Subject: Re: Center of mass???
Posted by davidf on Mon, 08 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Paul Hick (pphick@ucsd.edu) writes:

> Reasoning by analogy to the 2D case, this should work, I think:
>
> xcm = Total(Total(Total(array,3),2) * Indgen(s[0])) / totalMass
> ycm = Total(Total(Total(array,3),1) * Indgen(s[1])) / totalMass
> zcm = Total(Total(Total(array,2),1) * Indgen(s[2])) / totalMass

Well, it seems to work in the simple-minded cases I
tried it on. :-)

I'll write it up in an article if no one else has
serious objections.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Center of mass???
Posted by Paul Hick on Mon, 08 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Anders Wennerberg (anders@mrc.ks.se) writes:
>
>> I have tried to find a routine that could calculate the center of
>> mass, in preliminary 2D but in near future 3D. I would be happy if
>> anyone could give direction where to find that kind of routine.
>
> Here is how you do it in 2D. I've never had to extend
> this to 3D, but when you figure it out, please let us
> know. I'll write up an article on it. :-)
>
> s = Size(array, /Dimensions)
> totalMass = Total(array)
> xcm = Total(Total(array,2) * Indgen(s[0])) / totalMass

>   ycm = Total(Total(array,1) * Indgen(s[1])) / totalMass
>
> Cheers,
>
> David

Reasoning by analogy to the 2D case, this should work, I think:

xcm = Total(Total(Total(array,3),2) * Indgen(s[0])) / totalMass
ycm = Total(Total(Total(array,3),1) * Indgen(s[1])) / totalMass
zcm = Total(Total(Total(array,2),1) * Indgen(s[2])) / totalMass

--

 _____ _____
| Paul Hick (pphick@ucsd.edu) |
| Office : SERF Rm. 302 Smail : UCSD/CASS/0424 |
| Phone : (858) 534-8965 9500 Gilman Drive |
| Fax : (858) 534-0177 La Jolla CA 92093-0424 |
| WWW : http://casswww.ucsd.edu/personal/Plh.html |
 |_____ _____|

Subject: Re: Center of mass???
Posted by davidf on Mon, 08 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Anders Wennerberg (anders@mrc.ks.se) writes:

> I have tried to find a routine that could calculate the center of
> mass, in preliminary 2D but in near future 3D. I would be happy if
> anyone could give direction where to find that kind of routine.

Here is how you do it in 2D. I've never had to extend
this to 3D, but when you figure it out, please let us
know. I'll write up an article on it. :-)

```
  s = Size(array, /Dimensions)
  totalMass = Total(array)
  xcm = Total(Total(array,2) * Indgen(s[0])) / totalMass
  ycm = Total(Total(array,1) * Indgen(s[1])) / totalMass
```

Cheers,

David

P.S. Thanks to David Foster for the algorithm above.

--

David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Center of mass???
Posted by Jonathan Joseph on Tue, 09 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Well, I'm not going to take up JD's challenge,
but are you all sure you are answering the right
question?

I mean sure, great, if you happen to have
an MxNx.... array of masses then you've got
everything you need. But when I first read
Anders' post, I thought, "gee that sounds simple."

I thought of N masses at N locations,

m = 1D array of N masses
pos = D x N array of locations of the masses in D dimensions

then:

```
s=size(pos, /dimensions)
mm = m ## replicate(1,s(0))
cm = total(pos * mm, 2) / total(m)
```

Please someone correct me if I'm wrong.
Also, Is there a better way of multiplying
an MxN array by a one dimensional array of
length N such that each row of the MxN array
is multiplied by the corresponding element
of the one dimensional array?

-Jonathan

Subject: Re: Center of mass???
Posted by J.D. Smith on Tue, 09 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>

> Paul Hick (pphick@ucsd.edu) writes:
>
>> Reasoning by analogy to the 2D case, this should work, I think:
>>
>> xcm = Total(Total(Total(array,3),2) * Indgen(s[0])) / totalMass
>> ycm = Total(Total(Total(array,3),1) * Indgen(s[1])) / totalMass
>> zcm = Total(Total(Total(array,2),1) * Indgen(s[2])) / totalMass
>
> Well, it seems to work in the simple-minded cases I
> tried it on. :-)
>
> I'll write it up in an article if no one else has
> serious objections.
>

Why stop at 3 dimensions? How about any dimension:

```
function com, arr,DOUBLE=dbl
  s=size(arr,/DIMENSIONS)
  n=n_elements(s)
  tot=total(arr,DOUBLE=dbl)
  if keyword_set(dbl) then ret=dblarr(n,/NOZERO) else
ret=fltarr(n,/NOZERO)
  for i=0,n-1 do begin
    tmp=arr
    for j=0,i-1   do tmp=total(tmp,1,DOUBLE=dbl)
    for j=i+1,n-1 do tmp=total(tmp,2,DOUBLE=dbl)
    ret[i]=total(findgen(s[i])*tmp,DOUBLE=dbl)/tot
  endfor
  return,ret
end
```

I would have posted this yesterday, but I couldn't help but feel there must be a better way to do it.

