Subject: To reduce an n^2 problem
Posted by simulana on Mon, 21 Sep 2015 21:55:35 GMT
View Forum Message <> Reply to Message

Let's say I have a gravitational potential problem in 2D array (or any Poisson problem), where the potential at each cell is determined by the sum of the contributions of all other cells, of the form m/r. This is naturally an n^2 problem, with a simple solution requiring a for loop over each cell and determining the distances from every other cell and then the sum of their contributions.

First question - has anyone ever done this using IDL? I know that multigrid solvers and FFT solvers have been written into many other languages to solve this problem. But, it seems like a direct solution could be obtained using IDL's array capabilities (a la distance_measure). What remains is serious indexing trouble, either with mapping the distance_measure results back to a 2D array (since it must be correlated to density), or if you skip distance_measure altogether and just attempt to make an (nx,ny,nx*ny) array for all of the distances.

Second question - even if you haven't done this in IDL, here is a thought experiment. Would using IDL arrays actually reduce this problem below n^2?

Thanks for your time.

Subject: Re: To reduce an n^2 problem
Posted by Jeremy Bailin on Tue, 22 Sep 2015 03:29:22 GMT
View Forum Message <> Reply to Message

On Monday, September 21, 2015 at 4:55:38 PM UTC-5, simu...@gmail.com wrote:
> Let's say I have a gravitational potential problem in 2D array (or any Poisson problem), where the potential at each cell is determined by the sum of the contributions of all other cells, of the form m/r. This is naturally an n^2 problem, with a simple solution requiring a for loop over each cell and determining the distances from every other cell and then the sum of their contributions.
>
> First question - has anyone ever done this using IDL? I know that multigrid solvers and FFT solvers have been written into many other languages to solve this problem. But, it seems like a direct solution could be obtained using IDL's array capabilities (a la distance_measure). What remains is serious indexing trouble, either with mapping the distance_measure results back to a 2D array (since it must be correlated to density), or if you skip distance_measure altogether and just attempt to make an (nx,ny,nx*ny) array for all of the distances.
>
> Second question - even if you haven't done this in IDL, here is a thought experiment. Would using IDL arrays actually reduce this problem below n^2?
>
> Thanks for your time.

Yes, I've done this. Not for any good reason, just for the hell of it. :)

No, it does not reduce below n^2 when you use direct summation as the algorithm, no matter what you stuff into IDL pre-made functions. You can try to reduce the time constant, but there's no

possible way to do n^2 calculations in less than O(n^2) time. The only way to reduce it below O(n^2) is to use a different algorithm that gives an approximation.

-Jeremy.
