♥MoHPC♠ *The Museum of HP Calculators*

HP Forum Archive 15

[Return to Index | Top of Index]

More "Matrix Results on HP and TI Machines"

*Message #1 Posted by **Valentin Albillo** on 26 Apr 2005, 10:38 a.m.*

Hi, all:

The original thread was already quite deep and I want you to consider this. In a nutshell, **Palmer O. Hanson, Jr.** was using the Nth-order Hilbert matrix to test the respective accuracies of assorted brands and models of calculators and computers, by computing its determinant for orders 7 to 10, and then analyzing the results and drawing bold conclusions from them.

And then, after you posted a number of results and conclusions, **Rodger Rosenbaum** posted this:

*"The calculator couldn't get that result even if it could do *perfect* floating point arithmetic, because the matrix it's starting with *isn't* the Hilbert matrix."*

and I absolutely agree with him because he's absolutely correct: you're *not* computing the determinant of a Hilbert matrix to begin with, but an approximation to said matrix, because terms such as 1/3 and 1/7 are represented internally with different accuracy (i.e.: different values) in different calculators.

As the *initial* matrix being used is *not* the same, and as precisely the Hilbert matrices are *extremely ill-conditioned*, meaning that the *smallest* change in the input brings out a *large* change in the output, it's fairly obvious that the results can't be compared because, by definition of ill-conditioned matrices and by the fact that the initial matrix is extremely ill-conditioned, you would get *different* results even in the very same machine and using the very same program if you were to start with terms such as 1/3 being initially stored as 0.3333333333, then as 0.333333333333. Try it.

On the other hand, Palmer's original idea of using some suitably large, difficult (read "ill-conditioned") matrix is inherently a good accuracy test, as it requires so many arithmetic operations, and many of them carried near the limits of internal accuracy, where errors are usually largely amplified by the combined effects of the finite accuracy of the initial values' internal representations *and* the choice of basic arithmetic algorithms.

What to do ? The best of both worlds, namely:

- Let's try and use a suitably not-too-large, difficult matrix ...

- ... but let this initial matrix be exactly representable, so that the very same matrix is actually processed in all machines, and so that the results are indeed comparable and really shed some valid light on the respective accuracies.

To that effect, I propose we repeat all tests using the **"Albillo's Matrix (tm)"** (:-)) that I've carefully crafted for this thread, i.e:

```
58  71  67  36  35  19  60
50  71  71  56  45  20  52
64  40  84  50  51  43  69
31  28  41  54  31  18  33
45  23  46  38  50  43  50
41  10  28  17  33  41  46
66  72  71  38  40  27  69
```

which is a *random* 7x7 matrix (so that even the HP-15C can find its determinant, 8x8 would be too large) with quite *small* elements, yet suitably difficult indeed, as you'll see.

Of course we could have used the Hilbert matrices, multiplying all elements by some large value in order to make sure they were integer to begin with. But as you can easily check, this results in a matrix which has both very large elements and very small ones at the same time, i.e.: *very unbalanced*, which is not a fair test either. On the other hand, my 7x7 "Albillo's Matrix" consist entirely of small, two-digit integer values, *perfectly balanced*: all the elements are of the same order of magnitude.

Try it with all the machines you can. In exact arithmetic, its determinant should come out as **1**. What do you get instead ? What's the relative error ? You might be surprised.

Best regards from V.

Edited: 26 Apr 2005, 10:48 a.m.

Good idea, but how about ...

*Message #2 Posted by **Gene** on 26 Apr 2005, 10:56 a.m.,*
in response to message #1 by Valentin Albillo

Crafting two or three random matrices so that we won't have one that might, through chance, work out very well for one machine or the other?

There you are

*Message #3 Posted by **Valentin Albillo** on 26 Apr 2005, 11:53 a.m.,*
in response to message #2 by Gene

Hi, Gene:

Gene posted:

"Good idea, but how about crafting two or three random matrices so that we won't have one that might, through chance, work out very well for one machine or the other?"

There you are, you may try **"Albillo's Matrix 2"**:

```
86  69  53  33  18  10  87
70  65  63  55  20  25  73
44  53  63  67  45  44  49
32  27  50  84  36  12  33
28  41  39  64  51  45  33
11  20  13  17  19  33  15
88  73  55  35  21  18  90
```

and **"Albillo's Matrix 3"**:

```
60  53  61  65  50  37  64
49  78  67  67  24  10  50
31  52  59  60  32  19  33
18  51  50  82  35  27  21
10  18  30  36  38  12  11
 6  18  14  33  13  33  10
61  57  63  72  51  45  66
```

which is even *worse*. Both determinants are exactly **1**.

For added fun, try and compute their inverses, which also have determinants equal to 1 and integer elements. And no, I'm *not* giving the inverse matrices, see if you can compute them and to what "accuracy" (for lack of a better word ...)

Best regards from V.

Re: There you are

Message #4 Posted by ***whuy*** *on 26 Apr 2005, 5:10 p.m.,*
in response to message #3 by Valentin Albillo

Now tell us, Valentin, how did you create these examples? It's quite easy to create a matrix with det=1.. just create a random matrix L with 1's on the diagonal and likewise a matrix U with 1's on the diagonal, then multiply them together.. but how do you obtain the obvious 'balance'?? Werner

Re: There you are

Message #5 Posted by ***Rodger Rosenbaum*** *on 26 Apr 2005, 5:34 p.m.,*
in response to message #4 by whuy

Please don't tell us right away, Valentin!

And a hint for you Werner. Compute the correlation matrix for Valentin's examples and note how the row wise correlation varies as you move from row 1 to row 7.

Re: There you are

Message #6 Posted by ***John Smitherman*** *on 27 Apr 2005, 2:34 p.m.,*
in response to message #3 by Valentin Albillo

Hi Valentin. Using the TI 83+ I get the following determinants:

V.M. #1: 0.999646804 V.M. #2: 1.000499334 V.M. #3: 1.000972569

Regards,

John

results

Message #7 Posted by ***Mike (Stgt)*** *on 28 Apr 2005, 3:14 a.m.,*
in response to message #3 by Valentin Albillo

On a HP-40G (emulated) the DET of matrix 1 and 3 are 1, alas for the inverse it is _not_.

Ciao.....Mike

Re: More "Matrix Results on HP and TI Machines"

Message #8 Posted by ***Arnaud Amiel*** *on 26 Apr 2005, 11:13 a.m.,*
in response to message #1 by Valentin Albillo

The 49 gives 1 in exact or numeric mode...

