Subject: Displaying 3-D vector fields
Posted by jim.blackwell on Wed, 06 Nov 2002 20:37:10 GMT
View Forum Message <> Reply to Message

Hi all,

After playing with several pieces of code I've found in the archives,
and not having any luck, I figured I'd ask someone here.

I have X, Y, Z points in space with a,b,c vector component values.
I'd like to plot these in 3-D space. The data points are in the form
of a rectangular regularly spaced grid.

Any help would be appreciated

Jim Blackwell

Subject: Re: Displaying 3-D vector fields
Posted by Rick Towler on Thu, 07 Nov 2002 17:04:58 GMT
View Forum Message <> Reply to Message

This sounds like a job for object graphics.

Someone has to have written a vector object which consists of a few
polylines that make up the body and head in a model. Use would be as simple
as defining the location and magnitude.

Once you have that, something as simple as this would work:

```
;  Your vector locations - XYZ (empty array used as example)
location=FLTARR(100,3)

;  Your vector magnitudes - ABC (empty array used as example)
magnitude=FLTARR(100,3)

;  Create a model to put all of our vectors in
model = OBJ_NEW('IDLgrModel')

;  Fill it up with vector objects
vectors = OBJARR(100)
for n=0, 99 do $
    vectors[n] = OBJ_NEW('vector', LOCATION=location[n,*], $
        MAGNITUDE=magnitude[n,*])

;  Add the array of vectors to our model
model -> Add, vectors
```

```
;  Display the contents of the model using xobjview
xobjview, model, /BLOCK

;  Destroy the objects
OBJ_DESTROY, model
```

If you want to animate the vectors you'll have to do a little more work but
it would be simple.

The trick is finding the "vector" object. Someone on this list has to have
written something similar. I was giving this a day hoping someone with such
an object would step up... Try searching the usual code archives. I
thought Mark Hadfield had something like this but his webpage isn't up
anymore.

If you want to try and write the vector object yourself left me know and I
can help get you started.

-Rick

"Jim" <jim.blackwell@gsfc.nasa.gov> wrote in message
news:95167173.0211061237.389f387a@posting.google.com...
> Hi all,
>
> After playing with several pieces of code I've found in the archives,
> and not having any luck, I figured I'd ask someone here.
>
> I have X, Y, Z points in space with a,b,c vector component values.
> I'd like to plot these in 3-D space. The data points are in the form
> of a rectangular regularly spaced grid.
>
> Any help would be appreciated
>
> Jim Blackwell

Subject: Re: Displaying 3-D vector fields
Posted by rmw092001 on Fri, 08 Nov 2002 07:38:16 GMT

View Forum Message <> Reply to Message

jim.blackwell@gsfc.nasa.gov (Jim) wrote in message
news:<95167173.0211061237.389f387a@posting.google.com>...
> Hi all,
>

> After playing with several pieces of code I've found in the archives,
> and not having any luck, I figured I'd ask someone here.
>
> I have X, Y, Z points in space with a,b,c vector component values.
> I'd like to plot these in 3-D space. The data points are in the form
> of a rectangular regularly spaced grid.
>
> Any help would be appreciated
>
> Jim Blackwell

This plots an arrow in a 3d plot, at any Z value, however the arrow is 'parallel' to the XY plane...

```
pro arrow_3d,xcen,ycen,z,a,arrowsize,thick=thick,color=color

; plots an arrow on a 3d plot
; arrowhead is in the horizontal, xy  plane
;
; INPUT PARAMETERS:
;    xcen, ycen = starting point of arrow in data coordinates
;    z          = height of arrow above xy plane in data coordinates
;    a   = angle, degrees c/clockwise from +X direction, in xy plane
;
; OPTIONAL INPUT PARAMETERS:
; thick      = usual IDL meaning, default = 2.0
; color      = usual IDL meaning, default = !P.COLOR
;
; MODIFICATION HISTORY:
;       merged from one_arrow.pro and one_ray.pro, 15 Aug 2002

 dtr=!pi/180.

 if N_params() LT 5 then begin
  print,'Err: arrow_3d,xcen,ycen,z,angle,arrowsize,[,thick=,color=]'
  return
 endif

 if not keyword_set(thick)     then thick     = 2.0
 if not keyword_set(color)     then color     = !P.COLOR

; forward half of arrow stalk
 plots,[xcen,xcen+arrowsize*cos(a*dtr)/2.],$
       [ycen,ycen+arrowsize*sin(a*dtr)/2.],$
       [z,z], color=color, thick=thick, /t3d
; backward half of arrow stalk
 plots,[xcen,xcen-arrowsize*cos(a*dtr)/2.],$
       [ycen,ycen-arrowsize*sin(a*dtr)/2.],$
```

```
      [z,z], color=color, thick=thick, /t3d
; move to arrow head
 xcen=xcen+arrowsize*cos(a*dtr)/2.
 ycen=ycen+arrowsize*sin(a*dtr)/2.
; plot arrow head
; it's 1/3 the length of stalk, at angle 35 degrees -
 plots,[xcen,xcen+arrowsize*cos((180.+a+35.)*dtr)/3.],$
      [ycen,ycen+arrowsize*sin((180.+a+35.)*dtr)/3.],$
      [z,z], color=color, thick=thick, /t3d
 plots,[xcen,xcen+arrowsize*cos((180.+a-35.)*dtr)/3.],$
      [ycen,ycen+arrowsize*sin((180.+a-35.)*dtr)/3.],$
      [z,z], color=color, thick=thick, /t3d
 return
 end
```

Subject: Re: Displaying 3-D vector fields
Posted by jim.blackwell on Fri, 08 Nov 2002 14:44:34 GMT

View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<aqe77h$1juo$1@nntp6.u.washington.edu>...

```
>  This sounds like a job for object graphics.
>
>  Someone has to have written a vector object which consists of a few
>  polylines that make up the body and head in a model.  Use would be as simple
>  as defining the location and magnitude.
>
>  Once you have that, something as simple as this would work:
>
>  ;  Your vector locations - XYZ (empty array used as example)
>  location=FLTARR(100,3)
>
>  ;  Your vector magnitudes - ABC (empty array used as example)
>  magnitude=FLTARR(100,3)
>
>  ;  Create a model to put all of our vectors in
>  model = OBJ_NEW('IDLgrModel')
>
>  ;  Fill it up with vector objects
>  vectors = OBJARR(100)
>  for n=0, 99 do $
>     vectors[n] = OBJ_NEW('vector', LOCATION=location[n,*], $
>         MAGNITUDE=magnitude[n,*])
>
>  ;  Add the array of vectors to our model
>  model -> Add, vectors
>
```

