666735  =        9.444455835
1/7  x  -1527.38319   =     -218.1975886
1/8  x  12712.24137   =     1589.030171
1/9  x -46566.496     =    -5174.055111
1/10 x  83755.0102    =      837.50102
1/11 x -72541.81401   =    -6594.710365
1/12 x  24167.84911   =     2013.97426
```

where the pencil and paper sum and the backward sum are both 0.999998235. The forward sum throws away some of the less significant digits from the first two products when the third product is added..

Still to be completed: CC-40 solutions for Albillo 2 and Albillo 3.

CAUTION: I do not have my printer with me so all of the presented results have been transferred by hand from the display of the CC-40 or HP-41 to my word processor. I have proofread the numbers twice, but the possibility exists that I may have entered a typo or two. I submit in my defense the following quotation from page 634 of Volume I of Donbald Knuth's The Art of Computer Programming -- "Any inaccuracies in this index may be explained by the fact that it has been prepared with the help of a computer." In that spirit I note the following typographic errors in the earlier thread:

1. In Rodger's submission of April 24 the numerical sequence after the sixth paragraph should have one more zero in each denominator.

2. In Gene's submission of April 27 of more significant digits for the inverse of Albillo 1 th last element in he third row should have a decimal point between the 5 and the 6.

Re: Even More Results with Matrices

*Message #2 Posted by **Palmer O. Hanson, Jr.** on 30 May 2005, 11:20 p.m.,*
in response to message #1 by Palmer O. Hanson, Jr.

My April 21 submission "Some Matrix Results on HP and TI Machines" included a table of determinants for seventh through tenth order Hilbert matrices as obtained from various machines. Rodger Rosenbaum's April 24 response added "should get" calculations to my table, provided some relative error calculations, and noted that additional significant digits would allow improved results from relative error calculations. The following table adds more significant digits for determinants for seventh through ninth order. I deleted the tenth order data from my origiinal table since most machines yielded such inferior results that relative error calculations would be of little value; e.g., some machines did not even yield the correct sign.

Size	7x7	8x8	9x9
Multiplier	E-23	E-33	E-43
HP-41 Math Pac	4.820822260	2.437673496	-0.1605584046
HP-41 Advantage	4.836648xxx	2.704536xxx	6.435130xxx
HP-49 Nonexact	4.83558259986	2.73629571087	9.80241422242
HP-28S	4.83559186278	2.73636500755	9.81951403413
TI-59 ML-02	4.835806848848	2.737081996634	9.687516422225
TI-95 Math Module	4.835770481055	2.736820529910	9.728024777820
CC-40/TI-74	4.835789386188	2.736781152588	9.689026314506
TI-85	4.8357950177555	2.7370039152287	9.7212663242188
HP-49 Exact	4.83580262392	2.73705011379	9.72023431191
HP-41 should get	4.835822505	2.709988486	8.587780880
HP Saturn should get	4.83558199919	2.73630981825	9.80342803933
TI-85 should get	4.8357998501314	2.7370410057623	9.7218653313429

where the xxx's in the HP-41 Advantage row indicate that I do not have access to that module.

Using the seven significant digits in the original table Rodger calculated a relative error of -1.0029E-06 for the TI-85 solution for the seventh order Hilbert. Use of the fourteen digits available from the TI-85 reduces the relative error to -9.9929E-07. Similarly, using seven significant digits Rodger calculated the relative error for the HP-49 Nonexact as 2.068E-07. Use of the twelve digits available from the HP-49 reduces the relative error to 1.242E-07. Rodger's relative errors were calculated relative to his "should get" values which change as the mantisa length changes. The following table presents relative errors for the seventh order Hilbert with respect to the "should get" values and with respect to the exact "can't get" values that I prefer.

Machine	"should get"	"can't get"
HP-41 Math Pac	-3.1018E-03	-3.0978E-03
TI CC-40	-2.1638E-06	-2.7374E-06
HP-28S	2.0398E-06	-4.3583E-05
TI-85	-9.9929E-07	-1.5729E-06
HP-49 Nonexact	1.2422E-07	-4.5499E-05

I was uncomfortable with the values for HP-41 Math Pac where the relative error for the "should get" case is larger that the the relative error for the "can't get" case. I got access to an HP-49 and calculated the "should get" value for the HP-41. I got Rodger's value. Just one more thing I don't understand.

Re: Even More Results with Matrices

*Message #3 Posted by **Valentin Albillo** on 31 May 2005, 4:44 a.m.,*
in response to message #2 by Palmer O. Hanson, Jr.

Hi, Palmer:

Thanks for your hard work to get and post those interesting results, but may I suggest once again that using Hilbert matrices for such tests is actually a less than good idea and the three matrices I offered would actually be much better suited to the task.

The **big** problem with Hilbert matrices is that they cannot be entered *exactly* in any finite precision, so when dealing with them you're actually getting two kinds of intertwined errors, namely the error due to algorithm choice and finite-precision intermediate computations, plus the error due to *inexact* input data, which, as Hilbert matrices are so ill-conditioned, quickly renders further calculations *meaningless*.

To prove the point, I used Mathematica to *exactly* invert and solve linear systems using Hilbert matrices. Even though there were no algorithmic errors and the inversion or solution was carried out *exactly* (i.e., no errors whatsoever in the presence of exact input data), the slightest change of precision and/or rounding in the input data gave results that were orders of magnitude different from the ones for the exact, symbolic Hilbert matrix, even for moderate order N.

So, unless you explicitly want to test and deal with both kinds of errors at the same time, I suggest you put aside Hilbert matrices and stick to the ones I gave or other similar, where the input data are absolutely exact to begin with and can be stored without precision loss or rounding errors into any calculator or

computer.

Else, you'll be extracting dire and bold conclusions about the quality of the respective algorithms and/or extra-precision afforded by guard digits in models such-and-such, when, actually, your *main* source of error is caused by *inexact* input data and you wouldn't get rid of it even using exact symbolic math with infinite precision for the intermediate computations.

Best regards from V.

Re: Even More Results with Matrices

Message #4 Posted by ***Palmer O. Hanson, Jr.*** *on 31 May 2005, 11:31 p.m.,*
in response to message #3 by Valentin Albillo

Here are my results for analysis of Albillo 2 on the CC-40. I am leaving for my winter home in Florida tomorrow morning and will not be back for about ten days. At that time I will do the same analysis on Albillo 3.

Inverse of Albillo 2:

The table presents all the digits which are available. If you look closely at the table you will see that when there are an even number of digits to the left of the decimal point then there are fourteen significant figures, but when there are an odd number of digits to the left of the decimal point then there are only thirteen significant figures. This is the result of the radix 100 implementation in the CC-40. Because of this idiosyncrasy I tend to describe the CC-40 as carrying a thirteen and one-half digit mantissa.