Re: More "Matrix Results on HP and TI Machines"

Message #9 Posted by ***Whuy*** *on 26 Apr 2005, 11:20 a.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin. A word of caution for the HP-48G and HP-49 (don't know if it's present in the 48-S): When flag -54 is clear (as I believe it is, by default), then an extra check is performed before the determinant is calculated: to see whether all matrix elements are integer - for then, the result must be integer, too, and it is *changed* to the nearest integer! With Albillo's matrix, flag -54 clear, the result is a magical 1. With flag -54 set, it is .999945522778 The condition number being roughly 10^11, and knowing that 15 digits are used internally, it indeed delivers 4 correct digits. So, set that flag -54 on your 48's and 49's.

Cheers, Werner

Re: More "Matrix Results on HP and TI Machines"

Message #10 Posted by ***Arnaud Amiel*** *on 26 Apr 2005, 11:56 a.m.,*
in response to message #9 by Whuy

Thanks for the clarification. I could not figure out why I got 1. I tried quite a few things but didn't think of this flag as its name is not very desciptive and I never had any experience playing with it until now.

Arnaud

Moral Of The Story

Message #11 Posted by ***bill platt*** *on 26 Apr 2005, 11:30 a.m.,*
in response to message #1 by Valentin Albillo

Subtitle: why it is so hard to find young graduates who can actually do math

If you do not know matrix algebra, or use it infrequently, then do not think that the magical calculator is going to save you the time and effort to understand matrix math.

More importantly if you are rusty or ignorant with matrices, then it is dangerous to start using a calculator to do them to solve problems.

This is from my own experience.

Regards,

Bill

I second you, Bill! Also...

Message #12 Posted by ***Vieira, Luiz C. (Brazil)*** *on 26 Apr 2005, 11:37 a.m.,*
in response to message #11 by bill platt

I'd add to the following:

> Quote:
>
> More importantly if you are rusty or ignorant with matrices, then it is dangerous to start using a calculator to do them to solve problems.

More importantly if you are rusty or ignorant with matrices ***or any mid to advanced scientific subject, mainly number-related***, then it is dangerous to start using a calculator to do them to solve problems.

Cheers.

Luiz (a Brazilian engineer and teacher...)

Edited: 26 Apr 2005, 11:37 a.m.

Re: I second you, Bill! Also...

Message #13 Posted by ***EL*** *on 26 Apr 2005, 5:00 p.m.,*
in response to message #12 by Vieira, Luiz C. (Brazil)

Yes! I agree, imagine in thermodynamics, routine plug and chug...and the result is a negative Sdotgen...impossible! Yet you could get it from an equation.

EL

Re: Moral Of The Story

Message #14 Posted by ***Dave Shaffer*** *on 26 Apr 2005, 5:59 p.m.,*
in response to message #11 by bill platt

As Bill said:

"This is from my own experience."

Many of you have probably seen this, but just in case:

How do you get good judgment? From experience.

How do you get experience? From bad judgment!

More "Matrix Results on HP and TI Machines"

*Message #15 Posted by **Jeff** on 26 Apr 2005, 12:11 p.m.,*
in response to message #1 by Valentin Albillo

Welp, I hate to say it, but electronic calculators are NOT machines.

Re: More "Matrix Results on HP and TI Machines"

*Message #16 Posted by **WigglePig** on 27 Apr 2005, 9:53 a.m.,*
in response to message #15 by Jeff

Not machines? Howso?

OED defines "machine" as:

"A material structure designed for a specific purpose, and related uses."

and/or

"An apparatus constructed to perform a task or for some other purpose; also in derived senses."

Or even more succinctly:

"A complex device, consisting of a number of interrelated parts, each having a definite function, together applying, using, or generating mechanical or (later) electrical power to perform a certain kind of work (often specified by a preceding verbal noun)."

So, I propose that a calculator be known henceforth as a "machine" if the writer so wishes.

With learned regards

JasonG

Re: More "Matrix Results on HP and TI Machines"

*Message #17 Posted by **WigglePig** on 27 Apr 2005, 9:56 a.m.,*
in response to message #16 by WigglePig

And I missed this one:

"Used contextually for the particular kind of machine which the speaker or writer intends, as: a sewing machine; a printing machine or mechanical printing press; a shearing-machine (Austral. and N.Z.); a typewriter; a calculating machine or computer; a slot machine; a washing machine; etc."

Note the specific use of "calculating machine". :-)

Tra

WigglePig, reverting to ubiquitous anonymity.

Re: More "Matrix Results on HP and TI Machines"

*Message #18 Posted by **John Limpert** on 27 Apr 2005, 6:54 p.m.,*
in response to message #15 by Jeff

It's still a machine. The moving parts are just smaller :-).

Re: More "Matrix Results on HP and TI Machines"

*Message #19 Posted by **Rodger Rosenbaum** on 26 Apr 2005, 12:52 p.m.,*
in response to message #1 by Valentin Albillo

Valentin, I notice that the number 71 appears 4 times in this first test matrix you provide, and 17 appears once. Is this an unconscious indication of your love for the HP71?

Re: More "Matrix Results on HP and TI Machines"

*Message #20 Posted by **Valentin Albillo** on 26 Apr 2005, 1:13 p.m.,*
in response to message #19 by Rodger Rosenbaum

Hi, Rodger:

Rodger posted: *"Valentin, I notice that the number 71 appears 4 times in this first test matrix you provide, and 17 appears once. Is this an unconscious indication of your love for the HP71?"*

Yessss !! :-) An HP-71B with a Math ROM and plenty of RAM it's just about a dream come true for anyone who loves Numerical Analysis and wants to do complicated things fast and, above all, *easily*.

Want some proof? Just have a look at my two ***"HP-71B Math ROM Baker's Dozen"*** articles recently published in Datafile to see what you can do with this combination in at most 4 or 5 lines of extremely simple code.

Next best choices would be the HP42S, the HP-41CX + Advantage ROM, and the HP-15C, as long as we stay within HP models. Otherwise, a number of SHARP models would also be eligible.

P.D.: Apart from the four 71's and the one 17, you'll notice there are also three 41's, one 67, two 46's, two 45's, one 35, two 33's, two 31's, one 27, one 19, and one 10, to name a few !

Best regards from V.

Edited: 26 Apr 2005, 1:20 p.m.