> ; Display the contents of the model using xobjview
> xobjview, model, /BLOCK
>
> ; Destroy the objects
> OBJ_DESTROY, model
>
>
> If you want to animate the vectors you'll have to do a little more work but
> it would be simple.
>
>
> The trick is finding the "vector" object. Someone on this list has to have
> written something similar. I was giving this a day hoping someone with such
> an object would step up... Try searching the usual code archives. I
> thought Mark Hadfield had something like this but his webpage isn't up
> anymore.
>
> If you want to try and write the vector object yourself left me know and I
> can help get you started.
>
> -Rick

Rick,

Thanks for the advice. As far as a vector object, I presume one could
take the program offered in another reply to this posting and make it
an object ? Not being familiar with Object Graphics other than for
some examples I've tried to figure out, I need some help here.

Jim Blackwell

Subject: Re: Displaying 3-D vector fields
Posted by jim.blackwell on Fri, 08 Nov 2002 15:35:15 GMT
View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<aqe77h$1juo$1@nntp6.u.washington.edu>...
> This sounds like a job for object graphics.
>
> Someone has to have written a vector object which consists of a few
> polylines that make up the body and head in a model. Use would be as simple
> as defining the location and magnitude.
>
> Once you have that, something as simple as this would work:
>
> ; Your vector locations - XYZ (empty array used as example)
> location=FLTARR(100,3)

```
>
> ;  Your vector magnitudes - ABC (empty array used as example)
> magnitude=FLTARR(100,3)
>
> ;  Create a model to put all of our vectors in
> model = OBJ_NEW('IDLgrModel')
>
> ;  Fill it up with vector objects
> vectors = OBJARR(100)
> for n=0, 99 do $
>     vectors[n] = OBJ_NEW('vector', LOCATION=location[n,*], $
>         MAGNITUDE=magnitude[n,*])
>
> ;  Add the array of vectors to our model
> model -> Add, vectors
>
> ;  Display the contents of the model using xobjview
> xobjview, model, /BLOCK
>
> ;  Destroy the objects
> OBJ_DESTROY, model
>
```

>

> If you want to animate the vectors you'll have to do a little more work but
> it would be simple.
>
>
> The trick is finding the "vector" object. Someone on this list has to have
> written something similar. I was giving this a day hoping someone with such
> an object would step up... Try searching the usual code archives. I
> thought Mark Hadfield had something like this but his webpage isn't up
> anymore.
>
> If you want to try and write the vector object yourself left me know and I
> can help get you started.
>
> -Rick
>
>
>
> "Jim" <jim.blackwell@gsfc.nasa.gov> wrote in message
> news:95167173.0211061237.389f387a@posting.google.com...
>> Hi all,
>>
>> After playing with several pieces of code I've found in the archives,
>> and not having any luck, I figured I'd ask someone here.
>>
>> I have X, Y, Z points in space with a,b,c vector component values.

>> I'd like to plot these in 3-D space. The data points are in the form
>> of a rectangular regularly spaced grid.
>>
>> Any help would be appreciated
>>
>> Jim Blackwell

Hey, read about the example program show_stream.pro and it seems to do
almost what I want if I could just figure out how it wants me to input
my data. Any ideas ?

Subject: Re: Displaying 3-D vector fields
Posted by James Kuyper on Fri, 08 Nov 2002 16:05:01 GMT

View Forum Message <> Reply to Message

Jim wrote:
>
> Hi all,
>
> After playing with several pieces of code I've found in the archives,
> and not having any luck, I figured I'd ask someone here.
>
> I have X, Y, Z points in space with a,b,c vector component values.
> I'd like to plot these in 3-D space. The data points are in the form
> of a rectangular regularly spaced grid.
>
> Any help would be appreciated
>
> Jim Blackwell

Would an approach using FLOW3 meet your needs? Even if this isn't quite
what you need, flow3.pro might be a suitable starting point for a
routine customized to your needs.

Subject: Re: Displaying 3-D vector fields
Posted by Rick Towler on Fri, 08 Nov 2002 19:31:12 GMT

View Forum Message <> Reply to Message

"Jim" <jim.blackwell@gsfc.nasa.gov> wrote
> "Rick Towler" <rtowler@u.washington.edu>
>> This sounds like a job for object graphics.
>>
>> Someone has to have written a vector object which consists of a few
>> polylines that make up the body and head in a model. Use would be as
simple

```
>>  as defining the location and magnitude.
>>
>>  Once you have that, something as simple as this would work:
>>
>>  ;  Your vector locations - XYZ (empty array used as example)
>>  location=FLTARR(100,3)
>>
>>  ;  Your vector magnitudes - ABC (empty array used as example)
>>  magnitude=FLTARR(100,3)
>>
>>  ;  Create a model to put all of our vectors in
>>  model = OBJ_NEW('IDLgrModel')
>>
>>  ;  Fill it up with vector objects
>>  vectors = OBJARR(100)
>>  for n=0, 99 do $
>>      vectors[n] = OBJ_NEW('vector', LOCATION=location[n,*], $
>>          MAGNITUDE=magnitude[n,*])
>>
>>  ;  Add the array of vectors to our model
>>  model -> Add, vectors
>>
>>  ;  Display the contents of the model using xobjview
>>  xobjview, model, /BLOCK
>>
>>  ;  Destroy the objects
>>  OBJ_DESTROY, model
>>
>>
>>  If you want to animate the vectors you'll have to do a little more work
but
>>  it would be simple.
>>
>>
>>  The trick is finding the "vector" object.  Someone on this list has to
have
>>  written something similar.  I was giving this a day hoping someone with
such
>>  an object would step up...  Try searching the usual code archives.  I
>>  thought Mark Hadfield had something like this but his webpage isn't up
>>  anymore.
>>
>>  If you want to try and write the vector object yourself left me know and
I
>>  can help get you started.
>>
>>  -Rick
>
```

> Thanks for the advice. As far as a vector object, I presume one could
> take the program offered in another reply to this posting and make it
> an object ? Not being familiar with Object Graphics other than for
> some examples I've tried to figure out, I need some help here.

