HP Forum Archive 17

[Return to Index | Top of Index]

Ok, non-extended BASIC programs for matrix work - where?

*Message #1 Posted by **Gene** on 30 Apr 2007, 9:27 p.m.*

Suppose you have a BASIC capable machine without any real math extensions but that has numeric arrays.

I thought there would be somewhere on the web programs to do matrix determinant and inverses for arbitrary M x M matrices, for example.

Can't find them.

Anyone have hints?

Won't do any good if it has specialized functions. Just basic arithmetic, loops, and matrices. :-)

Re: Ok, non-extended BASIC programs for matrix work - where?

*Message #2 Posted by **Namir** on 30 Apr 2007, 10:44 p.m.,*
in response to message #1 by Gene

Gene,

Numerical Recipes (Cambridge Press) the BASIC version is a very good book. I think the code for this book is in QuickBasic or QBasic, and is therefore a bit more structures.

McGraw Hill published in 1981 two volumes "Basic Scientific Subroutines" by Ruckdeschel. The code in this book is the old line-numbered BASIC. It should have subroutines for matrix manipulation.

If you can live with Visual Basic (version 3) code, then I wrote the book "Mathematical Algorithms in Visual Basic for Scientists & Engineers". Unfortunately you need VB 3 to read this companion disk!

I do have a VB Class (for Excel VBA and VB6) that does vector and matrix operations from scratch. The code is well commented and should be easy to follow.

Namir

Re: Ok, non-extended BASIC programs for matrix work - where?

*Message #3 Posted by **Valentin Albillo** on 30 Apr 2007, 10:49 p.m.,*
in response to message #1 by Gene

Hi, Gene:

Too late here (4:50 A.M) so I'm going to bed right now but you may want to have a look at these URLs

Matrix inversion

Matrix determinant

which I found instantly by googling searching for 'matrix inversion "next i" "next j"' and variants like that. You might try as well if the links I provided aren't what you need.

Hope it helps.

Best regards from V.

Thanks, Valentin...that looks like what I want!

*Message #4 Posted by **Gene** on 30 Apr 2007, 10:53 p.m.,*
in response to message #3 by Valentin Albillo

I was doing searches on "Basic determinant -visual" etc stuff.

Plan to use these with the Sharp Wizard 8000S I now have with the BASIC card. I'm curious to see how accurate it will be with all the array elements in double precision.

Re: Thanks, Valentin...that looks like what I want!

*Message #5 Posted by **GE** on 2 May 2007, 11:40 a.m.,*
in response to message #4 by Gene

Do you have the 32K or the 128K version of the Basic card ?
I think the 8000 + 128K Basic card is the best Sharp computer ever.

Re: Thanks, Valentin...that looks like what I want!

*Message #6 Posted by **Gene** on 2 May 2007, 12:49 p.m.,*
in response to message #5 by GE

I only have the 32K version...so far. :-)

I really like the double-precision mode when in Basic.

I did the usual 9 sin cos tan atan acos asin a minue ago and got 9.0000000000000000005 as the answer. NOT BAD.

I'm hoping to put in these BASIC matrix programs into it and run them in double precision mode to see how they do.

I'm hoping to use Valentin's "really mean" matrices he posted here and in Datafile to see how the determinant and inverse do.

That's why I'm hoping to find non-special function containing BASIC programs for matrix work to use on the 8000 model.

Re: Thanks, Valentin...that looks like what I want!

*Message #7 Posted by **Rodger Rosenbaum** on 2 May 2007, 3:07 p.m.,*
in response to message #6 by Gene

Hi, Gene,

You might also want to try the "Savage" benchmark.

http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv016.cgi?read=103356#103356

Savage in double precision on the Wizard 8000 w/Basic card

*Message #8 Posted by **Gene** on 2 May 2007, 4:05 p.m.,*
in response to message #7 by Rodger Rosenbaum

The value returned for A is 2499.9999999999999999

Again, not bad.

Re: Savage in double precision on the Wizard 8000 w/Basic card

*Message #9 Posted by **Rodger Rosenbaum** on 3 May 2007, 2:32 a.m.,*
in response to message #8 by Gene

> Quote:
>
> The value returned for A is 2499.9999999999999
>
> Again, not bad.