[LONG] Commented Results for HP-71B (and HP-15C)

*Message #21 Posted by **Valentin Albillo** on 27 Apr 2005, 6:32 a.m.,*
in response to message #1 by Valentin Albillo

Hi all,

These are the results you get when processing the **"Albillo's Matrix 1"** with an **HP-71B + Math ROM** and an **HP-15C**.

As you will see, despite the "innocent", even "easy" look of the matrix (a small 7x7 matrix, with small 2-digit positive integer elements), even 15 digits of internal accuracy won't get us more than two correct digits in our results. Less internal digits, say 13, won't give even one correct digit in some computations. This is all worsened to the extreme if the results are to be used as some intermediate step in a chain calculation. You'll end up having no correct digits at all extremely soon.

If you're aware that the matrix is difficult to begin with (like those nasty-looking Hilbert matrices), you may we forewarned to extensively check the accuracy of the results you get. But if you happen to inadvertently stumble upon such an "innocent" looking matrix as this one, blindly trusting your results can result in utterly catastrophic failure. You've been warned ! :-)

The results are structured like this:

- **Exact Results:** this includes:
    - Exact Determinant

- Exact Inverse
- Exact Determinant of Exact Inverse
- Exact Product of Matrix * Inverse
- Exact Row Norm of Exact Product
- Exact Column Norm of Exact Product
- Exact Frobenius Norm of Exact Product

- **Computed Results for the HP-71B - Method 1:** this includes:
  - Computed Determinant
  - Computed Inverse
  - Computed Determinant of Computed Inverse
  - Computed Product of Matrix * Computed Inverse
  - Computed Row Norm of Computed Product
  - Computed Column Norm of Computed Product
  - Computed Frobenius Norm of Computed Product
- **Computed Results for the HP-71B - Method 2:** this includes the same results as those for Method 1
- **Computed Results for the HP-15C:** this includes:
  - Computed Determinant
  - Computed Inverse (only 1st and last elements)
  - Computed Determinant of Computed Inverse

Now for the commented results:

- **Exact Results**:

```
Albillo's Matrix 1 =


    58  71  67  36  35  19  60
    50  71  71  56  45  20  52
    64  40  84  50  51  43  69
    31  28  41  54  31  18  33
    45  23  46  38  50  43  50
    41  10  28  17  33  41  46
    66  72  71  38  40  27  69
```

```
Exact Determinant = 1


Exact Inverse =


  96360245    320206   -537449    2323650  -11354863    30196318   -96509411
      4480        15       -25        108       -528        1404       -4487
    -39436      -131       220       -951       4647      -12358       39497
    273240       908     -1524       6589     -32198       85625     -273663
  -1846174     -6135     10297     -44519     217549     -578534     1849032
  13150347     43699    -73346     317110   -1549606     4120912   -13170704
 -96360787   -320208    537452   -2323663   11354927   -30196488    96509954


Exact Determinant of Inverse = 1


Exact Product of Matrix * Inverse = 7x7 Identity Matrix =


   1  0  0  0  0  0  0
   0  1  0  0  0  0  0
   0  0  1  0  0  0  0
   0  0  0  1  0  0  0
   0  0  0  0  1  0  0
   0  0  0  0  0  1  0
   0  0  0  0  0  0  1


      Exact Row Norm of Exact Product = 1
   Exact Column Norm of Exact Product = 1
Exact Frobenius Norm of Exact Product = Sqrt(7) = 2.64575131106+
```

- **Computed Results for HP-71B - Method 1:**

```
Computed Determinant (with DET) = 0.97095056196  (error = 2.91%)


Computed Inverse (with MAT INV) = (shown truncated to integer values)


  99243204+   329786+  -553528+   2393170+ -11694584+   31099748+  -99396833+
      4614+       15+      -25+       111+      -543+       1446+      -4621+
    -40615+     -134+      226+      -979+      4786+     -12727+      40678+
    281414+      935+    -1569+      6786+    -33161+      88186+    -281850+
  -1901408+    -6318+    10605+    -45850+    224057+    -595842+    1904352+
  13543786+    45006+   -75540+    326597+  -1595967+    4244203+  -13564752+
 -99243762+  -329788+   553531+  -2393183+  11694650+  -31099923+   99397392+


Computed Determinant of Computed Inverse (with DET) =  1.05907852363 (error = 5.91%)


Computed Product of Matrix * Computed Inverse (with MAT *) =  (actual values)


   0.99558  0.0000196  0.0000278  -0.000153   -0.000064  -0.00325  0.00274
  -0.00411  1.0000161  0.000025   -0.000139   -0.000034  -0.00287  0.00245
  -0.00527  0.0000212  1.0000303  -0.000173   -0.000095  -0.00354  0.00211
  -0.00244  0.0000104  0.0000143   0.9999186  -0.00005   -0.00171  0.0011
  -0.00382  0.0000151  0.0000205  -0.000122    0.999901  -0.00243  0.00084
  -0.00334  0.0000143  0.000018   -0.000107   -0.000108   0.99783  0.00053
  -0.03492 -0.0000768  0.0001978  -0.000894    0.003433  -0.01303  1.03274


       Computed Row Norm of Computed Product (with RNORM) =  1.0852916     (error = 8.53%)
    Computed Column Norm of Computed Product (with CNORM) =  1.04948       (error = 4.95%)
Computed Frobenius Norm of Computed Product (with FNORM) =  2.65606227412 (error = 0.39%)
```

- **Comment:**

  This is a vey difficult matrix, and though the HP-71B carries 15 digits internally a significant loss in accuracy is to be expected. So we get a 2.91% error when calculating the determinant, which is good to only two digits, and an inverse which is good to only a digit and a half, at best. The error in the computed determinant of the inverse rises to 5.91%, though the product of both is actually surprisingly close to the identity matrix, despite the notorious inaccuracy of the inverse. Even so, the norms of the product are in error by 8.53%, 4.95% and 0.39%.

