

## Re: Who uses clapack?

**Source:** <http://sci.tech-archive.net/Archive/sci.math.num-analysis/2004-12/0333.html>

---

**From:** Ron Shepard ([ron-shepard\\_at\\_NOSPAM.comcast.net](mailto:ron-shepard_at_NOSPAM.comcast.net))

**Date:** 12/11/04

Date: Fri, 10 Dec 2004 22:56:51 -0600

In article <41BA43C3.3040502@telus.net>,  
Jentje Goslinga <goslinga@telus.net> wrote:

> > *The issue here is scaling the elements of a vector by a scalar. I*  
> > *don't think it matters if that scalar is stored as a 64-bit or as an*  
> > *extended 80-bit result, the result, the vector of 64-bit values,*  
> > *will be the same. In other words, those 16-bits are not used (and*  
> > *cannot be used) in either case.*

>

> *Not quite so, there are four operations: (1) pulling each*  
> *element of the array through the processor, squaring and*  
> *adding the result to the accumulator, (2) taking the square*  
> *root of the accumulator, (3) dividing the result into unity*  
> *and (4) loading the scale factor and pulling each element*  
> *of the vector through the processor once more and scaling*  
> *it. Use of traditional BLAS interposes a mandatory memory save*  
> *between (2) and (3)*

First, the `dnrm2()` function is not required to store the internal result of the accumulation in 64-bit form. It is all internal to the `dnrm2()` function with no external storage required for the intermediate square root. Your `dnrm2()` function may or may not do the `sqrt()` in extended precision, but it is not forced to truncate by the design of the BLAS.

Even the reference fortran version of the `dnrm2()` function does not do a simple dot product to compute the vector norm, it accumulates into several scalar values based on the magnitude of the individual elements. The goal of this approach is to reduce the effect of roundoff error of the additions.

> *and another one between (3) and (4)*

Second, even if truncation to 64-bits occurs for the scale factor, the reciprocal is still accurate to 64-bit floating point precision. If the reciprocal is accurate to 64-bit precision, then the final scale operation will produce a full vector of values that are correct to full 64-bit precision.

sci.math.num-analysis: Re: Who uses clapack?

- > *assuming standard C program stack based parameter passing.*
- > *The rounding may not make much difference in this case but*
- > *it is not the same.*

Can you give a simple example to show the difference?

- > *More importantly, it is unnecessary: check your programs*
- > *with BLAS calls and you will find that a large percentage*
- > *of norm calculations are for the ultimate purpose of vector*
- > *unitization.*
- > *[The remainder are often for termination criteria where the*
- > *accuracy is less important.]*

Sure, one call instead of two would be simpler in these cases, but I don't think that your argument about loss of precision with two calls is correct.

- >> *This does not apply to computing dot products or vector norms. In*
- >> *these cases additions are involved rather than simple*
- >> *multiplications, so the extended precision can make a difference.*
- >
- > *No, for the inner product and vector norm it really does*
- > *not make any difference since any compiler will retain the*
- > *single scalar result in a FPU register.*

You say "no", but you meant "yes". We are both saying that extended precision while computing dot products or vector norms results in different values than if the results are computed with truncated precision.

- > *Trust me, I am an*
- > *Assembler programmer too.*

You can convince me by showing a simple example to prove your point.

At the time the level-1 BLAS were designed, many of the numerical analyst involved in the project were advocates of extended precision for dot product calculations. These guys were not idiots, they knew what they were doing.

\$.02 –Ron Shepard