Subject: computation time for convolution
Posted by fra on Wed, 09 Jul 2014 08:12:30 GMT
View Forum Message <> Reply to Message

I am a little puzzled about the computation time required by different convolution routines. I need to compute several times the convolution of large arrays and I always used the convolve routine of the astrolib. Since I need to speed up the processing I compared the computation time for array of different size (but using sizes power of 2, which should be the best case for FFT) convolved with different routines. The best result (by far) is obtained with the function convol of the IDL standard library, the worst is convol_fft and convolve is somewhat in the middle. This does not make sense to me, I was sure that the FFT approach is the fastest. What am I missing or doing wrong?

These are the results:

```
           4x           4
convolve:     0.038000107
convol:      0.00000000
convol_fft:   0.00099992752
           8x           8
convolve:      0.00000000
convol:      0.00000000
convol_fft:      0.00000000
          16x          16
convolve:      0.00000000
convol:      0.00000000
convol_fft:      0.00000000
          32x          32
convolve:      0.00000000
convol:      0.00000000
convol_fft:    0.0010001659
          64x          64
convolve:      0.00000000
convol:      0.00000000
convol_fft:    0.0019998550
         128x         128
convolve:   0.00099992752
convol:    0.0010001659
convol_fft:    0.0079998970
         256x         256
convolve:    0.0080001354
convol:    0.0019998550
convol_fft:     0.035000086
         512x         512
convolve:     0.036000013
convol:    0.0069999695
convol_fft:      0.28600001
        1024x        1024
convolve:      0.25300002
```

```
convol:     0.026999950
convol_fft:       1.4849999
        2048x        2048
convolve:       1.6410000
convol:      0.11600018
convol_fft:       6.6910000
        4096x        4096
convolve:       7.4190001
convol:      0.43299985
convol_fft:       26.736000
```

and this is the code I used for this test:

```
for i=2,12 do begin
  a=fltarr(2l^i,2l^i)
  b=a
  time0=systime(1)
  c=convolve(a,b)
  time1=systime(1)
  c=convol(a,b)
  time2=systime(1)
  c=convol_fft(a,b)
  time3=systime(1)
  print,2l^i,'x',2l^i
  print,'convolve:', time1-time0
  print,'convol:', time2-time1
  print,'convol_fft:', time3-time2
endfor
```

Subject: Re: computation time for convolution
Posted by wlandsman on Wed, 09 Jul 2014 16:48:26 GMT

View Forum Message <> Reply to Message

It looks like the FFT calculations are giving double precision output. I believe this is a bug -- they should not give double precision output when all inputs are floating point. I've updated http://idlastro.gsfc.nasa.gov/ftp/pro/image/convolve.pro so that it no longer does this. It is curious that the standard IDL routine CONVOL_FFT seems to have the same problem.

Both FFT routines will run much faster by using the /NO_PAD keyword, though this can give spurious results near the edges. Conversely, the standard convolution CONVOL() seems to run much slower when using one of the EDGE_* keywords.

Having said this, yeah I don't understand why IDL CONVOL() is so fast -- or conversely why IDL FFT() is so slow. --Wayne

On Wednesday, July 9, 2014 4:12:30 AM UTC-4, fraro...@yahoo.it wrote:

> I am a little puzzled about the computation time required by different convolution routines. I need to compute several times the convolution of large arrays and I always used the convolve routine of the astrolib. Since I need to speed up the processing I compared the computation time for array of different size (but using sizes power of 2, which should be the best case for FFT) convolved with different routines. The best result (by far) is obtained with the function convol of the IDL standard library, the worst is convol_fft and convolve is somewhat in the middle. This does not make sense to me, I was sure that the FFT approach is the fastest. What am I missing or doing wrong?

Subject: Re: computation time for convolution
Posted by chris_torrence@NOSPAM on Thu, 10 Jul 2014 00:59:29 GMT
View Forum Message <> Reply to Message

On Wednesday, July 9, 2014 10:48:26 AM UTC-6, wlandsman wrote:
>
> Having said this, yeah I don't understand why IDL CONVOL() is so fast
>

Because David Stern was a god.

Subject: Re: computation time for convolution
Posted by Lajos Foldy on Thu, 10 Jul 2014 08:47:44 GMT
View Forum Message <> Reply to Message