Here's why:

In the 3D example above, Paul calculates total(array,3) twice, which means once too many. It could have been saved between steps.

The number of times total() is called on the array in this straightforward, brute-force method is n*(n-1) (for each of n dimensions total the array over the other n-1), not counting the final index total. And many of those are repeats to the exact same total() call with the exact same array! Since we needed to do all directional sub-totals, there must be a more efficient way, analogous to the 3D case of saving total(array,3).

After a bit of scratching on paper, I found that I could do it with only (n-1)(n+2)/2 calls to total(), by working simultaneously from the last dimension backward and the first dimension forward, saving sub-totals which can be shared by subsequent calls in both cases. For 10 dimensions that becomes 54 vs. 90 calls to total() (though only 5 vs. 6 for 3 dimens -- 1 saved call as described above). The problem is how to code this model. Two reciprocating mutually recursive functions should do the trick, but I haven't had time to explore. Any takers? Anybody think they can save more calls to total()?

This may be folly, since few will use it for more than 3D, but it's an interesting problem.

JD

P.S. Another wrinkle: The total()'s you'd prefer to save are the ones that do the most work... those which total the "largest" dimensions. So sorting by dimension size first of all might speed things up even more.

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|

Subject: Re: Center of mass???
Posted by Dick Jackson on Tue, 09 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Paul Hick (pphick@ucsd.edu) writes:

> Reasoning by analogy to the 2D case, this should work, I think:
>
> xcm = Total(Total(Total(array,3),2) * Indgen(s[0])) / totalMass
> ycm = Total(Total(Total(array,3),1) * Indgen(s[1])) / totalMass
> zcm = Total(Total(Total(array,2),1) * Indgen(s[2])) / totalMass

Right, but as a wise man once told me (Dr. Coyote, or something like that), the fastest dimension to run across with things like Total is usually the second dimension. Empirical testing confirms this, so I propose the following, now extended to do 2D or 3D. I also pulled the "/ totalMass" inside a bit to keep the numbers closer to 1, lessen the possibility of overflow and perhaps maintain more precision.

FUNCTION CenterOfMass, array

```
s = Size(array, /Dimensions)
totalMass = Total(array)

CASE Size(array, /N_Dimensions) OF
  2: BEGIN
     xcm = Total(Total(array,2) / totalMass * Indgen(s[0]))
     ycm = Total(Total(array,1) / totalMass * Indgen(s[1]))
     Return, [xcm, ycm]
  END
  3: BEGIN
     totalAcross2 = Total(array, 2)    ; 2 is fastest dim to total across
     xcm = Total(Total(totalAcross2,   2) / totalMass * Indgen(s[0]))
     ycm = Total(Total(Total(array,1), 2) / totalMass * Indgen(s[1]))
     zcm = Total(Total(totalAcross2,   1) / totalMass * Indgen(s[2]))
     Return, [xcm, ycm, zcm]
  END
ENDCASE

END


; Time testing was done as follows:

array = findgen(200, 200, 200)
t0 = systime(1)
print, CenterOfMass(array)
print, systime(1)-t0
```

My timings went from 1.2 seconds (with the previous approach) down to 0.8.

I'd love to see the CASE statement disappear. Who will dare to generalize this to N dimensions, while ensuring that we total over dimension 2 wherever possible?

--
Cheers,
-Dick

Dick Jackson Fanning Software Consulting, Canadian Office
djackson@dfanning.com Calgary, Alberta Voice/Fax: (403) 242-7398
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Center of mass???
Posted by J.D. Smith on Wed, 10 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Jonathan Joseph wrote:

>
> Well, I'm not going to take up JD's challenge,
> but are you all sure you are answering the right
> question?
>
> I mean sure, great, if you happen to have
> an MxNx.... array of masses then you've got
> everything you need. But when I first read
> Anders' post, I thought, "gee that sounds simple."
>
> I thought of N masses at N locations,
>
> m = 1D array of N masses
> pos = D x N array of locations of the masses in D dimensions
>
> then:
>
> s=size(pos, /dimensions)
> mm = m ## replicate(1,s(0))
> cm = total(pos * mm, 2) / total(m)
>
> Please someone correct me if I'm wrong.
> Also, Is there a better way of multiplying
> an MxN array by a one dimensional array of
> length N such that each row of the MxN array
> is multiplied by the corresponding element
> of the one dimensional array?