If the benchmark were carried with 23 digit arithmetic, and used round-to-even, the result *should* be 2499.9999999999999993.

Your actual result is very good, compared to what other calculators get on this benchmark.

Apparently, in a running program, intermediate results are kept to 23 digits, with the final result given to 20 digits. This is good, and not typical of other calculators, AFAIK.

However, on most calculators, when functions such as trig functions are calculated, the higher internal precision returns a result which is then rounded (or truncated) to the lesser displayed precision. Then the display precision value probably only has a few ULP's of error. The full internal precision function value, on the other hand, probably has hundreds of ULP's of error, and this may be visible in results from programs using those internal values. This doesn't seem to have happened here, and that makes me wonder if they might even use more than 23 internal digits sometimes.

The Durabrand 828 calculates that way, sort of anyway

*Message #10 Posted by **Palmer O. Hanson, Jr.** on 3 May 2007, 9:21 p.m.,*
in response to message #9 by Rodger Rosenbaum

Rodger:

You wrote in part

> Quote:
>
> ... Apparently, in a running program, intermediate results are kept to 23 digits, with the final result given to 20 digits. This is good, and not typical of other calculators, AFAIK.
>
> However, on most calculators, when functions such as trig functions are calculated, the higher internal precision returns a result which is then rounded (or truncated) to the lesser displayed precision. Then the display precision value probably

only has a few ULP's of error. The full internal precision function value, on the other hand, probably has hundreds of ULP's of error, and this may be visible in results from programs using those internal values. This doesn't seem to have happened here, and that makes me wonder if they might even use more than 23 internal digits sometimes.

A currently available machine which performs in that manner (sort of, at least) is the Durabrand 828. It returns pi, square roots, trigonometric functions and calculated results to the display register which are typcally correct to sixteen digits in the truncated sense. It displays results rounded to ten digits and stores results truncated to twelve digits in data registers. I tested the word length using a modification of the your test by calculating 100000000/7 - 14285714 and saw 0.28571428 in the display confirming that the word length is sixteen digits.

If you enter arcsin arccos arctan tan cos sin 9 and press EXE you get 8.999999998 in the display with 8.999999998078897 in the display register which is stored in a data register as 8.99999999807.

If you store each intermediate result,with a sequence such as sin 9 -> A EXE , cos A -> A EXE , etc., you effectively truncate each intermediate result from sixteen digits to twelve digits and get 9.000007164 in the display with 9.000007164103007 in the display register which is stored in a data register as 9.0000071641

Re: The Durabrand 828 calculates that way, sort of anyway

*Message #11 Posted by **Rodger Rosenbaum** on 4 May 2007, 12:14 a.m.,*
in response to message #10 by Palmer O. Hanson, Jr.

I've been thinking about calculator arithmetic tests that don't use the higher math functions, because those vary a lot among calculators, and the IEEE spec doesn't define their accuracy.

Here is a test that only checks the caalculator's basic arithmetic accuracy. It is based on the observation that if you reciprocate certain integers twice, you don't get the original integer back, and indeed you shouldn't on a finite precision BCD machine.

For example, type 6 1/x 1/x and you should see 5.99999999... on any calculator that does rounded arithmetic properly. If you see exactly 6 then the calculator isn't displaying all the digits in the result, or it's a "pleaser", like the HP30 (you should get 6.00000...0002 if it truncates properly).

So, the idea is to program a loop, apply 1/x twice for a range of integers, each time subtracting the result from the original integer that was reciprocated, and summing the absolute values of those small differences. This is another of my "what should we get" tests, because the result is determinate; there is a correct result, depending on how many digits are used, and the rounding mode.

Here's the program:

```
10 N=500 REM NUMBER OF ITERATIONS
20 S=0 REM INITIALIZE RUNNING SUM
30 FOR I = 1 TO N
40 S=S+ABS(I-1/(1/I))
50 NEXT I
60 PRINT S
```

The 1/x function is required by the IEEE standard to give properly rounded results, so there should be one correct result from the program from any calculator that complies with the standard.

The square root function is also specified by the IEEE, so line 40 in the program can be modified to:

```
40 S=S+ABS(I-SQRT(I)*SQRT(I))
```

and the program should give a specific result on a compliant machine.