- **Computed Results for HP-71B - Method 2:**

```
Computed Inverse (with MAT SYS) = (shown truncated to 2 decimal places)


   96273474.83+   319931.64+ -536969.47+  2321578.89+ -11344786.58+   30170494.73+  -96423005.18+
       4475.96+       14.98+     -24.97+      107.90+       -527.53+       1402.79+      -4482.98+
     -39400.48+     -130.88+     219.80+     -950.15+       4642.87+     -12347.43+      39461.63+
     272993.95+      907.22+   -1522.64+     6583.12+     -32169.42+      85551.77+    -273417.98+
   -1844511.56+    -6129.74+   10287.81+   -44479.31+     217355.94+    -578039.24+    1847376.54+
   13138505.41+    43661.55+  -73280.55+   316827.35+   -1548230.86+    4117387.88+  -13158912.14+
  -96274016.34+  -319933.63+  536972.47+ -2321591.88+   11344850.52+  -30170664.59+   96423547.70+


Computed Determinant of Computed Inverse (with DET) = 1.01830872148 (error = 1.83%)
Computed Product of Matrix * Computed Inverse (with MAT *) =


   0.99914  -0.0000373  -0.0000001  -0.000171   -0.000299  0.00053  0.00226
  -0.00084   0.9999681   0.0000001  -0.000147   -0.000254  0.00045  0.00179
  -0.00187  -0.0000408   0.9999996  -0.000191   -0.000253  0.00054  0.00157
  -0.00076  -0.0000196   0.0000001   0.9999079  -0.000137  0.00027  0.00088
  -0.00182  -0.0000285  -0.0000001  -0.000135    0.999861  0.00036  0.00057
  -0.0018   -0.000026   -0.0000005  -0.000124   -0.00011   1.00032  0.00045
  -0.00032  -0.0000394  -0.0000051  -0.000174   -0.000424  0.00086  1.00144



      Computed Row Norm of Computed Product (with RNORM) =  1.0044644     (error = 0.45%)
   Computed Column Norm of Computed Product (with CNORM) =  1.00896       (error = 0.90%)
Computed Frobenius Norm of Computed Product (with FNORM) =  2.64599734715 (error = 0.0093%)
```

- **Comment:**

  Though the first method makes use of the obvious **MAT INV** matrix statement to compute the inverse, this is not the only way to do it nor the best, and the ***Owner's Handbook*** itself instructs us to use **MAT SYS** instead for extended accuracy and speed (though at the cost of extra memory being used). Doing so pays handsomely, as the inverse matrix comes out much more accurate, and its determinant is in error by a mere 1.83%, more than 3 times smaller than using **MAT INV** (5.91%). The product also resembles the identity matrix much more closely, and its norms show a very marked improvement (0.45%, 0.90%, 0.0093%) when compared to those of Method 1 (8.53%, 4.95%, 0.39%). This is obviously the way to go in all cases, if RAM allows.

- **Computed Results for HP-15C:**

```
Computed Determinant (with MATRIX 9) = 1.080204421  (error = 8.02%)


Computed Inverse (with 1/X) = (only 1st and last element shown)


   89205564.29 ...
               ... 89344157.45


Computed Determinant of Computed Inverse (with MATRIX 9) = 0.6635047876 (error = 33.64%)
```

- **Comment:**

  Despite the extremely difficult original matrix and the HP-15C internal algorithms being limited to 13 digits, the determinant is still within 8% error, i.e.: nearly two correct digits. However, the inverse matrix isn't that good and its determinant is already at 34% error, which means not even one decimal digit is correct.

Due to its limited memory it isn't possible to compute the product of the original matrix and its inverse and the corresponding norms.

Best regards from V.

Re: [LONG] Commented Results for HP-71B (and HP-15C)

Message #22 Posted by ***whuy*** *on 27 Apr 2005, 7:16 a.m.,*
in response to message #21 by Valentin Albillo

Valentin - what are the reasons that MAT SYS is more accurate than MAT INV? In-place inversion performs the same calculations, in a different order. On a 48G, for instance, the results of INV and / (the way to SOLVE) are not exactly the same, but almost, and certainly equally accurate (or inaccurate). And I know for sure that different algorithms are used in these cases. Cheers, Werner

Re: [LONG] Commented Results for HP-71B (and HP-15C)

Message #23 Posted by ***Valentin Albillo*** *on 27 Apr 2005, 9:41 a.m.,*
in response to message #22 by whuy

Hi, Werner:

Werner posted: *"Valentin - what are the reasons that MAT SYS is more accurate than MAT INV? In-place inversion performs the same calculations, in a different order."*

I don't think so. Inverting a matrix using **MAT SYS** is equivalent to solving N systems of N equations with N unknowns at the same time, each system giving one column of the inverse matrix.

This being so, you need extra memory (as you're solving N systems at a time, instead of one), but you only need to perform the first stage of the reduction process and you need to do it only once for all N systems. Once you've got a triangularized matrix, a simple and fast procedure of back substitution will give all columns of the inverted matrix very quickly.

On the other hand, **MAT INV** does an in-place inversion, so it needs no extra memory, but has to do a lot more work than a simple triangularization.

Best regards from V.

Re: [LONG] Commented Results for HP-71B (and HP-15C)

Message #24 Posted by ***whuy*** *on 27 Apr 2005, 10:31 a.m.,*
in response to message #23 by Valentin Albillo

Hi Valentin. I'm sorry to say but that is simply NOT true. In-place inversion follows essentially the same steps as linear equation solving - but makes implicit use of the very simple 'right-hand-side'. Both in-place inversion (well it still needs a pivot array and a workspace the size of a row if memory serves me right) and equation solving start off computing the LU-factorization

A = LU (we'll disregard pivoting)

to find the inverse, we have to find a matrix X so that

LUX = I

Following the same algorithm as normal equation solving, we must find Y so that

LY = I - equivalent to finding INV(L). There is a way of doing that 'in-place', doing the same calculations as a 'blind' elimination step, but using your advance knowledge of where the zeroes and ones are. If you write it out for a 5x5 matrix you'll see the pattern quickly.

Once Y is determined, we have to find X such that UX=Y It turns out that this 'backsubstitution step' can also be performed in-place, making use of the fact that U is upper (unit) triangular and Y lower triangular. Still, the operations are the same for both in-place inversion or equation solving, only the order of operations has changed (a bit like the difference between Gaussian elimination and the Crout method). (in the 48/49, this step is done in two substeps: calculate INV(U) and then multiply INV(U)*Y - still, the operations count is the same, but a slight difference in roundoff errors may be observed)

I may not know any longer what a correlation matrix is or what it is used for, but I'm quite certain about this.

So, therefore, I ask again: what does the 71-B User's Guide state as reason for the obvious increase in accuracy of SYS over INV? Does it do an extra 'residual correction' step?

