Subject: Re: need to speed up Runge-Kutta section
Posted by Kenneth P. Bowman on Thu, 05 May 2005 02:48:38 GMT
View Forum Message <> Reply to Message

In article <1115246704.219698.144050@o13g2000cwo.googlegroups.com>,
 "pdeyoung" <deyoung@hope.edu> wrote:

> The code below basically tracks particles through a magnetic field
> using 4th order RK. We are still running validation checks to find the
> typos but we know now that it is too slow. As currently written is
> does 1000 trajectories in about 0.5 second but ultimately we will need
> to generate about 10^6 for the project. The current speed is doable
> but I wonder if anyone can see a way to speed up the RK section (search
> for SYSTIME to find the beginning and end of the slow RK section). I
> know there is an RK4 built-in but worried that all the function calls
> would be even slower. Thankyou in advance for any suggestions. I am
> using IDL6.1
>
> Paul DeYoung
> deyoung@hope.edu

Do you want to do 1000 steps, or 1000 particles? If 1000 particles, you can vectorize your code. As it is, you have the whole RK section inside a FOR loop, so it is running as purely scalar, interpolated code. (Very slow in IDL.) You need to think IDL-style and do all these operations as vectors.

When you convert to vector operations, you will probably find that the bottleneck is the interpolation, as that tends to access memory in random order (which uses the cache inefficiently). The RK section should vectorize essentially completely.

You can do some minor optimizations in the RK code by converting divisions to multiplications and avoiding unnecessary type conversions. Some of your constants are integers, but should be floats.

After converting to vectors, you can insert a few SYSTIME calls to determine where the slow sections are.

Ken Bowman

Subject: Re: need to speed up Runge-Kutta section
Posted by Chris Lee on Thu, 05 May 2005 07:50:11 GMT
View Forum Message <> Reply to Message

In article <kpb-71F178.21483804052005@news.tamu.edu>, "Kenneth P. Bowman"
<kpb@null.com> wrote:

> In article <1115246704.219698.144050@o13g2000cwo.googlegroups.com>,
> "pdeyoung" <deyoung@hope.edu> wrote:
>
>> The code below basically tracks particles through a magnetic field
>> using 4th order RK. We are still running validation checks to find the
>> typos but we know now that it is too slow. As currently written is
>> does 1000 trajectories in about 0.5 second but ultimately we will need
>> to generate about 10^6 for the project. The current speed is doable
>> but I wonder if anyone can see a way to speed up the RK section (search
>> for SYSTIME to find the beginning and end of the slow RK section). I
>> know there is an RK4 built-in but worried that all the function calls
>> would be even slower. Thankyou in advance for any suggestions. I am
>> using IDL6.1
>> Paul DeYoung
>> deyoung@hope.edu

How sure are you that your RK4, written in IDL is faster than the RK4 built-in, written in C? The built-in IDL RK4 can be vectorized in the number of particles.

The RK4 takes a vector of (X,Y,Z) values for an arbitrary function F(X,Y,Z), if you use vectors of particles, you can get a speedup of about 30 times depending on the interpolation.

From my own 2D particle tracking (interpolate only works in 3D so you can't interpolate a 3D field in time easily), the efficiency peaks at about 1000 particle in parallel, with an efficiency of ~45. You can happily use more particles. On a Xeon 2.8Ghz with 2G of ram, the efficiency dropped to 38 at 1,000,000 particles.

You can also vectorize interpol(ate), by simply giving interpol(ate) a vector of particle locations. Basically, if your code is written in an IDL way, you should be able to decide how many particles in the input data, not in the function.

Scribblings in my log-book suggest you can get about 1,000,000 integrations / second.

Chris.

// efficiency = n * t_1 / t_n