This can be done on a non-programmable machine, with a lot of button pushing, but it's more convenient on a programmable machine!

The HP-71 is a good machine to test this on, because it is known to have IEEE compliant basic arithmetic, and its rounding modes can be changed.

The result using 1/x in line 40, with 500 iterations and OPTION ROUND NEAR set, is 6.803E-8.

With OPTION ROUND ZERO (truncating arithmetic), the result is 6.672E-8.

Both these results are exactly what they should be for a 12 digit machine.

I'm sure you'll want to try it on your own calculators.

A mini-challenge would be to verify the correct result for various numbers of digits used in the calculations, and with different rounding modes.

More on testing arithmetic

*Message #12 Posted by **Palmer O. Hanson, Jr.** on 4 May 2007, 4:21 a.m.,*
in response to message #11 by Rodger Rosenbaum

Rodger:

You wrote:

> Quote:
>
> I've been thinking about calculator arithmetic tests that don't use the higher math functions, because those vary a lot among calculators, and the IEEE spec doesn't define their accuracy. ..."

It will take me a while to digest this. In the meantime I offer a response to a suggestion that you made later in this thread:

> Quote:
>
> Try doing the sin cos tan atan acos asin test all on one command line with starting values of 6,7,8,10,11, and 12 degrees and see if the results are all as good as when starting with 9 degrees.

I did those values and some more with my Durabrand 828 with the following results:

```
N          N - f(N)


 5         3.153E-09
 6         3.233E-09
 7         2.730E-09
 8         6.021E-10
 9         1.921E-09
10         5.934E-10
11         3.791E-10
12         4.073E-10
13         6.103E-10
14         8.093E-10
15         1.433E-09
20         8.937E-10
25         5.447E-10
30         3.095E-10
60         6.158E-10
```

where all of the values in the second column are positive. I'll get some results for this test with aome other machines before moving on to some of your proposed tests.

Question: Does anyone know how someone settled on 9 as the test integer to be used for this test?

Palmer

Re: More on testing arithmetic

*Message #13 Posted by **Klaus** on 4 May 2007, 5:21 a.m.,*
in response to message #12 by Palmer O. Hanson, Jr.

Yes, because the test number should be a bit away from 0 (as some algorithms have special cases for args near zero), and it should be a single keystroke to enter the number (possibly through the blister pack in stores).

Edited: 4 May 2007, 5:22 a.m.

Re: Thanks, Valentin...that looks like what I want!

*Message #14 Posted by **Palmer O. Hanson, Jr.** on 4 May 2007, 10:05 p.m.,*
in response to message #7 by Rodger Rosenbaum

Message No. 15 in the thread on the Savage benchmark as posted by Gerson Barbosa on 2 December 2006 quoted the Wlodek article as

> Quote:
>
> ...on older HP calculators trig is accurate to almost 12 digits, but the policy of removing the last two digits, instead of leaving them hidden, can lead to less accuracy too. Newer HP calculators with a Saturn CPU work to 15 digits precision (and accuracy) internally, but round results to only 12 digits at the end of each calculation - the TI-74 works to 14 digits accuracy and precision, and leaves all 14 digits on the stack (with 4 hidden digits), so the final accuracy of the TI-74 result is 2500.0000291436.

I don't have all of my references here in North Carolina but my recollection is that the TI-74, and its predecessors the CC-40 and TI-99/4, operate with base 100. The result is that some calculations are done to 13 digits and some to 14. In this particular case when A < 10 the value of A will have been stored to 13 digits not 14. I don't know what that means for the end result.

Re: Thanks, Valentin...that looks like what I want!

*Message #15 Posted by **GE** on 7 May 2007, 3:59 a.m.,*
in response to message #14 by Palmer O. Hanson, Jr.

I think I recall that the TI74 and TI95 don't use CORDIC but polynomial approximations.

Re: Thanks, Valentin...that looks like what I want!

*Message #16 Posted by **Rodger Rosenbaum** on 2 May 2007, 3:59 p.m.,*

in response to message #6 by Gene

> Quote:
>
> I did the usual 9 sin cos tan atan acos asin a minue ago and got 9.0000000000000000005 as the answer. NOT BAD.

Gene, would you post all the intermediate results from this test?