Cheers, Werner

Re: [LONG] Commented Results for HP-71B (and HP-15C)

Message #25 Posted by ***Rodger Rosenbaum*** *on 27 Apr 2005, 10:17 a.m.,*
in response to message #22 by whuy

Werner, I believe I remember reading somewhere (other than the math pac user manual; maybe a PPC publication) that the SYS keyword does iterative refinement as part of the solution process.

Re: [LONG] Commented Results for HP-71B (and HP-15C)

*Message #26 Posted by **whuy** on 27 Apr 2005, 10:32 a.m.,*
in response to message #25 by Rodger Rosenbaum

Thanks, Rodger - that's what I thought, even if the rusty memory could only come up with 'residual correction' i.o iterative refinement ;-)

Werner

TI CC40 basic beats the HP71B!

*Message #27 Posted by **Gene** on 27 Apr 2005, 9:39 a.m.,*
in response to message #21 by Valentin Albillo

All I have done so far is compute the determinant of the first matrix on the TI Compact Computer 40 running the Matrices program in the math cartridge.

Exact Determinant = 1

CC40 computed determinant = 1.00282671

Error of 0.2827%

Wow!

Gene

The Inverse to matrix 1 from the CC40

*Message #28 Posted by **Gene** on 27 Apr 2005, 10:47 a.m.,*
in response to message #27 by Gene

Here's the determinant of Valentin's matrix #1 from the TI CC40: 1.00282671

Here's the inverse, truncated to integer values.

```
 96088630+   319303+   -535934+   2317100+   -11322856+   30111202+   -96237375+
     4467+       14+       -24+       107+        -526+       1400+       -4474+
   -39324+     -130+       219+      -948+        4633+     -12323+       39385+
   272469+      905+     -1519+      6570+      -32107+      85383+     -272891+
 -1840970+    -6117+     10267+    -44393+      216935+    -576903+     1843820+
 13113279+    43575+    -73139+    316216+    -1545238+    4109296+   -13133579+
-96089170+  -319305+    535937+  -2317113+    11322920+  -30111371+    96237917+
```

Looks like the CC40 holds up rather well! Gene

Re: The Inverse to matrix 1 from the CC40

*Message #29 Posted by **Valentin Albillo** on 27 Apr 2005, 10:57 a.m.,*
in response to message #28 by Gene

Hi, Gene:

Thanks for your results, which are indeed impressive, by the way. Some suggestions:

- As the accuracy is so good, please post the inverted matrix truncated to two decimals, not integers. This will allow better assesment for elements such as 14+ and -24+.
- Compute also the determinant of the computed inverse and the product (matrix)*(computed inverse) and post the determinant and the matrix obtained (no truncation, all digits) as well as the respective norms, if possible.
- Try the two other sample matrices you asked for, i.e.: Albillo's Matrix 2 and 3. The third one is specially difficult, easily the most intractable of all three.

Thanks and best regards from V.

Re: The Inverse to matrix 1 from the CC40

*Message #30 Posted by **Rodger Rosenbaum** on 27 Apr 2005, 11:35 a.m.,*
in response to message #29 by Valentin Albillo

Valentin said: "Compute also the determinant of the computed inverse and the product (matrix)*(computed inverse) and post the determinant and the matrix obtained (no truncation, all digits) as well as the respective norms, if possible."

Valentin, it occurs to me that what we should really do here is subtract the 7x7 identity matrix from the product of the computed inverse and the original matrix, and then compute a norm. That gives us the norm of the error; I think that's the measure we really want.

And, did you notice my (edited) post of 10:40 AM? I get substantially different results for the determinant of the transpose.

Re: The Inverse to matrix 1 from the CC40

*Message #31 Posted by **Valentin Albillo** on 27 Apr 2005, 12:23 p.m.,*
in response to message #30 by Rodger Rosenbaum

Hi, Rodger:

Rodger posted:

"it occurs to me that what we should really do here is subtract the 7x7 identity matrix from the product of the computed inverse and the original matrix, and then compute a norm. That gives us the norm of the error; I think that's the measure we really want."

I think so as well, be my guest.

And, did you notice my (edited) post of 10:40 AM? I get substantially different results for the determinant of the transpose. "

I did notice your post but not the edited version. I don't know why working with the transpose instead results in much greater accuracy, but if I remember correctly, when Palmer issued more or less essentially this same topic a few months ago (must be a personal obsession of his!), he did include a particular 4x4 matrix and some results for the HP-15C. When I tried said matrix in my HP-15C I incorrectly read rows for columns in Palmer's example and did work with the transpose instead. My results were much more accurate than Palmer's, which was rather puzzling and the time, and obviously still is.

If anyone can explain the situation, I for one would be very interested to know.

Thanks and best regards from V.

Re: The Inverse to matrix 1 from the CC40

Message #32 Posted by ***Palmer O. Hanson, Jr.*** *on 27 Apr 2005, 11:05 p.m.,*
in response to message #31 by Valentin Albillo

The HP-15C results for the 4x4 that Valentin mentioned were not mine since I don't have an HP-15. The results were extracted from Kahan's paper "Mathematics written in Sand".

It may be an obsession. It all started two years ago when Richard Nelson and I agreed to do a history of the old contests between users of TI and HP machines. Solving matrix problems was one of the contests going all the way back to the HP-25 and the SR-52. The agreed-to problems involved inverting Hilberts and solving linear equations involving hilberts. I went over the old issues of TI PPC Notes and Richard's PPC Calculator Journal and I had some questions about the results we got way back then in the late 1970's. I tried to reproduce and to understand some of those old results, reexamined the results in Kahan's paper, and here we are.

Re: The Inverse to matrix 1 from the CC40

Message #33 Posted by ***Gene*** *on 27 Apr 2005, 11:43 a.m.,* *in response to message #28 by Gene*

Slightly more significant digits.

