Subject: Re: Approximate convolution - for loop problem
Posted by David Fanning on Sun, 21 Dec 2008 18:03:05 GMT
View Forum Message <> Reply to Message

samuel.leach@gmail.com writes:

> Hello everyone, I'm trying to execute a 1-d convolution of an array,
> signal.
>
> Using an analytic approximation, obtaining the convolved bolometer
> signal, bolo_signal, at time step ii, is given by the following:
>
> nsamp=n_elements(signal)
> const1 = exp(-tsamp/taubolo)
> const2 = 1.-const1
>
>
> bolo_signal = const2*signal
> for ii= 1L,nsamp-1L do begin
> bolo_signal[ii] += const1*bolo_signal[ii-1]
> endfor
>
> where tsamp and taubolo are scalars. Is there any way to avoid the for
> loop in this case? The hope is to speed up the execution.

I think this gives you the same results:

```
bolo_signal += const1 * shift(bolo_signal,-1)
```

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Approximate convolution - for loop problem
Posted by Sam on Sun, 21 Dec 2008 20:01:39 GMT
View Forum Message <> Reply to Message

Hi David, unfortunately shift() does not do the business for me, as
these two examples below show. So I'm still a bit stumped here.

```
; Array operation I'm trying to execute.
a=[1.,2.,3.,4.]
```

```
for ii=1,3 do a[ii] += 0.5*a[ii-1]
print,a
 1.00000      2.50000      4.25000      6.12500

; Attempt to perform this operation with shift()
a=[1.,2.,3.,4.]
a += 0.5*shift(a,-1)
print,a
 2.00000      3.50000      5.00000      4.50000
```

On Dec 21, 7:03 pm, David Fanning <n...@dfanning.com> wrote:
> samuel.le...@gmail.com writes:
>> Hello everyone, I'm trying to execute a 1-d convolution of an array,
>> signal.
>
>> Using an analytic approximation, obtaining the convolved bolometer
>> signal, bolo_signal, at time step ii, is given by the following:
>
>> nsamp=n_elements(signal)
>> const1 = exp(-tsamp/taubolo)
>> const2 = 1.-const1
>
>> bolo_signal = const2*signal
>> for ii= 1L,nsamp-1L do begin
>>   bolo_signal[ii] += const1*bolo_signal[ii-1]
>> endfor
>
>> where tsamp and taubolo are scalars. Is there any way to avoid the for
>> loop in this case? The hope is to speed up the execution.
>
> I think this gives you the same results:
>
>   bolo_signal += const1 * shift(bolo_signal,-1)
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Approximate convolution - for loop problem
Posted by David Fanning on Sun, 21 Dec 2008 20:44:02 GMT
View Forum Message <> Reply to Message

Sam writes:

> Hi David, unfortunately shift() does not do the business for me, as
> these two examples below show. So I'm still a bit stumped here.
>
> ; Array operation I'm trying to execute.
> a=3D[1.,2.,3.,4.]
> for ii=3D1,3 do a[ii] +=3D 0.5*a[ii-1]
> print,a
> 1.00000 2.50000 4.25000 6.12500
>
> ; Attempt to perform this operation with shift()
> a=3D[1.,2.,3.,4.]
> a +=3D 0.5*shift(a,-1)
> print,a
> 2.00000 3.50000 5.00000 4.50000

Humm. Yes, I see what you mean. I think you might
be out of luck without a loop here, since the *next*
calculation depends on the *previous*. I don't see how
it can be done without a loop.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Approximate convolution - for loop problem
Posted by Wout De Nolf on Mon, 22 Dec 2008 09:31:01 GMT
View Forum Message <> Reply to Message

On Sun, 21 Dec 2008 09:32:40 -0800 (PST), samuel.leach@gmail.com
wrote:

> nsamp=n_elements(signal)
> const1 = exp(-tsamp/taubolo)
> const2 = 1.-const1
>

> bolo_signal = const2*signal
> for ii= 1L,nsamp-1L do begin
>   bolo_signal[ii] += const1*bolo_signal[ii-1]
> endfor
>
> where tsamp and taubolo are scalars. Is there any way to avoid the for
> loop in this case? The hope is to speed up the execution.

This is without loop. Not sure it's faster though.

```
a=[1.,2.,3.,4.]
c=0.5
n=n_elements(a)

 a=reform([reform(rebin([a,0],n+1,n-1,/sample),n*n-1),a[0]],n ,n)
i=REBIN(LINDGEN(n), n, n)
j=REBIN(TRANSPOSE(LINDGEN(n)), n, n)
a*= i ge j

c=rebin(c^reverse(indgen(n)),n,n,/sample)
a=reverse(total(a*c,1))
print,a
```

Subject: Re: Approximate convolution - for loop problem
Posted by Sam on Mon, 22 Dec 2008 22:19:39 GMT
View Forum Message <> Reply to Message

It seems more memory demanding and I can not allocate the memory for
it.
Well, thank you for the suggestions. Best Regards, Sam.

On Dec 22, 10:31 am, Wox <s...@nomail.com> wrote:
> On Sun, 21 Dec 2008 09:32:40 -0800 (PST), samuel.le...@gmail.com
> wrote:
>
>> nsamp=n_elements(signal)
>> const1 = exp(-tsamp/taubolo)
>> const2 = 1.-const1
>
>> bolo_signal = const2*signal
>> for ii= 1L,nsamp-1L do begin
>>   bolo_signal[ii] += const1*bolo_signal[ii-1]
>> endfor
>
>> where tsamp and taubolo are scalars. Is there any way to avoid the for
>> loop in this case? The hope is to speed up the execution.