Re: Thanks, Valentin...that looks like what I want!

*Message #17 Posted by **Gene** on 2 May 2007, 4:19 p.m.,*
in response to message #16 by Rodger Rosenbaum

```
The D stands for double precision.


sin 9 = 1.564344650402308690 1D-01
cos   = 9.999627274288502412D-01
tan   = 1.745499985548866079 1D-01
atan  = 9.999627274288502409D-01
acos  = 1.564344650402314465 3D-01
asin  = 9.000000000000335019


Funny, but when it is typed in as one command line, I stll get the 9.0000000000000000005 result.


Hmm...
```

The table of the forensic results...

*Message #18 Posted by **Gene** on 2 May 2007, 4:22 p.m.,*
in response to message #17 by Gene

Contains an entry for the Sharp E-500 showing both results I have obtained - one in a single calculation, the other step by step.

http://forensics.calcinfo.com/

Re: The table of the forensic results...

*Message #19 Posted by **Rodger Rosenbaum** on 2 May 2007, 6:15 p.m.,*
in response to message #18 by Gene

Note that footnote "m" on that site says of the two different values: "Unexplained different results from different contributors."

I guess we know the reason for the different results now!

Re: The table of the forensic results...

*Message #20 Posted by **Gene** on 2 May 2007, 6:19 p.m.,*
in response to message #19 by Rodger Rosenbaum

Yes, we do. Very odd. Wonder what it is doing in a chain calculation that is different from operating on a double-precision intermediate result?

Hmm..

Re: Thanks, Valentin...that looks like what I want!

*Message #21 Posted by **Rodger Rosenbaum** on 2 May 2007, 6:21 p.m.,*
in response to message #17 by Gene

What do you get if you type all on one command line:

```
1/7-.142857142857
```

Wizard 8000 results

*Message #22 Posted by **Gene** on 2 May 2007, 10:14 p.m.,*
in response to message #21 by Rodger Rosenbaum

0.0000000000001 in single precision mode and 1.4285714285D-13 in double precision mode.

Re: Wizard 8000 results

*Message #23 Posted by **Rodger Rosenbaum** on 3 May 2007, 2:19 a.m.,*
in response to message #22 by Gene

> Quote:
>
> 0.0000000000001 in single precision mode and 1.4285714285D-13 in double precision mode.

From this result we can see that the calc is using 23 digits internally in double precision mode, and displaying 20 digits.

But, it didn't round the result properly; it should have gotten 1.4285714286D-13, and the issue of "round-to-even" doesn't even arise. It just truncated the result.

Now about the forensic result:

```
sin 9 = 1.564344650402308690 1D-01
cos   = 9.999962727428850241 2D-01
tan   = 1.745499985548866079 1D-01
atan  = 9.999962727428850240 9D-01
```

there's a 1 ULP error here; the correct result of atan(1.7454999855488660791D-01) is 9.9999627274288502409846 3D-01, which should have rounded to 9.999962727428850241 0D-01. The result is simply truncated.

```
acos  = 1.564344650402314465 3D-01
```

The correct result of acos(9.999962727428850240 9D-01) is 1.564344650402314464 4D-01, an error of 9 ULP's.

```
asin  = 9.000000000000033501 9
```

This is really not too bad for trig functions for these arguments.

No doubt, when you evaluate this test all on one command line, the calculator is keeping intermediate results to 23 digits instead of just 20 digits as when you do it a step at a time, and so you get better accuracy.

If the test is carried out starting with 9 degrees, using 23 digit arithmetic with round-to-even, the result should be 8.9999999999999999467, rather than the 9.0000000000000000005 you actually got.

It's possible that the result starting from 9 degrees is especially good just by accident, due to compensating errors.

Carrying 24 digits, the result should be 9.0000000000000000015.

Carrying 25 digits, the result should be 9.0000000000000000001.

Try doing the sin cos tan atan acos asin test all on one command line with starting values of 6,7,8,10,11, and 12 degrees and see if the results are all as good as when starting with 9 degrees.

It appears that the calculator is truncating at least some of the time, when rounding would be better.