```
 194861256.1204    696285.14407310  -1672882.975243    3126200.235442   -14847358.292932   68173386.824104  -195084326.8742
 7833.8473692431   28.128091105128  -67.307749497839   125.5751068924   -596.7313727831    2740.5414185121  -7842.8884061415
-64935.13786478   -232.0617436727    557.5509271839    1041.7670453274   4947.6402744146  -22717.95547559    65009.47801851
 506329.92428489   1809.2789169307  -4346.8907841848   8123.1709041340  -38579.53691375    177142.58994653  -506909.57662420
-4302621.442883   -15374.35145006    36938.02519037    69027.86583460    327836.15128499  -1505298.120587    4307546.981570
 25857181.340327   92393.78648257   -221983.77957013   414832.23813237  -1970175.360516    9046291.085270   -25886781.789591
-194862074.8673   -696288.15746139   1672890.007409   -3126213.295553    14847420.578420  -68173673.134954   195085146.6253


Determinant of Inverse =    -0.995424082
```

Albillo 2 x its inverse:

```
 0.98      0.000001    -0.0001       0          -0.0002     -0.0001      -0.01
 0         1.000001     0           -0.0002      0.0001      0.0001       0.01
-0.0032    0            0.99998    -0.0001      0           0           -0.0005
-0.0023    0           -0.000017    1           0          -0.0001      -0.0006
-0.0024    0           -0.000011    0           1           0           -0.0001
```

```
-0.0011   -0.000001    -0.000003    -0.000003     0           0.9998       0.0001
 0        -0.000015     0           -0.0001       0.0004     -0.0016       1.01


