

Re: Matrix Multiplication

Source: <http://sci.tech-archive.net/Archive/sci.math.num-analysis/2008-01/msg00027.html>

- *From:* Evgenii Rudnyi <usenet@xxxxxxxxxx>
 - *Date:* Fri, 4 Jan 2008 11:20:18 -0800 (PST)
-

Hello Gordon,

Thank you for your comments.

If one were looking at this like a referee one would be questioning an author who seems to like to use a 30 year old version of Fortran. There are three revisions since 1977. Fortran 90 introduced a lot of matrix operations which are never mentioned.

I have written that I know Fortran 77 only and I have mentioned that the newer versions are available. If you are interested why, the answer is simple – I have switched to C++ long ago.

You can find my comments Fortran vs. C++ here

<http://evgenii.rudnyi.ru/doc/misc/fortran.txt>

but we do have to agree on this. Taste differs.

When an author goes out of the way to "think old" one wonders how much of the rest of the material has the same attributes.

I would appreciate if you will be more specific here.

Both a good implementation of an interpreter and of Fortran 90 are free to use the best available subroutine libraries for their purposes. The GNU successor to G77 may or may not have reached the maturity of using a highly optimized run time for matrix multiply.

Is Intel Fortran good? If you want it to, I will compile the code with it the next week.

By the way, g95 produces exactly the same results as g77 with this

Re: Matrix Multiplication

code. If you could donate the code in newer Fortran, please. It would be my pleasure to compile it and compare timing.

There is also the question of the intended audience for an introductory text and whether that audience should be treated to content which is so dated.

I am not sure I understand you. Do you mean that NumPy, C, C++ and ATLAS are outdated?

In a fast skim I noticed some comments on memory issues. I did not notice comments on the issues of size scaling when the matrix fits in the cache memory and when it does not. When it does not then is the issue of subscript order that may or may not be cache friendly. The advice that good subroutine packages deal with such issues is good. One would like some indication of the problem they address even if the full technical nature of the solution is not given. Keeping such a discussion both accurate and introductory is not easy.

Well, the goal was learning by doing. One first observe something and then starts thinking why it is. I guess that it is understandable that a matrix of 1000x1000 does not fit in the processor cache.

Best wishes,

Evgenii
<http://MatrixProgramming.com>

.