Subject: Re: Accuracy problem
Posted by James Kuyper on Sat, 11 Mar 2006 11:56:40 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Juan Arrieta writes:
>
>> I am preparing a simple code to transform between cartesian vectors
>> and Keplerian elements (semimajor axis, inclination, eccentricity, and
>> so forth). This is a simple problem in astrodynamics.
>>
>> At some point in my code, I need to obtain unit vectors. For instance,
>> a line of the code is:
>>
>> U0 = ACOS((TRANSPOSE(NDVCT) # R) / (NORM(NDVCT) * NORM(R)))
>> % Program caused arithmetic error: Floating illegal operand
>> print,U0
>> NaN
...
> "fancy" programming. I don't know exactly what the problem
> is here, but clearly you need more precision. I'd start by
> doing everything in DOUBLE precision. You don't tell us
> what datatype NDVCT and R are but the NORM function is
> being done as a FLOAT, since you haven't set the DOUBLE
> keyword.
>
> I'd step back a couple of steps before you introduced us
> to the problem and make sure everything is done in double
> precision values. Then, after you have convinced yourself
> you've done everything humanly possible, I think you might
> be justified in confining the arguments to ACOS to the
> proper range:
>
> arg = ACOS(0.0 > expr < 1.0)

Using double precision floating point will help, but even with double precision, for some sufficiently small value epsilon, any expression whose mathematically exact result lies between 1.0-epsilon and 1.0 has a chance of evaluating numerically to a result >1.0, due to roundoff error. You HAVE to cap ACOS(-1.0 > expr < 1.0) to cope with these problems. I wouldn't recommend a lower limit of 0.0 unless that's actually indicated by the nature of your problem. Small negative arguments to ACOS don't pose the same kind of problem that arguments slightly greater than 1.0 do.