Well let me introduce you to the wonderful world of Object graphics. :)
Actually, let Ronn Kling do that with his book "Power Graphics with IDL".
You can get it from his website (www.kilvarock.com). You'll need it if you
want to go beyond the basics I outlined above.

I saw your other post too. I haven't looked at show_stream.pro so I can't
help you there. What I can do is provide you with a vector object. I just
whipped this up because I was trying to avoid other work so test it a bit
first to verify it does what it should. There are no guarantees...

Let me know how you make out.

-Rick

```
;+
; NAME:
;       VECTOR__DEFINE
;
; PURPOSE:
;
;       This is an example of a 3D vector class for plotting
;       vector fields.  This object is a subclass of IDLgrModel
;       which contains a polyline object representing a vector
;       provided a given location and magnitude.
;
; AUTHOR:
;       Rick Towler
;       School of Aquatic and Fishery Sciences
;       University of Washington
;       Box 355020
;       Seattle, WA 98195-5020
;       rtowler@u.washington.edu
;       www.acoustics.washington.edu
;
;
; CATEGORY: Object Graphics
;
;
; CALLING SEQUENCE:
;
```

```
;       vectorObject = OBJ_NEW('vector')
;
;
; KEYWORDS:
;
;   This object inherits keywords from it's superclass, IDLgrModel, and
;   passes keywords to IDLgrPolyline.
;
;       location:   A 3 element vector defining the X, Y and Z
;                   coordinates of the vector's location.
;
;       magnitude:  A 3 element vector defining the X, Y and Z
;                   magnitude of the vector.
;
;
; METHODS:
;
;       GetProperty:
;
;       SetProperty:
;
;
; DEPENDENCIES: None.
;
; EXAMPLE:
;
;       vecObj = OBJ_NEW('vector', LOCATION=[0,0,0], MAGNITUDE=[3,2,1], $
;           COLOR=[255,0,0], THICK=2.0)
;
;       xobjview, vecObj
;
;
; MODIFICATION HISTORY:
;       Written by: Rick Towler, 8 November 2002.
;
;-


function Vector::Init,  location=location, $
                        magnitude=magnitude, $
                        _ref_extra=extra


    ;  Check the keywords.
    self.location = (N_ELEMENTS(location) eq 0) ? [0,0,0] : location
    self.magnitude = (N_ELEMENTS(magnitude) eq 0) ? [0,0,-1] : magnitude

    ;  Initialize the superclass.
    ok = self->IDLgrModel::init(/SELECT_TARGET, _EXTRA=extra)
    if (not ok) then return, 0

    ;  Define the unit vector vertices.
    vertices = [[-0.1,0.0,-0.85], $
                [0.0,0.0,-1.0], $
                [0.1,0.0,-0.85], $
                [0,0,0]]

    ;  Connect the dots to form our vector
    polylines = [3,0,1,2,2,1,3]

    ;  Create the vector body
    self.oBody = OBJ_NEW('IDLgrPolyline', vertices, POLYLINES=polylines, $
        _EXTRA=extra)

    ;  Add the polyline to self.
    self -> Add, self.oBody

    ;  "Update" the vector to orient/translate/scale it correctly.
    self -> Update

    RETURN, 1

end


pro Vector::Update

    compile_opt idl2

    ;  Reset our transform.
    self -> Reset

    ;  Rotate the vector.
    lvn = TOTAL(self.magnitude^2)
    if (lvn eq 0.) then begin
        ;  Hide the vector if magnitude=0
        self -> SetProperty, /HIDE
        RETURN
    endif
    self -> SetProperty, HIDE=0
    lMag = SQRT(lvn)
    lvector = self.magnitude / lMag

    yaw = 180. + ATAN(lvector[0],lvector[2]) * !RADEG
    pitch = ATAN(lvector[1], SQRT(lvector[2]^2 + lvector[0]^2)) * !RADEG
    self -> Rotate, [1,0,0], pitch
    self -> Rotate, [0,1,0], yaw

    ;  Scale according to magnitude
    self -> Scale, lMag, lMag, lMag

    ;  Move the vector into place.
    self -> Translate, self.location[0], self.location[1], $
        self.location[2]

    RETURN

end


pro Vector::SetProperty,    location=location, $
                            magnitude=magnitude, $
                            _extra=extra

    compile_opt idl2

    update = 0B

    if (N_ELEMENTS(location) eq 3) then begin
        self.location = location
        update = 1B
    endif

    if (N_ELEMENTS(magnitude) eq 3) then begin
        self.magnitude = magnitude
        update = 1B
    endif

    if (update) then self -> Update

    self->IDLgrModel::SetProperty, _EXTRA=extra
    self.oBody->SetProperty, _EXTRA=extra

end


pro Vector::GetProperty,    location=location, $
                            magnitude=magnitude, $
                            _ref_extra=extra

    compile_opt idl2

    location = self.location
    magnitude = self.magnitude

    self->IDLgrModel::GetProperty, _EXTRA=extra
    self.oBody->GetProperty, _EXTRA=extra

end


pro Vector::Cleanup

    compile_opt idl2

    OBJ_DESTROY, self.oBody

    ;  Call our parents cleanup method
    self->IDLgrModel::Cleanup

end


pro Vector__Define

    struct={Vector, $
            inherits IDLgrModel, $
            oBody:OBJ_NEW(), $

            location:FLTARR(3), $
            magnitude:FLTARR(3) $
            }

end
```

Subject: Re: Displaying 3-D vector fields
Posted by jim.blackwell on Tue, 12 Nov 2002 19:20:07 GMT