I'll have to work on the determinant of the inverse, etc. I'm actually NOT that familiar with the CC40 (but given its accuracy, perhaps I should get that way!)

```
 96088630+      319303+   -535934+   2317100+   -11322856+   30111202+   -96237375+
     4467+       14.96+    -24.93+    107.70+     -526.51+    1400.04+    -4474.35+
 -39324.84+    -130.63+    219.38+   -948.32+      4633.9+  -12323.17+     3938567+
   272469+      905.44+   -1519.7+   6570.43+      -32107+      85383+     -272891+
 -1840970+    -6117.71+     10267+    -44393+      216935+    -576903+     1843820+
 13113279+       43575+    -73139+    316216+    -1545238+    4109296+   -13133579+
-96089170+     -319305+    535937+  -2317113+    11322920+  -30111371+    96237917+
```

Results for TI-95 PROCALC + Mathematics module

Message #34 Posted by ***Marcus von Cube*** *on 28 Apr 2005, 4:41 a.m.,*
in response to message #21 by Valentin Albillo

I entered AM#1 into my TI-95 and got the following results:

```
det=1.000416554


inverse:


 96320122.43  320072.67 -537225.22  2322682.48 -11350135.05  30183744.83  -96469226.32
     4478.13      14.99     -24.99      107.96      -527.78      1403.42      -4485.13
   -39419.58    -130.95     219.91     -950.60      4645.06    -12352.85      39480.55
   273126.23     907.62   -1523.37     6586.26    -32184.59     85589.34    -273549.05
 -1845405.29   -6132.45   10292.71   -44500.46    217458.42   -578293.11    1848262.10
 13144871.45   43680.80  -73315.46   316977.96  -1548960.77   4119196.13  -13165219.97
-96320664.2  -320074.67  537228.22 -2322695.47  11350199.02 -30183914.76   96469769.09
```

So far I was unable to compute the determinant of the inverse and there may be typing errors (I copied the values manually, too lazy to connect one of my printers...)

Re: Results for TI-95 PROCALC + Mathematics module

Message #35 Posted by ***Palmer O. Hanson, Jr.*** *on 29 Apr 2005, 12:16 a.m.,*
in response to message #34 by Marcus von Cube

I had run AM#1 on my TI-95. I did not get your answer. Then I ran the determinant of the inverse of AM#1 and I got your answer. I suspect that the reason for the difference is that you did not look closely at the subscript prompts. If you do you will see that the second prompt appears as a(2,1) which means that the user enters the matrix elements into the TI-95 by columns not by rows. That is the same convention that was used in the TI-59. TI uses entry by rows in the CC-40 and TI-74. To my knowledge all of the HP products use entry by rows.

The difference between our results is one more confirmation that the calculator generated determinant of a matrix will not be the same as the calculator generated determinant of the transpose of the matrix. The following table shows the differences for four calculators for AM#1:

```
                         AM#1           AM#1 Transpose


HP-41 Math Pac      -1.814762704         +1.495895865


HP-28S               0.970960198039       0.999403205613


TI-59                0.9983600987185      1.000223535604


TI-95                1.000676708424       1.000416554421
```

The TI-59 and the TI-95 both use 13 digit mantissas, but the TI-59 has the well known non-commutative multiply and truncates while the TI-95 does not have the multiply problem and rounds.

The HP-41 Math Pac results support the idea that AM#1 is really more difficult than a 7x7 Hilbert.

Re: Results for TI-95 PROCALC + Mathematics module

Message #36 Posted by ***Valentin Albillo*** *on 29 Apr 2005, 6:46 a.m.,*
in response to message #35 by Palmer O. Hanson, Jr.

Hi, Palmer:

Palmer posted:

"The HP-41 Math Pac results support the idea that AM#1 is really more difficult than a 7x7 Hilbert."

You bet ! :-) I carefully crafted my "Matrix Trilogy", AM#1-AM#3 to be random matrices, as innocent-looking as possible and with small, perfecty balanced integer elements, yet incredibly ill-conditioned. This makes them excellent for testing purposes.

As I see it, they have a number of important advantages when compared with Hilbert matrices:

- They have small, integer elements which can be represented exactly in any calculator/computer architecture, thus guaranteing that you start with the real McCoy. On the other hand, comparable Hilbert matrices have a large number of elements (such as 1/3, 1/7) which can't be exactly represented in any finite accuracy, so you're starting your tests with an *approximation* to the Hilbert matrix, *not* the real Hilbert matix.

  As Hilbert matrices are extremely ill-conditioned, this is the biggest No-No in the book. You can't get any meaningful comparisons among different calculators with different internal architectures, nor can your results be compared with those corresponding to the exact matrix, because your starting data are not the same. This is not the case with AM#1-3: you always start with the same, exact matrix in every calculator/computer and you can compare against exact results obtained by using some symbolic algebra package.

- Hilbert matrices aren't balanced at all. You deal with a matrix that has floating-point, inaccurate entries, of continuously decreasing size, mixing values across a number of orders of magnitude. My AM#1-3 matrices are perfectly balanced, all numbers are more or less of the same order of magnitude, which makes for a better test as the starting conditions do not imply arithmetic between disparage values right from the start.

- Hilbert matrices are symmetrical, which means you can't use them to check the different results you obtain from your algorithms when working with the matrix and its transpose. It also means some algorithms could hypothetically make use of their symmetry to apply special-purpose routines which can mislead as they won't work for general, non-symmetric matrices.

  Instead, AM#1-3 are random matrices, with no symmetry whatsoever. You can use them to test working with them in their original form versus their transposes, and no special-purpose algorithm applies to them.

- You can take the easy way out and compute the determinant of Hilbert matrices using exact formulae, which execute extremely quick for any dimension and produce results to any required accuracy, as demonstrated by my implementation for the HP-71B:

  ```
  10 ! ** Determinants for Hilbert matrices from order 7 through 10 **
  20 !
  30 FOR N=7 TO 10 @ P=1 @ FOR I=1 TO N-1 @ P=P*FACT(I) @ NEXT I @ Q=1
  40 FOR I=N TO 2*N-1 @ Q=Q*FACT(I) @ NEXT I @ DISP N;P^3/Q @ NEXT N


  >RUN


    7  4.83580262391E-25
    8  2.73705011379E-33
    9  9.72023431183E-43
   10  2.16417922642E-53
  ```

  You can't do that with AM#1-3, there's no such cop-out to obtain results. You need pretty accurate algorithms and the best architecture possible if you want your numbers to be any useable at all.

- And last but not least, AM#1-3 are incredibly ill-conditioned, in increasing order of difficulty, and will really give a run for his money to any and all algorithms, even more so than Hilbert matrices of comparable dimensions.

  If needed, matrices such as AM#1-3 can be crafted for any specific dimensions, with an exponentially increasing intractability. I fully doubt that any calculator would be able to get any significant figures at all when finding the determinant or the inverse of a 10x10 instance, not to mention solving a corresponding system of equations.

The bottom line: I heartedly suggest you put aside your "faulty" Hilbert matrices, and use instead AM#1-3. This will certainly help produce accurate and meaningful (if discouraging) results.

