
Subject: Re: Fast computation of pairwise array element differences

Posted by [Jeremy Bailin](#) on Sun, 04 Dec 2011 19:13:11 GMT

[View Forum Message](#) <> [Reply to Message](#)

On 12/4/11 11:28 AM, erik@bertram-kirn.de wrote:

> Hi folks,
>
> I have the following problem that I want to solve with IDL, but do not
> know how:
>
> I have a n^2 array (2D) with some arbitrary numerical data in it and a
> lengthscale L (let's say a certain number of cells smaller than n).
> What I want to do now is to compute all possible differences of
> elements in this 2D array that are L cells far away from each other,
> but *without using loops*!
>
> Of course, I could do the following (in pseudo code):
>
> -----
> diff = 0.d
>
> FOR x = 0, n do begin
> FOR y = 0, n do begin
> - Take array element (x, y)
> - Look for all cells that are L cells away from cell (x, y)
> (e.g. in a ring around (x, y) with radius of L cells)
> - compute all differences from (x, y) to the other cells in a
> radius of L
> - Summ differences up in a variable diff
> - continue with next array element and compute next differences
> and so on...
> ENDFOR
> ENDFOR
> -----
>
> This is an easy mathematical operation. Nevertheless, for an array of
> 500 x 500 it takes about hours to step through all the looping
> processes and to calculate the sum of really all possible differences
> on a lengthscale of L cells...
>
> Does someone have another idea how to avoid the loops in this case and
> to compute all possible differences in such an array very easily?
>
> Thank you very much!
>
> Kind regards,
>
> Erik

It sounds like you're trying to do a convolution with a kernel that looks something like (schematic):

```
0 0 0 1 1 1 0 0 0
0 0 1 0 0 0 1 0 0
0 1 0 0 0 0 0 1 0
0 1 0 0 -16 0 0 1 0
0 1 0 0 0 0 0 1 0
0 0 1 0 0 0 1 0 0
0 0 0 1 1 1 0 0 0
```

(for a particular L of 3, say). If so, try looking at CONVOL, and also convolution via FFT.

-Jeremy.