View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<aqh3iq$20ts$1@nntp6.u.washington.edu>...
> "Jim" <jim.blackwell@gsfc.nasa.gov> wrote
>> "Rick Towler" <rtowler@u.washington.edu>
>>> This sounds like a job for object graphics.
>>>
>>> Someone has to have written a vector object which consists of a few
>>> polylines that make up the body and head in a model. Use would be as
>   simple
>>> as defining the location and magnitude.
>>>

```
>>>  Once you have that, something as simple as this would work:
>>>
>>>  ;  Your vector locations - XYZ (empty array used as example)
>>>  location=FLTARR(100,3)
>>>
>>>  ;  Your vector magnitudes - ABC (empty array used as example)
>>>  magnitude=FLTARR(100,3)
>>>
>>>  ;  Create a model to put all of our vectors in
>>>  model = OBJ_NEW('IDLgrModel')
>>>
>>>  ;  Fill it up with vector objects
>>>  vectors = OBJARR(100)
>>>  for n=0, 99 do $
>>>      vectors[n] = OBJ_NEW('vector', LOCATION=location[n,*], $
>>>          MAGNITUDE=magnitude[n,*])
>>>
>>>  ;  Add the array of vectors to our model
>>>  model -> Add, vectors
>>>
>>>  ;  Display the contents of the model using xobjview
>>>  xobjview, model, /BLOCK
>>>
>>>  ;  Destroy the objects
>>>  OBJ_DESTROY, model
>>>
>>>
>>>  If you want to animate the vectors you'll have to do a little more work
>   but
>>>  it would be simple.
>>>
>>>
>>>  The trick is finding the "vector" object.  Someone on this list has to
>   have
>>>  written something similar.  I was giving this a day hoping someone with
>   such
>>>  an object would step up...  Try searching the usual code archives.  I
>>>  thought Mark Hadfield had something like this but his webpage isn't up
>>>  anymore.
>>>
>>>  If you want to try and write the vector object yourself left me know and
>   I
>>>  can help get you started.
>>>
>>>  -Rick
>>
>
>>  Thanks for the advice.  As far as a vector object, I presume one could
>>  take the program offered in another reply to this posting and make it
>>  an object ?  Not being familiar with Object Graphics other than for
>>  some examples I've tried to figure out, I need some help here.
>
>  Well let me introduce you to the wonderful world of Object graphics. :)
>  Actually, let Ronn Kling do that with his book "Power Graphics with IDL".
>  You can get it from his website (www.kilvarock.com).  You'll need it if you
>  want to go beyond the basics I outlined above.
>
>  I saw your other post too.  I haven't looked at show_stream.pro so I can't
>  help you there.  What I can do is provide you with a vector object.  I just
>  whipped this up because I was trying to avoid other work so test it a bit
>  first to verify it does what it should.  There are no guarantees...
>
>  Let me know how you make out.
>
>  -Rick
>
>
>
>  ;+
>  ; NAME:
>  ;       VECTOR__DEFINE
>  ;
>  ; PURPOSE:
>  ;
>  ;       This is an example of a 3D vector class for plotting
>  ;       vector fields.  This object is a subclass of IDLgrModel
>  ;       which contains a polyline object representing a vector
>  ;       provided a given location and magnitude.
>  ;
>  ; AUTHOR:
>  ;       Rick Towler
>  ;       School of Aquatic and Fishery Sciences
>  ;       University of Washington
>  ;       Box 355020
>  ;       Seattle, WA 98195-5020
>  ;       rtowler@u.washington.edu
>  ;       www.acoustics.washington.edu
>  ;
>  ;
>  ; CATEGORY: Object Graphics
>  ;
>  ;
>  ; CALLING SEQUENCE:
>  ;
>  ;       vectorObject = OBJ_NEW('vector')
>  ;
> ;
> ; KEYWORDS:
> ;
> ;   This object inherits keywords from it's superclass, IDLgrModel, and
> ;   passes keywords to IDLgrPolyline.
> ;
> ;       location:   A 3 element vector defining the X, Y and Z
> ;                   coordinates of the vector's location.
> ;
> ;       magnitude:  A 3 element vector defining the X, Y and Z
> ;                   magnitude of the vector.
> ;
> ;
> ; METHODS:
> ;
> ;       GetProperty:
> ;
> ;       SetProperty:
> ;
> ;
> ; DEPENDENCIES: None.
> ;
> ; EXAMPLE:
> ;
> ;       vecObj = OBJ_NEW('vector', LOCATION=[0,0,0], MAGNITUDE=[3,2,1], $
> ;           COLOR=[255,0,0], THICK=2.0)
> ;
> ;       xobjview, vecObj
> ;
> ;
> ; MODIFICATION HISTORY:
> ;       Written by: Rick Towler, 8 November 2002.
> ;
> ;-
>
>
> function Vector::Init,  location=location, $
>                         magnitude=magnitude, $
>                         _ref_extra=extra
>
>
>     ;  Check the keywords.
>     self.location = (N_ELEMENTS(location) eq 0) ? [0,0,0] : location
>     self.magnitude = (N_ELEMENTS(magnitude) eq 0) ? [0,0,-1] : magnitude
>
>     ;  Initialize the superclass.
>     ok = self->IDLgrModel::init(/SELECT_TARGET, _EXTRA=extra)
>     if (not ok) then return, 0
```

```
>
>     ;  Define the unit vector vertices.
>     vertices = [[-0.1,0.0,-0.85], $
>                 [0.0,0.0,-1.0], $
>                 [0.1,0.0,-0.85], $
>                 [0,0,0]]
>
>     ;  Connect the dots to form our vector
>     polylines = [3,0,1,2,2,1,3]
>
>     ;  Create the vector body
>     self.oBody = OBJ_NEW('IDLgrPolyline', vertices, POLYLINES=polylines, $
>         _EXTRA=extra)
>
>     ;  Add the polyline to self.
>     self -> Add, self.oBody
>
>     ;  "Update" the vector to orient/translate/scale it correctly.
>     self -> Update
>
>     RETURN, 1
>
>  end
>
>
>  pro Vector::Update
>
>     compile_opt idl2
>
>     ;  Reset our transform.
>     self -> Reset
>
>     ;  Rotate the vector.
>     lvn = TOTAL(self.magnitude^2)
>     if (lvn eq 0.) then begin
>         ;  Hide the vector if magnitude=0
>         self -> SetProperty, /HIDE
>         RETURN
>     endif
>     self -> SetProperty, HIDE=0
>     lMag = SQRT(lvn)
>     lvector = self.magnitude / lMag
>
>     yaw = 180. + ATAN(lvector[0],lvector[2]) * !RADEG
>     pitch = ATAN(lvector[1], SQRT(lvector[2]^2 + lvector[0]^2)) * !RADEG
>
>     self -> Rotate, [1,0,0], pitch
>     self -> Rotate, [0,1,0], yaw
```

```
>
>     ;  Scale according to magnitude
>     self -> Scale, lMag, lMag, lMag
>
>     ;  Move the vector into place.
>     self -> Translate, self.location[0], self.location[1], $
>         self.location[2]
>
>     RETURN
>
> end
>
>
> pro Vector::SetProperty,    location=location, $
>                             magnitude=magnitude, $
>                             _extra=extra
>
>     compile_opt idl2
>
>     update = 0B
>
>     if (N_ELEMENTS(location) eq 3) then begin
>         self.location = location
>         update = 1B
>     endif
>
>     if (N_ELEMENTS(magnitude) eq 3) then begin
>         self.magnitude = magnitude
>         update = 1B
>     endif
>
>     if (update) then self -> Update
>
>     self->IDLgrModel::SetProperty, _EXTRA=extra
>     self.oBody->SetProperty, _EXTRA=extra
>
> end
>
>
> pro Vector::GetProperty,    location=location, $
>                             magnitude=magnitude, $
>                             _ref_extra=extra
>
>     compile_opt idl2
>
>     location = self.location
>     magnitude = self.magnitude
>
```

```
>     self->IDLgrModel::GetProperty, _EXTRA=extra
>     self.oBody->GetProperty, _EXTRA=extra
>
> end
>
>
> pro Vector::Cleanup
>
>     compile_opt idl2
>
>     OBJ_DESTROY, self.oBody
>
>     ;  Call our parents cleanup method
>     self->IDLgrModel::Cleanup
>
> end
>
>
> pro Vector__Define
>
>     struct={Vector, $
>             inherits IDLgrModel, $
>             oBody:OBJ_NEW(), $
>
>             location:FLTARR(3), $
>             magnitude:FLTARR(3) $
>             }
>
> end
```

