

Re: Who uses clapack?

Source: <http://sci.tech-archive.net/Archive/sci.math.num-analysis/2004-12/0341.html>

From: Jentje Goslinga (goslinga_at_telus.net)

Date: 12/11/04

Date: Sat, 11 Dec 2004 06:27:37 GMT

Ron Shepard wrote:

> 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)

>

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

- >
- > *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.*
- >
- >
- >>*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?*

No, because I have not researched this particular one recently; I think the same consideration holds for the computation of the Householder vector.

- >>*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.*

Maybe it is not. My problem is that my thinking revolves around the Intel processor which I know very well and which does operate in extended precision for most operations such as multiply and add [sqrt and division are affected by the precision flag] and any normal store to memory truncates to normal double precision or single as the case may be.

- >>>*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.*

Re: Who uses clapack?

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

I meant it as I said but again with Intel it is hard to force the compiler not to compute and accumulate in extended precision unless one uses that obscure implementation defined switch which requires every intermediate result to be stored to memory.

>> *Trust me, I am an*

>> *Assembler programmer too.*

>

>

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

I have no time to do that at this point but may consider doing this and you may be right in the case of the dnm2/dscal combination or not, the difference is probably very small and platform dependent. I do think that having a "vector unify" function is an improvement.

I have tested a variety of the Lapack functions against functions of my own design and found that there is some minor room for improvement here and there of the precision. I have tested Schur 2x2 and SVD 2x2 and these can be improved somewhat. If I remember correctly, the Householder vector definitely benefits from containing the code within a single BLAS functions but again, I cannot precisely substantiate the effect on the whole computation at this point.

I do know that the net effect of the total of my measures is that the accuracy of a whole Eigensystem for example of a matrix with absurdly closely spaced eigenvalues (a few ULP apart) is about half the ULP than with Lapack with the D&C method but both are extremely accurate like 2/4 ULP for the eigenvalues and 8/16 ULP for QTQ-I. It is important to understand that accuracy is lost in only a few places and that the various orthogonal transformations when constructed to maximum precision are very accurate when applied.

> *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*

As I said, with Intel you get the extended precision with your dot product unless you do something special to thwart it.

In summary, the point of my article as regards to this post is that there may be some benefit in designing the BLAS more precisely to fit Lapack, not the other way around.

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

Thanks for your comments and regards,

Jentje Goslinga