Subject: Matrix operations with IDL: Avoiding for loops
Posted by vince33600 on Wed, 30 Dec 2015 00:57:26 GMT
View Forum Message <> Reply to Message

Dear all,

I was trying to improve the performance of some pieces of code that are taking forever to run. Basically, I'm trying to multiply a set of n matrix (3x3) by a set of n vectors (3x1) without using any for loops. The results of these operations should give me a set of n vectors (3x1).

Let's take a simplified example where n=2. Therefore, I have 2 matrixes (let's call them a and b) that needs to be multiplied to 2 vector (let's call them u and v).

I figured out that the operation could be done by reshaping (using rebin and reform for instance) the matrixes into a bigger array (let's call it M) whose diagonal elements are the a and b matrixes, so that:

M = | a 0 |
    | 0 b |

where a and b are the 3x3 matrixes, and by reshaping the n vectors into in single vector (called I), so that:

I = | u |
    | v |

Then, the results would be:
R = M.I

Finally, the n vectors would be obtained by reshaping the R vector into n (3x1) vector.

Coming for fortran, I initially coded that by decomposing every single matrix multiplication in a for loop. I then tried to apply the above solution, but it seems a real stretch for me to do it without any loops.

I was thinking that someone already might have faced that problem.

Thanks for your help!
Vincent

Subject: Re: Matrix operations with IDL: Avoiding for loops
Posted by Craig Markwardt on Thu, 31 Dec 2015 04:47:08 GMT
View Forum Message <> Reply to Message

On Tuesday, December 29, 2015 at 7:57:30 PM UTC-5, vince...@gmail.com wrote:
> Dear all,
>

> I was trying to improve the performance of some pieces of code that are taking forever to run.
> Basically, I'm trying to multiply a set of n matrix (3x3) by a set of n vectors (3x1) without using any for loops. The results of these operations should give me a set of n vectors (3x1).
>
> Let's take a simplified example where n=2. Therefore, I have 2 matrixes (let's call them a and b) that needs to be multiplied to 2 vector (let's call them u and v).
>
> I figured out that the operation could be done by reshaping (using rebin and reform for instance) the matrixes into a bigger array (let's call it M) whose diagonal elements are the a and b matrixes, so that:
>
> M = | a 0 |
> | 0 b |
>
> where a and b are the 3x3 matrixes, and by reshaping the n vectors into in single vector (called I), so that:
>
> I = | u |
> | v |
>
> Then, the results would be:
> R = M.I
>
> Finally, the n vectors would be obtained by reshaping the R vector into n (3x1) vector.
>
> Coming for fortran, I initially coded that by decomposing every single matrix multiplication in a for loop. I then tried to apply the above solution, but it seems a real stretch for me to do it without any loops.
>
> I was thinking that someone already might have faced that problem.

For IDL, FOR loops are not a problem as long as you do a lot of work per iteration. Here is an example, where I literally do the matrix multiplication "by hand."

```
;; Set up some dummy inputs
m = randomn(seed,3,3,1000)    ;; M = Your 3x3xN matrices
u = randomn(seed,3,1000)      ;; U = Your 3xN vectors
v = u*0                       ;; V = The final result

;; Boom!  Write out one row of matrix multiplication and do
;; that operation thrice.
for i = 0, 2 do v(i,*) = m(0,i,*)*u(0,*) + m(1,i,*)*u(1,*) + m(2,i,*)*u(2,*)
```

No FOR loops but it's so fast, who cares. Even with 100x as many matrices on my six year old laptop, it takes barely any time at all.

Craig

Subject: Re: Matrix operations with IDL: Avoiding for loops
Posted by vince33600 on Tue, 05 Jan 2016 21:45:59 GMT
View Forum Message <> Reply to Message

Le mercredi 30 décembre 2015 22:47:12 UTC-6, Craig Markwardt a écrit :
> On Tuesday, December 29, 2015 at 7:57:30 PM UTC-5, vince...@gmail.com wrote:
>> Dear all,
>>
>> I was trying to improve the performance of some pieces of code that are taking forever to run.
>> Basically, I'm trying to multiply a set of n matrix (3x3) by a set of n vectors (3x1) without using any for loops. The results of these operations should give me a set of n vectors (3x1).
>>
>> Let's take a simplified example where n=2. Therefore, I have 2 matrixes (let's call them a and b) that needs to be multiplied to 2 vector (let's call them u and v).
>>
>> I figured out that the operation could be done by reshaping (using rebin and reform for instance) the matrixes into a bigger array (let's call it M) whose diagonal elements are the a and b matrixes, so that:
>>
>> M = | a 0 |
>> | 0 b |
>>
>> where a and b are the 3x3 matrixes, and by reshaping the n vectors into in single vector (called I), so that:
>>
>> I = | u |
>> | v |
>>
>> Then, the results would be:
>> R = M.I
>>
>> Finally, the n vectors would be obtained by reshaping the R vector into n (3x1) vector.
>>
>> Coming for fortran, I initially coded that by decomposing every single matrix multiplication in a for loop. I then tried to apply the above solution, but it seems a real stretch for me to do it without any loops.
>>
>> I was thinking that someone already might have faced that problem.
>
> For IDL, FOR loops are not a problem as long as you do a lot of work per iteration. Here is an example, where I literally do the matrix multiplication "by hand."
>
> ;; Set up some dummy inputs
> m = randomn(seed,3,3,1000) ;; M = Your 3x3xN matrices
> u = randomn(seed,1000) ;; U = Your 3xN vectors
> v = u*0 ;; V = The final result
>
> ;; Boom! Write out one row of matrix multiplication and do
> ;; that operation thrice.

> for i = 0, 2 do v(i,*) = m(0,i,*)*u(0,*) + m(1,i,*)*u(1,*) + m(2,i,*)*u(2,*)
>
> No FOR loops but it's so fast, who cares. Even with 100x as many matrices on my six year old laptop, it takes barely any time at all.
>
> Craig

Thanks Craig!
I guess not every for loops are evil in IDL.

Thanks for the answer anyway,
Vincent