Hey this is almost what I need. How would one draw more than 1 vector
at a time in the same window ? 3-D axes ? I'll check into the Power
Graphics book, thanks.

Jim Blackwell

Subject: Re: Displaying 3-D vector fields
Posted by Rick Towler on Tue, 12 Nov 2002 20:47:10 GMT
View Forum Message <> Reply to Message

"Jim" <jim.blackwell@gsfc.nasa.gov> wrote

> Hey this is almost what I need. How would one draw more than 1 vector
> at a time in the same window ? 3-D axes ?

By following the steps I laid out in my original post.

Say you want 100 vectors:

```
;  create an object array to hold them

vectors = OBJARR(100)

;  create a 100 instances of my vector object

for n=0, 99 do vectors[n] = OBJ_NEW('vector')

;  Add our vectors to a model

vecModel = OBJ_NEW('IDLgrModel')
vecModel -> Add, vectors


;  Now you have 100 vectors rooted at [0,0,0]
;  with a magnitude of [0,0,1].  You probably want them
;  to do something now...

;  Use the SetProperty method to set each vectors
;  magnitude and location.  I assume you have 2
;  arrays named "mag" and "loc" containing this
;  data.

for n=0, 99 do vectors[n] -> Setproperty, MAGNITUDE=mag[n,*], $
   LOCATION=loc[n,*]

;  Now take a look at what we have.

xobjview, vecModel, /block


;  we're done for now, clean up

OBJ_DESTROY, vecModel
```

If you wanted to animate the vectors you would set up a loop around the call to the setproperty method where you would loop thru the time dimension of your array (if the locations were fixed you wouldn't need to change that property). The only problem with this is that you can't use XOBJVIEW to view an animation. I would suggest working on some static views making sure you understand what you are doing (using XOBJVIEW), then go over to David Fanning's website (www.dfanning.com) and take one of his object graphics programs and hack the vector animation stuff into it (I suggest FSC_SURFACE? I think that might come with axes too).

-Rick

Subject: Re: Displaying 3-D vector fields
Posted by jim.blackwell on Wed, 13 Nov 2002 15:06:01 GMT
View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<aqrphc$26s0$1@nntp6.u.washington.edu>...

```
>  "Jim" <jim.blackwell@gsfc.nasa.gov> wrote
>
>>  Hey this is almost what I need.  How would one draw more than 1 vector
>>  at a time in the same window ? 3-D axes ?
>
>  By following the steps I laid out in my original post.
>
>  Say you want 100 vectors:
>
>  ;  create an object array to hold them
>
>  vectors = OBJARR(100)
>
>  ;  create a 100 instances of my vector object
>
>  for n=0, 99 do vectors[n] = OBJ_NEW('vector')
>
>  ;  Add our vectors to a model
>
>  vecModel = OBJ_NEW('IDLgrModel')
>  vecModel -> Add, vectors
>
>
>  ;  Now you have 100 vectors rooted at [0,0,0]
>  ;  with a magnitude of [0,0,1].  You probably want them
>  ;  to do something now...
>
>  ;  Use the SetProperty method to set each vectors
>  ;  magnitude and location.  I assume you have 2
>  ;  arrays named "mag" and "loc" containing this
>  ;  data.
>
>  for n=0, 99 do vectors[n] -> Setproperty, MAGNITUDE=mag[n,*], $
>      LOCATION=loc[n,*]
>
>  ;  Now take a look at what we have.
>
>  xobjview, vecModel, /block
>
```

>
> ; we're done for now, clean up
>
> OBJ_DESTROY, vecModel
>
>
> If you wanted to animate the vectors you would set up a loop around the call
> to the setproperty method where you would loop thru the time dimension of
> your array (if the locations were fixed you wouldn't need to change that
> property). The only problem with this is that you can't use XOBJVIEW to
> view an animation. I would suggest working on some static views making sure
> you understand what you are doing (using XOBJVIEW), then go over to David
> Fanning's website (www.dfanning.com) and take one of his object graphics
> programs and hack the vector animation stuff into it (I suggest FSC_SURFACE?
> I think that might come with axes too).
>
> -Rick

Rick,

forgive me for being such a doofus, but shouldn't the mag and loc
arrays be 3-D ? Would it be as simple as just adding this code to the
existing code set for creating the vector object ?

Thanks

Jim Blackwell

Subject: Re: Displaying 3-D vector fields
Posted by Rick Towler on Wed, 13 Nov 2002 17:20:47 GMT
View Forum Message <> Reply to Message

"Jim" <jim.blackwell@gsfc.nasa.gov> wrote
>
> forgive me for being such a doofus, but shouldn't the mag and loc
> arrays be 3-D ?