Result for the SHARP PC-1475

*Message #24 Posted by **Valentin Albillo** on 3 May 2007, 3:25 p.m.,*
in response to message #6 by Gene

Hi,

The result for the SHARP PC-1475 model is (DEGrees mode):

```
>ASN ACS ATN TAN COS SIN 9#


    9.0000000000000000005
```

Best regards from V.

Edited: 3 May 2007, 3:26 p.m.

Re: Result for the SHARP PC-1475

*Message #25 Posted by **Gene** on 3 May 2007, 4:18 p.m.,*
in response to message #24 by Valentin Albillo

Double precision is very nice. I'll try the simple inversion routine in double precision and post the results.

That over-rated trigonometric forensic!

*Message #26 Posted by **Karl Schneider** on 5 May 2007, 2:38 a.m.,*
in response to message #6 by Gene

Quote:

I did the usual

```
"9 sin cos tan atan acos asin"
```

a minute ago and got 9.0000000000000000005 as the answer.

All --

I know that this topic has been discussed before, but I'm still chagrined that this "calculator forensic" procedure remains so commonly accepted as a *de facto* standard for ostensibly assessing the quality of a calculator's general accuracy.

The prescribed mathematical calculation is

```
asin(acos(atan(tan(cos(sin(9 deg))))))
```

in degrees mode, for which the exact answer is 9. Usually, the answer returned by handheld calculators is not 9, but very close to it -- within 0.0000151% (8.99999864267) for Saturn-based models, 0.00464% (9.000417403) for Spice/HP-41/Voyager models, and 0.0453% (9.004076901) for the venerable HP-35. The HP-30S returnsobi the exact answer of 9 in this test, because it uses 80-bit floating-point math that provides about 24 internal digits, then rounds those results that are extremely close to integers, in order to help filter out the inexactitude of floating-point representation.

The computational sequence is mathematically "symmetrical", but not particularly useful in common practice. Why would one take the tangent of a sine of a cosine? If the units of 9 are degrees, what are the units of sine of 9 degrees for the cosine calculation? The angular unit is assumed to be degrees, but sine produces a plain ol' number without any units of measurement.

Much of the inaccuracy is introduced by roundoff error when results are internally multiplied by 180/pi (~= 57.29578) as the last step in the arc-functions to convert a radian-valued output to degrees. Smaller roundoff inaccuracies are also introduced during calculation of the trigonometric functions, in which each input is *divided* internally by 180/pi.

Obviously, the more *internal* digits a calculator has, the less susceptible it is to the unit-conversion errors. More *external* digits reduce roundoff errors in the final result of each calculation that is fed to the next one. The Saturn-based models have 12 external and 15 internal digits; the Spice/HP-41/Voyager models have 10 external and 13 internal. The HP-35 has 10 external and presumably fewer than 13 internal.

For comparison's sake, try the same computational sequence in radians mode, in order to eliminate the errors of converting between degrees and radians. The starting value will be 9*(pi/180) radians, which is mathematically identical, save for the small roundoff error of *that* computation.

Saturn-based models give 0.157079632681, for an error of about 1.273E-09%.

Spice/HP-41/Voyager models give 0.1570796319, for an error within 5.093E-07%.

Quite an improvement, I'd say -- four orders of magnitude smaller in each case. It is also noteworthy that the relative errors of the Saturn-based models and their immediate predecessors differ by roughly 2.5 orders of magnitude, in either degrees or radians mode. This is consistent with the two extra internal and external digits the Saturn-based models carry.

Radians mode is not available on the HP-35, but the HP-45 produces a result of 0.1570796336, for an error of 5.730E-07%. *(A "thanks" to Gerson Barbosa)*

Another difference of the HP-35 from the later models is that it does not seem to maintain ten *significant* digits to match its ten *displayed* digits. For example, the results of the three trigonometric functions are as follows:

```
Model     tan(cos(sin(9 deg)))
HP-42S       0.0174549998555
HP-11C       0.01745499985
HP-35        0.0174549998
```

The above is not intended to assert that the "forensic technique" has no value. The number of internal, external (displayable), and significant digits carried, as well as the algorithmic sophistication, will be reflected in the final result obtained. Still, it is instructive to know why a given result was obtained, and to use diagnostic/forensic calcuations that are valid in real-world applications.

-- KS

Edited: 6 May 2007, 12:04 a.m. after one or more responses were posted

Re: That over-rated trigonometric forensic!

