Subject: Re: Looking for tetrahedra. Searching sorted lists.
Posted by James Kuyper on Tue, 16 Aug 2005 14:13:48 GMT
View Forum Message <> Reply to Message

cgguido wrote:
> Hi Karl,
>
> thanks for replying.
>
>> One thing worth pointing out is that nearly all mesh-related functions
>> in IDL are implemented in C because those algorithms call for a lot of
>> looping and other operations that are not efficient in IDL.
>
> Am not sure what you mean by mesh-related... my data comes from the 3D
> positions of ~5000 brownian particles diffusing around (each tagged
> with an id). I can determine whether particles are nearest neighbours
> two by two (which gives me the input matrix I mention in the original
> post) and now am looking for quadruplets of mutually nearest neighbours
> particles...

The links you've made between nearest neighbors define a mesh.

>> Since you've got an algorithm implemented in IDL and an apparent need to
>> run it repeatedly on lots of data that takes hours to process with IDL,
>> this may be a good example of something to translate to C and package up
>> as a DLM.
>
> That's a possibility I would have never thought of, thanks! Although I
> extensively use 'where' and 'uniq' and I don't know if they exist in C.
> (at the very least, they must be available as a library...)

Possibly, but nothing quite like 'where' or 'uniq' are available as
part of the C standard library. That's because you don't write C code
the same way you do IDL, mainly because of the huge performance penalty
for explicit loops in IDL. In most instances where IDL code uses where
or uniq, C code would use a loop, and therefore usually doesn't need to
store the complete list of indices; it just needs one index at a time:

IDL:
```
i = where(array eq 3)
; use 'i'
j = uniq(sorted_array)
; use 'j'
```

C:
```
int i;
for(i=0; i<n1; i++)
{
```

```
   if (array[i] == 3)
   {
      /* use i */
   }
}

int j;
for(j=0; j<n2; j++)
{
    if(j==n2-1 || sorted_array[j] != sorted_array[j+1])
    {
       /* use j */
    }
}
```

C code could store the complete list of matches, if needed, but then it would need to explicitly allocate the memory to store that list, and it would also need to explicitly deallocate it when the code is done with that list, which makes the user code more complicated than it would be in IDL. There's no free lunch here; those same issues apply to IDL, it's just handled internally by IDL itself. The cost of doing it internally is that's its done less efficiently, in some cases, than it could be if done under user control (by a sufficiently competent user!).