You tell me... Well I guess you just did. The 3rd dimension is time, I
presume. That doesn't change anything except how you subscript your data.

Say you have 100 samples from 100 data points. Your array will be in a form
similar to [point,sample,values] or [100,100,3] where the last dimension is
your x,y,z or u,v,w depending on if we're talking about your location array
or magnitude array. Is that correct?

In my example, I assumed 1 sample from 100 points [100,1,3] which I
simplified to [100,3]. You can change the subscripts in the example to work

with your data set. [n,*] would become something like [n,0,*] for your
first sample, [n,1,*] for your second and so on.

> Would it be as simple as just adding this code to the
> existing code set for creating the vector object ?

I don't understand what you are asking. But if you are asking if you should
add this code to the vector object. No. Stick with my example.

-Rick

Subject: Re: Displaying 3-D vector fields
Posted by jim.blackwell on Thu, 14 Nov 2002 15:33:04 GMT

View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<aqu1qc$28se$1@nntp6.u.washington.edu>...
> "Jim" <jim.blackwell@gsfc.nasa.gov> wrote
>>
>> forgive me for being such a doofus, but shouldn't the mag and loc
>> arrays be 3-D ?
>
> You tell me... Well I guess you just did. The 3rd dimension is time, I
> presume. That doesn't change anything except how you subscript your data.
>
> Say you have 100 samples from 100 data points. Your array will be in a form
> similar to [point,sample,values] or [100,100,3] where the last dimension is
> your x,y,z or u,v,w depending on if we're talking about your location array
> or magnitude array. Is that correct?
>
> In my example, I assumed 1 sample from 100 points [100,1,3] which I
> simplified to [100,3]. You can change the subscripts in the example to work
> with your data set. [n,*] would become something like [n,0,*] for your
> first sample, [n,1,*] for your second and so on.
>
>
>
>> Would it be as simple as just adding this code to the
>> existing code set for creating the vector object ?
>
> I don't understand what you are asking. But if you are asking if you should
> add this code to the vector object. No. Stick with my example.
>
> -Rick
Rick,

actually I have a data file containing the X, Y, Z positions of each
vector along with the magnitudes in the three dimensions. There are
then some (believe it or not) 18,000 individual vectors in the dataset
!

When I asked about adding in the code, I merely meant I wasn't sure
how to combine the vector code with the arrays to load the data.
Basically, how do I make it all work together, do I call the vector
object from the other code ? I'm really finding that I have trouble
thinking in 3-D for some reason and I have a Master's in Physics !!
:() Though it has been many years ago and we weren't programming such
things then.

Cheers,

Jim Blackwell

Subject: Re: Displaying 3-D vector fields
Posted by David Fanning on Thu, 14 Nov 2002 15:52:18 GMT
View Forum Message <> Reply to Message

Jim (jim.blackwell@gsfc.nasa.gov) writes:

> I'm really finding that I have trouble
> thinking in 3-D for some reason and I have a Master's in Physics !!

I used to think getting a degree in Physics was
the best thing that ever happened to me. I've
even argued with my three sons that they would
profit greatly my majoring in Physics in college.
(To blank stares, mostly.)

But lately I've been seeing more and more parallels
to Catholicism and the great quantities of guilt
that having a degree in Physics generates. Solving
physics problems makes you think you can do anything.
So, of course, when the toilet leaks, or the car won't
start, or the bicycle gears won't change, you think
"Oh, well, I can probably fix that." Only to find
(at least in my case) that a degree in Physics
doesn't help all that much. :-(

I spend all my time worrying about what I "should"
be able to do instead of just calling the damn
plumber. :-(

Cheers,

David

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

Subject: Re: Displaying 3-D vector fields
Posted by jim.blackwell on Thu, 14 Nov 2002 17:03:21 GMT

View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<aqu1qc$28se$1@nntp6.u.washington.edu>...
> "Jim" <jim.blackwell@gsfc.nasa.gov> wrote
>>
>> forgive me for being such a doofus, but shouldn't the mag and loc
>> arrays be 3-D ?
>
> You tell me... Well I guess you just did. The 3rd dimension is time, I
> presume. That doesn't change anything except how you subscript your data.
>
> Say you have 100 samples from 100 data points. Your array will be in a form
> similar to [point,sample,values] or [100,100,3] where the last dimension is
> your x,y,z or u,v,w depending on if we're talking about your location array
> or magnitude array. Is that correct?
>
> In my example, I assumed 1 sample from 100 points [100,1,3] which I
> simplified to [100,3]. You can change the subscripts in the example to work
> with your data set. [n,*] would become something like [n,0,*] for your
> first sample, [n,1,*] for your second and so on.
>
>
>
>> Would it be as simple as just adding this code to the
>> existing code set for creating the vector object ?
>
> I don't understand what you are asking. But if you are asking if you should
> add this code to the vector object. No. Stick with my example.
>
> -Rick

Rick,

Okay I get it now Thank God ! I seem to be running into a memory
problem in trying to display 18K vectors at a time though ? I don't
get any indication of this, but it bombs after some number less than
that number of vectors

Jim

Subject: Re: Displaying 3-D vector fields
Posted by jim.blackwell on Thu, 14 Nov 2002 18:11:42 GMT

View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<aqu1qc$28se$1@nntp6.u.washington.edu>...
> "Jim" <jim.blackwell@gsfc.nasa.gov> wrote
>>
>> forgive me for being such a doofus, but shouldn't the mag and loc
>> arrays be 3-D ?
>
> You tell me... Well I guess you just did. The 3rd dimension is time, I
> presume. That doesn't change anything except how you subscript your data.
>
> Say you have 100 samples from 100 data points. Your array will be in a form
> similar to [point,sample,values] or [100,100,3] where the last dimension is
> your x,y,z or u,v,w depending on if we're talking about your location array
> or magnitude array. Is that correct?
>
> In my example, I assumed 1 sample from 100 points [100,1,3] which I
> simplified to [100,3]. You can change the subscripts in the example to work
> with your data set. [n,*] would become something like [n,0,*] for your
> first sample, [n,1,*] for your second and so on.
>
>
>
>> Would it be as simple as just adding this code to the
>> existing code set for creating the vector object ?
>
> I don't understand what you are asking. But if you are asking if you should
> add this code to the vector object. No. Stick with my example.
>
> -Rick

Rick,

what scales the size of the arrow (the length of the arrow relative to
the shaft) ? Is the default vector that is drawn of unit length ?