*Message #27 Posted by **Gerson W. Barbosa** on 5 May 2007, 9:50 a.m.,*
in response to message #26 by Karl Schneider

Hello Karl,

> Quote:
>
> Why would one take the tangent of a sine of a cosine? If the units of 9 are degrees, what are the units of sine(9 deg)?

You're right! The only real life example involving the computation of two chained trigonometric functions I can think of now is cos(atan(Q/P)). I sometimes calculate power factors this way only as a convenience: 7 keystrokes versus 9 keystrokes on the HP-45, on which x^2 is not a shifted function, when compared to the equivalent P/sqrt(P^2 + Q^2), for P=4 and Q=3.

> Quote:
>
> No radians mode is available for the HP-35, but someone could try these with an HP-45.

0.1570796336 -> 5.730E-07%

Best regards,

Gerson.

Re: That over-rated trigonometric forensic!

*Message #28 Posted by **Gerson W. Barbosa** on 5 May 2007, 11:16 a.m.,*
in response to message #27 by Gerson W. Barbosa

> Quote:
>
> I sometimes calculate power factors this way only as a convenience: 7 keystrokes versus 9 keystrokes on the HP-45, on which x^2 is not a shifted function, when compared to the equivalent P/sqrt(P^2 + Q^2), for P=4 and Q=3.

Ok, on the HP-45 one could try also this 7-keystroke sequence:

```
3 ENTER 4 ->P 4 / 1/x
```

Trigonometrics for power factor

*Message #29 Posted by **Karl Schneider** on 5 May 2007, 2:51 p.m.,*
in response to message #27 by Gerson W. Barbosa

Hi, Gerson --

> Quote:
>
> The only real life example involving the computation of two chained trigonometric functions I can think of now is cos(atan(Q/P)). I sometimes calculate power factors this way...

Being in the same line of work, I sometimes use the inverse calculation: $Q/P = \tan(\cos^{-1}(pf))$. For a power factor of 0.95, the magnitude of reactive power Q is about one-third that of the active power P.

However, these calcuations are not the same as, for example, taking the cosine of a sine. Q/P is a dimensionless number; atan and acos produce an angle that has some unit of measure, which is the required input to tan and cos. Granted, the radian is *physically* dimensonless, but it is still a unit of measure.

-- KS

Re: That over-rated trigonometric forensic!

*Message #30 Posted by **Cameron Paine** on 5 May 2007, 11:48 a.m.,*
in response to message #26 by Karl Schneider

An aside from a person who only ever uses trigonometry (these days) for weekend carpentry projects...

I'm not qualified to comment on the usefulness of the chain calculation as a determinant of calculator accuracy or applicability. However I thought its original usefulness was precisely because its "inaccuracy" provided something akin to a fingerprint of the underlying calculation engine. That fingerprint allowed a curious person to guess at the genealogy of a particular calculator. In this context it says more about the possible relationships between different calculators than it does about their suitability for a particular application.

Cameron

Try this simple inversion routine, Gene

*Message #31 Posted by **Valentin Albillo** on 3 May 2007, 3:17 p.m.,*
in response to message #4 by Gene

Hi, Gene:

In addition to the code in the links I posted, try this extremely simple routine written and tested in plain vanilla BASIC:

```
>LIST


  1000 REM dimension and read matrix from sample data
  1010 RESTORE : READ N : DIM B(N,N)
  1020 FOR I=1 TO N : FOR J=1 TO N : READ B(I,J) : NEXT J : NEXT I
  1030 REM computing inverse
  1040 FOR P=1 TO N : K=B(P,P)
  1050 FOR I=1 TO N : FOR J=1 TO N : IF P<>J AND P<>I THEN B(I,J)=B(I,J)-B(I,P)*B(P,J)/K
  1060 NEXT J : NEXT I
  1070 FOR I=1 TO N : B(I,P)=B(I,P)/K : B(P,I)=-B(P,I)/K : NEXT I : B(P,P)=1/K : NEXT P
  1080 REM output
  1090 FOR I=1 TO N : FOR J=1 TO N : PRINT "B(";I;",";J;")=";B(I,J) : NEXT J : NEXT I : END
  1100 REM data for sample 4x4 matrix
  1110 DATA 4
  1120 DATA 6,-1,-3,1,-2,0,1,3,2,-1,0,1,-3,2,-1,0


  >RUN
```

```
    B( 1 , 1 )=-5
    B( 1 , 2 )=-6
    B( 1 , 3 )= 23
    B( 1 , 4 )= 9
    B( 2 , 1 )=-11
    B( 2 , 2 )=-13
    B( 2 , 3 )= 50
    B( 2 , 4 )= 20
    B( 3 , 1 )=-7
    B( 3 , 2 )=-8
    B( 3 , 3 )= 31
    B( 3 , 4 )= 12
    B( 4 , 1 )=-1
    B( 4 , 2 )=-1
    B( 4 , 3 )= 5
    B( 4 , 4 )= 2
>
```