Determinant of product = 0.9895825906


Row norm  =        0.030401


Column norm  =     0.0313


Frobinius norm =   0.026945149
```

A comment on manufacturers:

TI selected radix 100 arithmetic for the TI 99/4, for the CC-40 and for the TI-74. I have no idea why. TI made one major mistake with the CC-40 in my estimation. They made a 3/4 size true QWERTY keyboard which means that the rows of keys are displaced by half a key from each other. The user would have been able to touch type just as with a full size typewriter except for one thing. TI put the ENTER key where the right hand SHIFT key should have been. Of course, for those who never learned to touch type this doesn't mean a thing. HP sometimes does strange things as well. Some of their machines which are advertised to have an AOS capability have parentheses as a second function. That is like making an RPN machine and having ENTER a second function.

On what problems to use to test a calculator's capability:

I think I understand your objection to the Hilberts. I think that we see things from a different point of view. I want problems which test not only what the algorithms can deliver but what the total machine can deliver. The effect of truncation of the input data as a function of the mantissa length selected for a given machine is part of the problem.

Re: Even More Results with Matrices

*Message #5 Posted by **Valentin Albillo** on 1 June 2005, 5:01 a.m.,*
in response to message #4 by Palmer O. Hanson, Jr.

Hi, Palmer:

Palmer wrote:

"I want problems which test not only what the algorithms can deliver but what the total machine can deliver. The effect of truncation of the input data as a function of the mantissa length selected for a given machine is part of the problem."

As you wish, but be aware that then you aren't justified in making some of the claims you make and most of your results related to error magnitude are meaningless.

If your input matrix is not *exactly* the Hilbert matrix (and it can never be in finite, non-symbolic computation), then you can't compare the inverse you get with the *exact* inverse of the *exact* HM, because your inverse could ideally be the absolute exact inverse of your input matrix, yet it will seem to be in large error when compared to the exact inverse of the HM.

Consider this "Gedankenexperiment" case of internal computations being ideally done *exactly*, but *you don't know it*. By comparing your computed inverse to the exact inverse of the exact HM you are:

- not recognizing that the inverse you got is the exact inverse of your input matrix, so your algorithm is perfect and your internal computations are exact, yet you don't notice at all.
- also, you are erroneously thinking that the differences you get are a consequence of algorithm plus intermediate computation errors, when in fact they are *not*, and the only source of error is that you're comparing the computed inverse of your matrix with the inverse of *another* matrix.

What's the only way out of this ? Make your input matrix exactly representable to begin with, so that the inverse you compute should indeed be the inverse you're comparing it to.

By the way, it's all but proper that I mention I made your same error in the past, I was also convinced that HMs were a good test for machines and algorithms, till I realized I was never actually handling an actual HM matrix but some approximation to it and this, for matrices of order high enough and as ill-conditioned as HMs are, is tantamount to comparing with a random matrix, the errors would be about the same.

That said, you may do as you wish of course, but if you persist in posting results for Hilbert matrices, I for one will stop paying attention to them, as I consider them meaningless and a waste of my time. Other people might think differently and/or have more time to waste, of course.

Best regards from V.

Re: What is Radix 100 Arithmetic?

*Message #6 Posted by **Marcus von Cube** on 1 June 2005, 7:56 a.m.,*
in response to message #4 by Palmer O. Hanson, Jr.

> Quote:
>
> TI selected radix 100 arithmetic for the TI 99/4, for the CC-40 and for the TI-74. I have no idea why.

Can you explain what that means and what the implications are? Is it just an implementation of BCD arithmetic or is it completely different?

My TI-74 computes 84! with a result of ~ 3.31E+126. I can enter numbers with an exponent of +127 or -128. That looks like the exponent is an 8 bit binary value.

The TI-95 uses only two decimal digits for the exponent, just like its predecessors (TI-59 and the like).

Re: What is Radix 100 Arithmetic?

*Message #7 Posted by **Valentin Albillo** on 1 June 2005, 9:16 a.m.,*
in response to message #6 by Marcus von Cube

Hi,

Have a look at this PDF document where Mr. Kahan says what he thinks about radix-100 arithmetic (see page 3), and demonstrates some of its side effects.

Best regards from V.

Re: What is Radix 100 Arithmetic?

*Message #8 Posted by **John Smitherman** on 1 June 2005, 10:31 p.m.,*
in response to message #7 by Valentin Albillo

Thanks for the link Valentin. The paper contains some good observations and arguments that are still valid but it needs to be updated with results from some of today's calculators. I believe that most will fair much better.

Regards,

John

Re: What is Radix 100 Arithmetic?

*Message #9 Posted by **John Limpert** on 2 June 2005, 4:59 a.m.,*
in response to message #6 by Marcus von Cube

I suspect that it's used to get the benefits of decimal arithmetic on systems that don't have a fast BCD instruction set. BCD arithmetic was an extra-cost option on some older systems, and was ignored or deprecated on many modern systems.