Jim

Subject: Re: Displaying 3-D vector fields
Posted by Rick Towler on Thu, 14 Nov 2002 19:52:25 GMT
View Forum Message <> Reply to Message

> Okay I get it now Thank God ! I seem to be running into a memory
> problem in trying to display 18K vectors at a time though ? I don't
> get any indication of this, but it bombs after some number less than
> that number of vectors

18K... That's a lot, but hopefully not too many. You do realize that you will not be viewing these in real time. What are your system spex?

I used the test program below to generate an array of 20K vector objects. The array of objects and *one sample* of location and magnitude data required about 31MB. Displaying this dataset using XOBJVIEW required about 50MB. IDLDE required around 70MB total as reported by windows task manager.

So it seems like you should be able to do this on a relatively new PC given you manage your data. You probably can't just read in all of your data at once. I would read in a single sample (or time step or whatever you want to call it) at a time, plot it, then read in the next and plot, and so on. Something like:

```
nvectors=18000
nsamples=1000

;  create our data arrays
magnitudes=FLTARR(3,nvectors)
locations=FLTARR(3,nvectors)

;  create the vector objects
vectors = OBJARR(nvectors)
for n=0, nvectors-1 do vectors[n] = OBJ_NEW('vector', color=[255,0,0])

for n=0, nsamples-1 do begin

   ;  read in this samples data
   readf, 1, magnitudes
   readf, 2, locations

   ;  update the vectors
   for n=0, nvectors-1 do vectors[n] -> SetProperty, $
      MAGNITUDE=magnitudes[*,n], LOCATION=locations[*,n]

   ;  draw the scene
   oWindow -> Draw, oView

   ;  Grab the image in the window and write to file
```

```
endfor

;  clean up
obj_destroy, vectors
```

This is just a start. I have left a lot of detail out and the readf
statements are wrong I'm sure but the general form is there.

This does bring up the question as to how you are going to 'view' this
visualization. with 18k vectors you are not going to be able to view the
entire scene and make anything out I don't think. And since you will not be
able to view this in real time you won't be able to "fly around" in the
vector field (on my well endowed PC it took maybe 1.5 minutes for XOBJVIEW
to draw the 20k vector scene).

If you want to try to view the entire scene then you can't just rely on the
length of the vector to present your information. One thought would be to
modify the vector object so that it's color is representative of it's
magnitude. Another option would be to script camera movement and do a fly
thru/around of the vector field.

But I suppose we should concentrate on getting something on the screen
first. :)

-Rick

```
start_mem=MEMORY(/current)

locations=FLTARR(3,20000)
magnitudes=RANDOMU(SYSTIME(/seconds),3,20000)

;  create a regular grid of vectors 200x100 in the xz plane
for n=0, 199 do locations[0,n*100:(n*100)+99]=FINDGEN(100)
for n=0, 199 do locations[2,n*100:(n*100)+99]=n

vectors = objarr(20000)

for n=0, 19999 do vectors[n] = OBJ_NEW('vector', LOCATION=locations[*,n], $
   MAGNITUDE=magnitudes(*,n), COLOR=[0,0,220])

;print,'Mem in MB:',(MEMORY(/highwater) - start_mem) / 1024. / 1024.

xobjview, vectors, /block
```

print,'Mem in MB:',(MEMORY(/highwater) - start_mem) / 1024. / 1024.

obj_destroy, vectors

end

Subject: Re: Displaying 3-D vector fields
Posted by Rick Towler on Thu, 14 Nov 2002 20:13:42 GMT
View Forum Message <> Reply to Message

"Jim" <jim.blackwell@gsfc.nasa.gov> wrote

> what scales the size of the arrow (the length of the arrow relative to
> the shaft)?

The vector is scaled according to the sqrt() of the sum of the squares of the components of the magnitude. The *entire* vector is scaled, head and all since I am using the model's transformation matrix to do the work for me. The vector is not drawn if the magnitude is 0.

> Is the default vector that is drawn of unit length ?

Yes, the default vector is 1 unit long. It's default magnitude is [0,0,-1] which makes it parallel to the z axis pointing away from the viewer. It's default location is [0,0,0].

-Rick

Subject: Re: Displaying 3-D vector fields
Posted by Mark Hadfield on Thu, 14 Nov 2002 20:52:51 GMT
View Forum Message <> Reply to Message

"Jim" <jim.blackwell@gsfc.nasa.gov> wrote in message
news:95167173.0211140903.5db20620@posting.google.com...
> Okay I get it now Thank God ! I seem to be running into a memory
> problem in trying to display 18K vectors at a time though ? I don't
> get any indication of this, but it bombs after some number less than
> that number of vectors

As I understand this thread to date, you are displaying 18K vectors with an array of 18K vector objects. No??

If so, I'm not surprised you are running into memory problems. (If not, *I* am running into memory problems.) That approach won't work beyond 1000 or

so, in my experience. One the other hand, representing 18K vectors with 18K
line segments in an IDLgrPolyline is no big deal. (Though I must say viewing
them on the screen might be!)

Creating an IDLgrPolyline object to display these vectors isn't all that
hard, you just have to load an array specifying positions corresponding to
the ends of each vector (the DATA property) and the connectivity array which
tells the polyline which ones to connect (the POLYLINES property).

As Rick (I think) commented earlier, my IDL code library contains an object
class (MGHgrBarbPlot) that does exactly this & hides the details. You just
have to give it arrays defining the origin and displacement of each vector
in 3D space. Unfortunately my library is off the air at the moment. (I
really must do something about this.) Jim, why don't you email me & I can
send you the code for the barb-plot object plus some examples.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Displaying 3-D vector fields
Posted by Rick Towler on Thu, 14 Nov 2002 21:39:43 GMT

View Forum Message <> Reply to Message

"Mark Hadfield" <m.hadfield@niwa.co.nz> wrote

Now you speak up!

I was hoping someone would come up with a solution for Jim but since nobody
did, I wrote the vector object. At the time I had no idea he was going to
display 18K vectors so the single vector design seemed sound and making it a
subclass of IDLgrModel made dealing with the geometry of the head easy.