which will generically work for NxN matrices and you can easily customize for your particular input or output requirements.

Because of its extreme simplicity (the inversion code is just 4 lines) this routine won't work if there are any 0 elements in the main diagonal or if they do appear there while inverting, something which could be easily corrected with a few extra lines of code, but I don't think that's necessary for testing purposes: simply use it "as is" for examples where no 0's get in the way, such as my Mean Matrices (tm), for example, or randomly generated matrices.

Should you absolutely need to run it with some sample matrix which has that problem, exchanging two rows or two columns usually solves the problem (unless the matrix is singular, of course) or you can try to add the necessary code to skip 0 pivots till the next iteration (just an extra line of code, really).

Best regards from V.

Great, that worked...now, about that AM#1 matrix...

*Message #32 Posted by **Gene** on 3 May 2007, 4:48 p.m.,*
in response to message #31 by Valentin Albillo

What I really want to do is test the Wizard 8000 w/the double precision BASIC card on those tough matrices you posted here and in Datafile V24N4P10.

This routine won't do that, of course, since those were 7x7.

And, alas, the BASIC on the card is not very enhanced. That's why I was looking for a vanilla BASIC determinant and inverse listing.

It really is a nice unit. Just want it to play with the big dogs!

Re: Great, that worked...now, about that AM#1 matrix...

*Message #33 Posted by **Valentin Albillo** on 3 May 2007, 7:16 p.m.,*
in response to message #32 by Gene

Hi, Gene:

> Gene posted:
>
> *"This routine won't do that, of course, since those were 7x7. And, alas, the BASIC on the card is not very enhanced. That's why I was looking for a vanilla BASIC determinant and inverse listing."*
>
> I beg your pardon but I don't understand. The routine above is as generic as it can be, you just need to change the DATA statements to:
>
> ```
> 1110 DATA 7
> 1120 DATA ... the 49 elements of your 7x7 matrix
> ```
>
> and there you are. As for vanilla BASIC, the above routine is absolutely plain vanilla, it uses nothing which isn't standard BASIC.
>
> If that DIM B(N,N) troubles your BASIC, just remove line 1110 DATA 4 or DATA 7 or whatever and the corresponding READ N at line 1010 and simply directly specify DIM B(4,4) or DIM B(7,7) instead.
>
> If this routine doesn't run in your particular card or machine, then I'm not sure it does actually support plain vanilla BASIC ... Good luck and

Best regards from V.

Re: Great, that worked...now, about that AM#1 matrix...

*Message #34 Posted by **Gene** on 3 May 2007, 8:05 p.m.,*
in response to message #33 by Valentin Albillo

My fault. Sometimes English is tough for us Americans. I had mis-read your description of the code. I thought it would ONLY work for 4x4 matrices. Why? I'm stupid sometimes. :-)

It ran perfectly for the smaller data statements. I'll try AM#1 later and we'll see what it can do.

As Homer Simpson would say "Doh!"

Re: Great, that worked...now, about that AM#1 matrix...

*Message #35 Posted by **Gene** on 3 May 2007, 8:33 p.m.,*
in response to message #34 by Gene

And, the version you provided works just fine on the Sharp Wizard 8000 w/the BASIC card. I had simply misread / not read carefully your (obviously now) clear instructions. :-)

Gene

Re: Great, that worked...now, about that AM#1 matrix...

*Message #36 Posted by **Valentin Albillo** on 3 May 2007, 9:05 p.m.,*
in response to message #35 by Gene

Great, glad to hear that.