>
> This is without loop. Not sure it's faster though.
>
> a=[1.,2.,3.,4.]
> c=0.5
> n=n_elements(a)
>
> a=reform([reform(rebin([a,0],n+1,n-1,/sample),n*n-1),a[0]],n ,n)
> i=REBIN(LINDGEN(n), n, n)
> j=REBIN(TRANSPOSE(LINDGEN(n)), n, n)
> a*= i ge j
>
> c=rebin(c^reverse(indgen(n)),n,n,/sample)
> a=reverse(total(a*c,1))
> print,a

Subject: Re: Approximate convolution - for loop problem
Posted by andrews32940 on Tue, 23 Dec 2008 19:46:11 GMT
View Forum Message <> Reply to Message

This doesn't get rid of the FOR loop but it should be faster with
minimal memory overhead:

```
  a = [1.,2.,3.,4.]
  Print, 'Start a = ', a
  na = N_Elements(a)
  const1 = 0.5
  coef = Make_Array(na, /DOUBLE, VALUE=const1)
  coef[0] = 1.0D ; First index is a special case
  scale = Reverse(Product(coef, /CUMULATIVE))
  for ii = na-1L, 0L, -1L DO a[ii] = Total(a[0:ii]*scale[na-1-
ii:na-1])
  Print, 'End a = ', a

Start a =       1.00000      2.00000      3.00000      4.00000
End a =       1.00000      2.50000      4.25000      6.12500
```

On Dec 21, 3:01 pm, Sam <samuel.le...@gmail.com> wrote:
> Hi David, unfortunately shift() does not do the business for me, as
> these two examples below show. So I'm still a bit stumped here.
>
> ; Array operation I'm trying to execute.
> a=[1.,2.,3.,4.]
> for ii=1,3 do a[ii] += 0.5*a[ii-1]
> print,a
> 1.00000 2.50000 4.25000 6.12500
>

> ; Attempt to perform this operation with shift()
> a=[1.,2.,3.,4.]
> a += 0.5*shift(a,-1)
> print,a
> 2.00000 3.50000 5.00000 4.50000
>
> On Dec 21, 7:03 pm, David Fanning <n...@dfanning.com> wrote:
>
>
>
>> samuel.le...@gmail.com writes:
>>> Hello everyone, I'm trying to execute a 1-d convolution of an array,
>>> signal.
>
>>> Using an analytic approximation, obtaining the convolved bolometer
>>> signal, bolo_signal, at time step ii, is given by the following:
>
>>> nsamp=n_elements(signal)
>>> const1 = exp(-tsamp/taubolo)
>>> const2 = 1.-const1
>
>>> bolo_signal = const2*signal
>>> for ii= 1L,nsamp-1L do begin
>>> bolo_signal[ii] += const1*bolo_signal[ii-1]
>>> endfor
>
>>> where tsamp and taubolo are scalars. Is there any way to avoid the for
>>> loop in this case? The hope is to speed up the execution.
>
>> I think this gives you the same results:
>
>> bolo_signal += const1 * shift(bolo_signal,-1)
>
>> Cheers,
>
>> David
>> --
>> David Fanning, Ph.D.
>> Fanning Software Consulting, Inc.
>> Coyote's Guide to IDL Programming:http://www.dfanning.com/
>> Sepore ma de ni thui. ("Perhaps thou speakest truth.")- Hide quoted text -
>
> - Show quoted text -

Subject: Re: Approximate convolution - for loop problem
Posted by Jeremy Bailin on Wed, 24 Dec 2008 02:27:07 GMT

On Dec 21, 3:01 pm, Sam <samuel.le...@gmail.com> wrote:
> Hi David, unfortunately shift() does not do the business for me, as
> these two examples below show. So I'm still a bit stumped here.
>
> ; Array operation I'm trying to execute.
> a=[1.,2.,3.,4.]
> for ii=1,3 do a[ii] += 0.5*a[ii-1]
> print,a
> 1.00000 2.50000 4.25000 6.12500
>
> ; Attempt to perform this operation with shift()
> a=[1.,2.,3.,4.]
> a += 0.5*shift(a,-1)
> print,a
> 2.00000 3.50000 5.00000 4.50000
>
> On Dec 21, 7:03 pm, David Fanning <n...@dfanning.com> wrote:
>
>> samuel.le...@gmail.com writes:
>>> Hello everyone, I'm trying to execute a 1-d convolution of an array,
>>> signal.
>
>>> Using an analytic approximation, obtaining the convolved bolometer
>>> signal, bolo_signal, at time step ii, is given by the following:
>
>>> nsamp=n_elements(signal)
>>> const1 = exp(-tsamp/taubolo)
>>> const2 = 1.-const1
>
>>> bolo_signal = const2*signal
>>> for ii= 1L,nsamp-1L do begin
>>> bolo_signal[ii] += const1*bolo_signal[ii-1]
>>> endfor
>
>>> where tsamp and taubolo are scalars. Is there any way to avoid the for
>>> loop in this case? The hope is to speed up the execution.
>
>> I think this gives you the same results:
>
>> bolo_signal += const1 * shift(bolo_signal,-1)
>
>> Cheers,
>
>> David
>> --
>> David Fanning, Ph.D.
>> Fanning Software Consulting, Inc.

>> Coyote's Guide to IDL Programming:http://www.dfanning.com/
>> Sepore ma de ni thui. ("Perhaps thou speakest truth.")
>
>

How about this:

```
a = [1.,2.,3.,4.]
n = n_elements(a)
c = 0.5^reverse(indgen(n))
new_a = total(a*c, /cumulative) / c
```

-Jeremy.