Thanks for your interest and best regards from V.

Re: Results for TI-95 PROCALC + Mathematics module

Message #37 Posted by ***whuy*** *on 29 Apr 2005, 9:28 a.m.,*
in response to message #36 by Valentin Albillo

Please don't leave me (us?) in the dark any longer, Valentin! How did you create them?

Werner

Re: Results for TI-95 PROCALC + Mathematics module

Message #38 Posted by ***Palmer O. Hanson, Jr.*** *on 29 Apr 2005, 10:14 p.m.,*
in response to message #36 by Valentin Albillo

One advantage that you didn't mention for your matrices is that they are easier to display so that one can check one's entry. Of course this comment doesn't apply to the HP-49 where one can get an exact Hilbert of any size entered with a few keystrokes.

In his transmission of April 27 Rodger wrote:

"All other things being equal (ceteris parabus, as the Romans would say), it is better to have more digits."

My original comparison table of Hilbert results which started this whole exercise supports that statement. I have't seen anything in the discussion of your matrices which contradicts that statement. Do you disagree?

Re: Results for TI-95 PROCALC + Mathematics module

Message #39 Posted by ***Marcus von Cube*** *on 29 Apr 2005, 9:47 a.m.,* *in response to message #35 by Palmer O. Hanson, Jr.*

Quote:

> I did not get your answer. Then I ran the determinant of the inverse of AM#1 and I got your answer.

You are perfectly right (except that you should have said "transposed" instead of "inverse".)

I think the main reason for the different results is the way, columns are rearranged by pivoting.

Re: More "Matrix Results on HP and TI Machines"

Message #40 Posted by ***Rodger Rosenbaum*** *on 27 Apr 2005, 10:40 a.m.,*
in response to message #1 by Valentin Albillo

N.B. Edited to include transpose results. Some results from three calculators for the determinant of Albillo's Matrix #1 and its transpose:

```
           AM#1         Transpose of AM#1
HP48S  .970960198039      .999403205613


HP48G  .999945522778      .999982262077


TI-86  .999646804338      1.00003051678
```

The HP48G is using 15 digit arithmetic for all internal matrix calculations, and it shows.

It's interesting that the transpose gives so much better results for the less accurate machines.

Edited: 27 Apr 2005, 11:18 a.m. after one or more responses were posted

But...15 digits?

Message #41 Posted by ***Gene*** *on 27 Apr 2005, 10:48 a.m.,*
in response to message #40 by Rodger Rosenbaum

Valentin's always saying (and I thought it true too), that the HP71B carries 15 digits too.

Why would the 48g series be so much more accurate?

Re: But...15 digits?

Message #42 Posted by ***Arnaud Amiel*** *on 27 Apr 2005, 11:05 a.m.,*
in response to message #41 by Gene

The 48 and 49 use for some algorithms some extended reals which have a 15 digits mantissa. I supose on the 71 the 15 digits are used for 12 digit mantissa, sign and exponent.

Arnaud

1234 to remove if it is not true...

Re: But...15 digits?

Message #43 Posted by ***Rodger Rosenbaum*** *on 27 Apr 2005, 11:11 a.m.,*
in response to message #41 by Gene

"Valentin's always saying (and I thought it true too), that the HP71B carries 15 digits too. Why would the 48g series be so much more accurate?"

Because the HP71 and the HP48S don't maintain those 15 digits for every partial result when doing matrix computations; they sometimes round things to 12 digits before proceeding with the rest of the computation. Paul McClellan reworked the matrix math in the HP48G to use 15 digits all the way through every internal matrix function. He also added all those extra nifty matrix functions such as SVD, LU, QR, etc.

Exact results with the HP ...

Message #44 Posted by ***Nenad (Croatia)*** *on 28 Apr 2005, 7:02 a.m.,*
in response to message #40 by Rodger Rosenbaum

Implementing one of my HP machines (physically even smaller than the HP71B) the following result is obtained in 11 seconds:

DET (Albillo Matrix #3)=1

without any tricks, programming, etc.

Anybody interested which HP and how :-)

Re: Exact results with the HP ...

Message #45 Posted by ***HrastProgrammer*** *on 29 Apr 2005, 12:42 a.m.,*
in response to message #44 by Nenad (Croatia)

Anybody interested which HP and how :-)

Jornada / Derive ?

Re: Exact results with the HP ...

Message #46 Posted by ***Nenad (Croatia)*** *on 29 Apr 2005, 4:01 a.m.,*
in response to message #45 by HrastProgrammer

You are right, Hrast (as always, if I might add...)

HP Jornada 720, with Derive 3.14 running on Pocket DOS

Just wanted to point out that a really powerful CAS can be implemented on a machine like HP J 720/728 without any problems. The only issue is that this is a very old DOS version of Derive intended for PC/XT, but it works as a beast.

Derive itself is extremely powerful (and a very small) program. I remember that a DOS version was able to fit on a single 5,25" floppy. In addition to this, implementation of Pocket DOS on a HP J preserves the capabilities of many old and wonderful programs written for good old DOS.

49g+ results for Valentin's #1 matrix

Message #47 Posted by ***Gene*** *on 29 Apr 2005, 9:33 a.m.,*
in response to message #40 by Rodger Rosenbaum

Of course, in exact mode, the 49g+ gives the exact results for Valentin's #1 matrix.

However, in APPROXIMATE mode, I get very interesting results. Here's what I did...

I took the exact matrix of integers, switched to approximate mode, then multiplied the integer matrix by 1. to convert all of the values to reals (they all now have "." after them in the matrix.

When I compute the determinant of this "real" matrix, I get:

1. --- Error = 0?

When I compute the determinant of the inverse of the matrix, I get:

0.998239455633

Error = 0.17605% or so.

The RowNorm of the product of the inverse with the original matrix comes up 1.03, error of 3%

The ColumnNorm of the product of the inverse with the original matrix comes up 1.014991, error of 1.4991%.

I don't believe the 49g+ has the "Frobenius Norm" built-in.

I'm really interested in the 1. determinant in approximate mode.

The inverse is not exact - and I can post that if someone wants to see it.

Re: 49g+ results for Valentin's #1 matrix

*Message #48 Posted by **whuy** on 29 Apr 2005, 9:57 a.m.,*
in response to message #47 by Gene

Gene - Set Flag -54 (Use Tiny Element) By default, it is Clear. One of the places where that makes a difference is in the computation of a determinant. An extra check is performed up front to see whether all entries are integers. If so, the result is rounded to the nearest integer. Of course, the inverse matrix is not integer any more, hence the real result.