Try AM#7 as well if you can. It's even more troublesome ...

Best regards from V.

AM#1 row 7 inverse results...

*Message #37 Posted by **Gene** on 3 May 2007, 10:34 p.m.,*
in response to message #36 by Valentin Albillo

```
(7,1) = -96360789.999826628746
(7,2) = -320207.99999942387722
(7,3) =  537451.99999903302359
(7,4) = -2323662.9999958192904
(7,5) =  11354926.999979570331
(7,6) = -30196487.999945670803
(7,7) = 96509953.999826360353
```

Re: AM#1 row 7 inverse results...

*Message #38 Posted by **Rodger Rosenbaum** on 4 May 2007, 3:38 a.m.,*
in response to message #37 by Gene

I'm surprised by your results. I would have expected better. You got 5 correct digits except for the 3rd item which has 4 (but almost 5) correct digits.

The condition number of the matrix is nearly 10^11, which means that you should lose 11 digits in your results. If you do 20 digit arithmetic, you should have gotten about 9 correct digits.

The HP50 gets 4 correct digits. It uses 15 digit arithmetic internally, so 15 - 11 (the exponent of the condition number) is 4, the number of correct digits we should expect.

Perhaps this is because the routine you used doesn't do full pivoting.

Prof. Kahan's routine should do better.

Re: AM#1 row 7 inverse results...

*Message #39 Posted by **Gene** on 4 May 2007, 4:17 p.m.,*
in response to message #38 by Rodger Rosenbaum

Hi Rodger!

Are you sure about correct digits?

Valentin's article in Datafile says that the last row of the inverse should be:

Value 7,1: -96360787 compared to my computed -96360789.999, so I have 7 correct digits here.

Value 7,2: -320208 compared to -320207.999999, which is truely 5 digits, but oh so close to more.

Value 7,3: 537452 compared to computed 537451.999999, again, 5 correct but ...

Value 7,4: -2323663 compared to computed -2323662.99999, which is 6 digits, but...

Value 7,5: 11354927 compared to computed 11354926.9999, which is 7 digits.

Value 7,6: -30196488 compared to computed -30196487.9999, which is 7 digits.

Value 7,7: 96509954 compared to computed 96509953.999, which is 7 digits.

That's an average correct of 6.28 digits correct across the 7 elements, not counting the instances where it was off by 0.001 or less. :-)

The actual values only average 7.28 real digits across the 7 elements.

The HP71B in the datafile article using Valentin's Method 1 only got an average of 1.43 digits correct for the last row of the inverse, for the record.

Aren't there more correct digits from the Sharp 8000 than you indicated? :-)

Re: AM#1 row 7 inverse results...

*Message #40 Posted by **Rodger Rosenbaum** on 5 May 2007, 4:45 p.m.,*
in response to message #39 by Gene

Sheesh!!

You're quite right!

I was using the 1st row to compare your results, not the seventh!

They're quite similar, if you ignore the signs, which I did.

AM#7 Row 7 inverse results

*Message #41 Posted by **Gene** on 4 May 2007, 8:19 a.m.,*
in response to message #37 by Gene

```
(7,1) = -133357.00188984163849
(7,2) =  952047.01349177555456
(7,3) =  3994038202.6008614399
(7,4) =  68560103.971588342704
(7,5) = -497464616.04973872238
(7,6) = -6667765.0944911542154
(7,7) = -3995675584.6240653375
```

Modifications to Valentin's program...

*Message #42 Posted by **Gene** on 4 May 2007, 8:21 a.m.,*
in response to message #41 by Gene

I made the matrix B and the constant K both double precision. Everything else was single precision. I think those are the values that were used in the computations.

Re: Modifications to Valentin's program...

*Message #43 Posted by **Rodger Rosenbaum** on 4 May 2007, 3:55 p.m.,*
in response to message #42 by Gene

> Quote:
>
> I made the matrix B and the constant K both double precision. Everything else was single precision. I think those are the values that were used in the computations.

This is much better. The condition number of the matrix is about 3E12, so you should expect somewhat less than 20 - 12 = 8 correct digits. Your results have 7 and 8 digits correct. Very nice.

[Return to Index | Top of Index]

Go back to the main exhibit hall