You can use :

rebin(reform(m,1,N),D,N,/SAMP)*pos

but the array multiplication method also works. The relative speeds are system dependent.

As far as your method of C.O.M. calculation, it's clearly good when you have such a DxN array, or even a sparse array of almost all zeroes (which you can safely ignore in the calculation). However, the application I imagine is some plane or cube or hypercube of data for which the C.O.M. is required. In that case, to use your method, you'd have to inflate your data by a factor of D. That is, you're paying for all those repeated indices being multiplied *before* totalling the data. This is an Index first rather than total first method.

Here is a replacement routine using your idea which takes regular MxNx... arrays of data and makes the DxN index array.

function com2, arr,DOUBLE=dbl

```
  s=size(arr,/DIMENSIONS)
  d=n_elements(s)
  n=n_elements(arr)
  inds=lonarr(d,n,/NOZERO)

  fac=1
  for i=0,d-1 do begin
    inds[i,*]=lindgen(n)/fac mod s[i]
    fac=fac*s[i]
  endfor

  return, total(inds*rebin(reform(arr,1,n),d,n,/SAMP),2,DOUBLE=dbl)/ $
   total(arr,DOUBLE=dbl)
end
```

You see the work here is in generating the index array and inflating the data array. I compared this routine to my other one for a random array of size 10x10x10x10x10. The results were:

Index First Method Time: 0.45424998
Total First Method Time: 0.048227489

How about 1024x1024:

Index First Method Time: 1.9181580
Total First Method Time: 0.23625147

And for something really ludicrous... 5x5x5x5x5x5x5x5

Index First Method Time: 2.7887635
Total First Method Time: 0.44504005

So you see, even for many dimension, for which the Total First routine is currently inefficient, it is always faster. And for big arrays, like 100x100x100x20, I couldn't even get the Index First method to run (memory issues). Too much copying of data.

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|

Subject: Re: Center of mass???
Posted by davidf on Thu, 11 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Anders Wennerberg (anders@mrc.ks.se) writes:

> Thanks for the help!

> David Fanning wrote:
>
>> Paul Hick (pphick@ucsd.edu) writes:
>>
>>> Reasoning by analogy to the 2D case, this should work, I think:
>>>
>>> xcm = Total(Total(Total(array,3),2) * Indgen(s[0])) / totalMass
>>> ycm = Total(Total(Total(array,3),1) * Indgen(s[1])) / totalMass
>>> zcm = Total(Total(Total(array,2),1) * Indgen(s[2])) / totalMass

Well, I'll say this for the simple-minded approach:
it apparently works. :-)

Cheers,

David

P.S. Let's just say I'm still scratching my head
and getting a headache sorting through JD's more
general approach to the problem. I wish I would
have paid more attention in those math classes
way back when. I was still trying to get through
the Alice in Wonderland required reading. :-(

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Center of mass???
Posted by Anders Wennerberg on Thu, 11 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Thanks for the help!
Yours
Anders
David Fanning wrote:

> Paul Hick (pphick@ucsd.edu) writes:
>
>> Reasoning by analogy to the 2D case, this should work, I think:
>>
>> xcm = Total(Total(Total(array,3),2) * Indgen(s[0])) / totalMass
>> ycm = Total(Total(Total(array,3),1) * Indgen(s[1])) / totalMass
>> zcm = Total(Total(Total(array,2),1) * Indgen(s[2])) / totalMass
>
> Well, it seems to work in the simple-minded cases I
> tried it on. :-)
>
> I'll write it up in an article if no one else has
> serious objections.
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting
> Phone: 970-221-0438 E-Mail: davidf@dfanning.com
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Toll-Free IDL Book Orders: 1-888-461-0155

--

B _____
| Anders Wennerberg, PhD, MD The normal disclaimer... .->-.
/o_ Karolinska Hospital & Institutet Tel: +46 (0)8 51772836 `-<-'
| _)Dept of Clinicl Neuroscience Fax: +46 (0)8 339953
\o Clinical Neurophysiology E-mail: anders@mrc.ks.se Transmitted on
| SE-171 76 STOCKHOLM 100% recycled
B Sweden electrons
"Var redo" "Alltid redo"
"Be prepared" "Always prepared"