Cheers, Werner

Updated...Re: 49g+ results for Valentin's #1 matrix

*Message #49 Posted by **Gene** on 29 Apr 2005, 10:28 a.m.,*
in response to message #48 by whuy

Thanks. With Flag 54 set, the determinant of Valentin's #1 matrix is now:

0.999945522778, an error of 0.0054478% or so.

Interestingly, the determinant of the INVERSE is:

0.999945522778, an error of 0.0054478% or so.

No joke. It is exactly the same.

Re: Updated...Re: 49g+ results for Valentin's #1 matrix

Message #50 Posted by ***whuy*** *on 29 Apr 2005, 10:50 a.m.,*
in response to message #49 by Gene

? No it's .998239455633 like you wrote in your previous post Werner

Re: Updated...Re: 49g+ results for Valentin's #1 matrix

Message #51 Posted by ***Gene*** *on 29 Apr 2005, 11:25 a.m.,*
in response to message #50 by whuy

You're right. I think I won't sign any financial documents today...must not be following things too closely. :-)

Re: 49g+ results for Valentin's #1 matrix

Message #52 Posted by ***Rodger Rosenbaum*** *on 29 Apr 2005, 9:12 p.m.,*
in response to message #47 by Gene

Gene said:"I don't believe the 49g+ has the "Frobenius Norm" built-in."

The Frobenius norm of a matrix is obtained with ABS().

Edited: 29 Apr 2005, 9:13 p.m.

Exact results with the HP-71B

Message #53 Posted by ***Valentin Albillo*** *on 28 Apr 2005, 5:03 a.m.,*
in response to message #1 by Valentin Albillo

Hi, all:

Though using **DET** to compute the Albillo's matrices determinants does result in quite poor results (due to their extreme ill-conditioning and inherent loss of precision caused by the many division operations the internal triangularization algorithm entails), this doesn't mean we can't obtain exact results in a 71B.

In fact, we just need another approach that doesn't involve divisions at any step, and one such approach is to use the "determinant expansion by minors" technique. This algorithm is of considerable theoretical interest, and can be recursively used to compute any determinant using just multiplications and additions, never a division in sight. As such it is *highly inefficient* for matrices above 4x4, say, but for just the purpose of showing how you can *in theory* compute those difficult determinants *exactly*, it will do.

Thanks to the powerful HP-71B's BASIC implementation, a simple version of the expansion by minors can be coded as a *recursive* subprogram in just 4 lines (with *arbitrary* line numbers):

```
  100 SUB XDET(A(,),D) @ DIM N,E,I,J,K @ N=UBND(A,1)
  110 IF N=2 THEN D=A(1,1)*A(2,2)-A(1,2)*A(2,1) @ END ELSE DIM B(N-1,N-1) @ D=0
  120 FOR K=1 TO N @ FOR I=2 TO N @ C=1 @ FOR J=1 TO N @ IF J#K THEN B(I-1,C)=A(I,J) @ C=C+1
  130 NEXT J @ NEXT I @ CALL XDET(B,E) @ D=D-(-1)^K*A(1,K)*E @ NEXT K
```

You must previously dimension the matrix (with OPTION BASE 1) and a real variable to return the value of the determinant, then populate the matrix and call the subprogram, like this:

Albillo Matrix 1:

From the keyboard do:

```
>OPTION BASE 1 @ DIM A(7,7) @ MAT INPUT A  [ENTER]


A(1,1)? 58,71,67,36,35,19,60,50,71,71,56,45,20,52,64,40,84,50,51,43,69  [ENTER]
A(4,1)? 31,28,41,54,31,18,33,45,23,46,38,50,43,50,41,10,28,17,33,41,46  [ENTER]
A(7,1)? 66,72,71,38,40,27,69  [ENTER]


>DET(A)  [ENTER]


 .97095056196  (inexact result with DET)


>CALL XDET(A,D) @ DISP D  [ENTER]


 1   (exact result with XDET)
```

Albillo Matrix 2:

```
>MAT INPUT A  [ENTER]


A(1,1)? 86,69,53,33,18,10,87,70,65,63,55,20,25,73,44,53,63,67,45,44,49 [ENTER]
A(4,1)? 32,27,50,84,36,12,33,28,41,39,64,51,45,33,11,20,13,17,19,33,15 [ENTER]
A(7,1)? 88,73,55,35,21,18,90  [ENTER]


>DET(A)  [ENTER]
```

```
 1.04720740631     (inexact result with DET)


>CALL XDET(A,D) @ DISP D  [ENTER]


 1     (exact result with XDET)
```

Albillo Matrix 3:

```
>MAT INPUT A  [ENTER]


A(1,1)? 60,53,61,65,50,37,64,49,78,67,67,24,10,50,31,52,59,60,32,19,33  [ENTER]
A(4,1)? 18,51,50,82,35,27,21,10,18,30,36,38,12,11,6,18,14,33,13,33,10   [ENTER]
A(7,1)? 61,57,63,72,51,45,66  [ENTER]


>DET(A)  [ENTER]


 .914527613074     (inexact result with DET)


>CALL XDET(A,D) @ DISP D  [ENTER]


 1     (exact result with XDET)
```

Please keep in mind that this version of **XDET** is just a proof-of-concept implementation, to show that it can be done, and as such has no error handling (the matrix must be at least 2x2, for instance), and is *extremely* inefficient, as it has to recursively call itself (N-1)!/2 times to compute an NxN determinant.

This means that for each of our three examples it calls itself (7-1)!/2 = 360 times, and it will take some 2 min. in Emu71 running on a 2.4 Ghz PC, and one or two hours on a real 71B. It also uses up large amounts of RAM as it deepens the recursion level.

However, it can be easily optimized in a number of ways. For instance, a preliminary search to locate rows/columns containing 0's, then expanding by minors along these rows/columns would mean avoiding a whole sub-branch of recursion for each 0 located. If no 0's are present, they can be created by suitably adding and subtracting rows.

Also, when N is 2, **XDET** refrains from recursing again to "compute" the required 1x1 determinants, but computes instead the 2x2 determinant directly (halving the number of recursive calls needed). This could be done for N=3 instead, which would result in greater time savings.

Best regards from V.