On Wednesday, July 9, 2014 10:12:30 AM UTC+2, fraro...@yahoo.it wrote:
> I am a little puzzled about the computation time required by different convolution routines. I need to compute several times the convolution of large arrays and I always used the convolve routine of the astrolib. Since I need to speed up the processing I compared the computation time for array of different size (but using sizes power of 2, which should be the best case for FFT) convolved with different routines. The best result (by far) is obtained with the function convol of the IDL standard library, the worst is convol_fft and convolve is somewhat in the middle. This does not make sense to me, I was sure that the FFT approach is the fastest. What am I missing or doing wrong?
>
>
>
> These are the results:
>
>
>
```
>           4x          4
>
> convolve:     0.038000107
>
> convol:      0.00000000
```

```
>
> convol_fft:   0.00099992752
>
>           8x          8
>
> convolve:      0.00000000
>
> convol:      0.00000000
>
> convol_fft:      0.00000000
>
>          16x         16
>
> convolve:      0.00000000
>
> convol:      0.00000000
>
> convol_fft:      0.00000000
>
>          32x         32
>
> convolve:      0.00000000
>
> convol:      0.00000000
>
> convol_fft:    0.0010001659
>
>          64x         64
>
> convolve:      0.00000000
>
> convol:      0.00000000
>
> convol_fft:    0.0019998550
>
>         128x        128
>
> convolve:   0.00099992752
>
> convol:    0.0010001659
>
> convol_fft:    0.0079998970
>
>         256x        256
>
> convolve:    0.0080001354
>
> convol:    0.0019998550
```

```
>
> convol_fft:     0.035000086
>
>         512x        512
>
> convolve:     0.036000013
>
> convol:    0.0069999695
>
> convol_fft:      0.28600001
>
>        1024x       1024
>
> convolve:      0.25300002
>
> convol:     0.026999950
>
> convol_fft:       1.4849999
>
>        2048x       2048
>
> convolve:       1.6410000
>
> convol:      0.11600018
>
> convol_fft:       6.6910000
>
>        4096x       4096
>
> convolve:       7.4190001
>
> convol:      0.43299985
>
> convol_fft:       26.736000
>
>
>
> and this is the code I used for this test:
>
>
>
> for i=2,12 do begin
>
>   a=fltarr(2l^i,2l^i)
>
>   b=a
>
>   time0=systime(1)
```

```
>
>    c=convolve(a,b)
>
>    time1=systime(1)
>
>    c=convol(a,b)
>
>    time2=systime(1)
>
>    c=convol_fft(a,b)
>
>    time3=systime(1)
>
>    print,2l^i,'x',2l^i
>
>    print,'convolve:', time1-time0
>
>    print,'convol:', time2-time1
>
>    print,'convol_fft:', time3-time2
>
>  endfor
```

You are not testing convol, you are testing a very special case (convol(a,a) calculates the sum in a single position, all other array elements are set to zero).

You should use a more realistic kernel, eg b=dist(2l^(i-1)). With this I got:

```
      1024x       1024
convolve:      2.4647841
convol:      35.345544
convol_fft:      2.8855970
```

regards,
Lajos

Subject: Re: computation time for convolution
Posted by wlandsman on Thu, 10 Jul 2014 12:16:56 GMT

View Forum Message <> Reply to Message

I knew that the OP was showing a special case but I was still finding CONVOL() to be faster than using an FFT.   I now realize that the FFT compute time is independent of the relative size of the kernel and the image (since the kernel must be converted to the same size as the image prior to the FFT multiplication).   So instead of varying the size of the image, I kept the image size fixed at 1024x1024 and varied the size of the kernel.   The speed of CONVOL varies with the number of points in the kernel, while the FFT speed is almost indecent of the kernel size.   For a small kernel (my usual case) CONVOL remains much faster.   --Wayne

```
Kernel size:          16
convolve:      1.2780330
convol:     0.053172827
convol_fft:      1.2723532

Kernel size:          32
convolve:      1.2661600
convol:      0.20477700
convol_fft:      1.2720819

Kernel size:          64
convolve:      1.2892501
convol:      0.80016303
convol_fft:      1.2787650

Kernel size:         128
convolve:      1.2785149
convol:       2.8997500
convol_fft:      1.2978389

Kernel size:         256
convolve:      1.2797129
convol:       9.5446420
convol_fft:      1.2797601

Kernel size:         512
convolve:      1.3157580
convol:       22.437527
convol_fft:      1.3163621
```

Code:

```
pro test
a = randomn(seed,1024,1024)
for i=4,9 do begin
  b = dist(2^i)
  time0=systime(1)
  c1=convolve(a,b)
  time1=systime(1)
  c2=convol(a,b)
  time2=systime(1)
  c3=convolve(a,b)
  time3=systime(1)
  print,'Kernel size: ',2l^i
  print,'convolve:', time1-time0
  print,'convol:', time2-time1
  print,'convol_fft:', time3-time2
```

endfor
end

>
> You are not testing convol, you are testing a very special case (convol(a,a) calculates the sum in a single position, all other array elements are set to zero).
>
>
>
> You should use a more realistic kernel, eg b=dist(2l^(i-1)). With this I got:
>
>
>

```
>         1024x       1024
>
> convolve:       2.4647841
>
> convol:       35.345544
>
> convol_fft:       2.8855970
>
>
>
```

> regards,
>
> Lajos

Subject: Re: computation time for convolution
Posted by fra on Thu, 10 Jul 2014 21:19:03 GMT
View Forum Message <> Reply to Message

thanks a lot!
now it makes much more sense
I just assumed that the computation time for convol does not depend on the contents of the input arrays, as for the FFT-based algorithms

The particular problem for which I started to compare the performance of the convolution algorithms was about convolving a very large array with a small kernel, so it seems that convol is ok, but I don't have to change all the other pieces of my code where I use convolve for couples of large (but not too large) arrays with similar size. Thanks for the improvement to convolve !

Subject: Re: computation time for convolution

Posted by wlandsman on Fri, 11 Jul 2014 02:17:33 GMT
View Forum Message <> Reply to Message

Here's an interesting post addressing the question -- for what size kernel is the Fourier transform more efficient that direct convolution?

 http://programmers.stackexchange.com/questions/171757/comput
ational-complexity-of-correlation-in-time-vs-multiplication- in-frequency-s

Given a kernel of width K and an image of width W, the Fourier transform method is more efficient than direct convolution when

K > sqrt(8*alog(W)/alog(2))

The author makes a lot of approximations.   F0r example, I do think that David Stern was able to be more efficient than (K^2)*(W^2)in his implementation of direct convolution.   For a 1024 x 1024 image the above formula says that Fourier transforms are preferred when K > 9, whereas my simple experiments suggest that K ~ 64 is more appropriate for IDL.   --Wayne