> I'm not surprised you are running into memory problems. (If not, *I*
> am running into memory problems.) That approach won't work beyond 1000 or
> so, in my experience. One the other hand, representing 18K vectors with
18K
> line segments in an IDLgrPolyline is no big deal. (Though I must say
viewing
> them on the screen might be!)

I can easily create 20k individual vector objects so I don't know what you
have run into in the past. This approach will cost him an extra 8MB or so
with the IDLgrModel overhead but what the heck, memory is cheap!

> Creating an IDLgrPolyline object to display these vectors isn't all that
> hard, you just have to load an array specifying positions corresponding to
> the ends of each vector (the DATA property) and the connectivity array
which
> tells the polyline which ones to connect (the POLYLINES property).

True, but dealing with the vector head using this approach is an, um,
headache.

> Unfortunately my library is off the air at the moment. (I
> really must do something about this.)

Please do!

-Rick

Subject: Re: Displaying 3-D vector fields
Posted by Mark Hadfield on Thu, 14 Nov 2002 23:47:06 GMT
View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:ar15bu$uku$1@nntp6.u.washington.edu...

> Now you speak up!

My apologies. I did submit a message to this thread earlier but
somehow it never showed up.

> I can easily create 20k individual vector objects so I don't know
> what you have run into in the past. This approach will cost him an
> extra 8MB or so with the IDLgrModel overhead but what the heck,
> memory is cheap!

OK, it's not memory that kills you when you have a large number of
objects, it's redraw speed. (Another apology, this time for
looseness.) I based this assertion on experience with scatter plots,
discussed on the group several times in the past: when you're dealing
with > 1000 geometric entities, combining them into one object gives
much faster redraws than having a separate object for each one. I
thought I should check that this principle out for the current case,
so I just compared a barb plot with 1000 line segments in one
IDlgrPolyline vs a barb plot with a 1000 IDLgrPolylines. Sure enough,

the single-object version is redrawn very snappily, the multi-object
version takes 0.5 s or so per redraw. So, on my machine at least, 18 K
objects would be pretty slow.

> True, but dealing with the vector head using this approach is an,
> um, headache.

Oh right, I hadn't thought about this much. I have given up on trying
to draw heads on my barbs (can't be bothered with all the geometry)
but my MGHgrBarbPlot object can draw a symbol at the base of each
barb. I find I prefer the look of this anyway.

>> Unfortunately my library is off the air at the moment. (I really
>> must do something about this.) > > Please do!

Yes, I really really will. I promise.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Displaying 3-D vector fields
Posted by David Fanning on Fri, 15 Nov 2002 00:17:19 GMT
View Forum Message <> Reply to Message

Mark Hadfield (m.hadfield@niwa.co.nz) writes:

>>> Unfortunately my library is off the air at the moment. (I really
>>> must do something about this.) > > Please do!
>
> Yes, I really really will. I promise.

I'd be happy to host this library on a temporary basis,
Mark. It would give us the opportunity to, uh, communicate. :-)

Cheers,

David

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

Subject: Re: Displaying 3-D vector fields
Posted by jim.blackwell on Mon, 18 Nov 2002 16:43:27 GMT
View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<ar10ak$22do$1@nntp6.u.washington.edu>...
> "Jim" <jim.blackwell@gsfc.nasa.gov> wrote
>
>> what scales the size of the arrow (the length of the arrow relative to
>> the shaft)?
>
> The vector is scaled according to the sqrt() of the sum of the squares of
> the components of the magnitude. The *entire* vector is scaled, head and
> all since I am using the model's transformation matrix to do the work for
> me. The vector is not drawn if the magnitude is 0.
>
>
>> Is the default vector that is drawn of unit length ?
>
> Yes, the default vector is 1 unit long. It's default magnitude is [0,0,-1]
> which makes it parallel to the z axis pointing away from the viewer. It's
> default location is [0,0,0].
>
> -Rick

I see. Say have you looked at the IDL program "vector_field" ? Seems
to do something similar in that it creates vectors, however I don't
know offhand how to implement this in IDL object graphics to see what
it looks like ?

Subject: Re: Displaying 3-D vector fields
Posted by Rick Towler on Mon, 18 Nov 2002 19:54:39 GMT
View Forum Message <> Reply to Message

"Jim" <jim.blackwell@gsfc.nasa.gov> wrote
> "Rick Towler" <rtowler@u.washington.edu> wrote
>> "Jim" <jim.blackwell@gsfc.nasa.gov> wrote
>>
>>> what scales the size of the arrow (the length of the arrow relative to
>>> the shaft)?
>>
>> The vector is scaled according to the sqrt() of the sum of the squares
of
>> the components of the magnitude. The *entire* vector is scaled, head
and
>> all since I am using the model's transformation matrix to do the work
for

>> me. The vector is not drawn if the magnitude is 0.
>>
>>
>>> Is the default vector that is drawn of unit length ?
>>
>> Yes, the default vector is 1 unit long. It's default magnitude is
[0,0,-1]
>> which makes it parallel to the z axis pointing away from the viewer.
It's
>> default location is [0,0,0].
>>
>> -Rick
>
> I see. Say have you looked at the IDL program "vector_field" ? Seems
> to do something similar in that it creates vectors, however I don't
> know offhand how to implement this in IDL object graphics to see what
> it looks like ?

No. I haven't looked at it. I was satisfied with my vector as is. Well,
sort of.

After reading Mark's comments I was curious as to the impact of having large
numbers of IDLgrModel objects in a scene. In our case it comes down to the
cost of calculating the numerous transforms and internal procedural
overhead.

Ideally, as Mark has implemented in his barb plot, you'll have one model and
one atom. Ignoring any parent models, IDL calculates a transform based on
the one model and one atom transform (3x3 coord_conv transform) and applies
it to all the vertices. Very efficient, but limiting.

On the other end of the spectrum is a scene with 20k of my vector objects.
Assuming we put our 20k models into a parent model, we need to do 40k matrix
multiplies then apply each of those 20k transforms to the 4 verts that make
up the vector. Not a model of efficiency but infinitely flexible.

Somewhere in the middle is a single model which contains 20k IDLgrPolyline
objects. I modified my original vector object and created a 3d vector field
object following this design. I end up calculating the transform and
applying it to the vertex data manually when the location or magnitude data
is changed. What little I did play with it showed around a 30-40% increase
in redraw speed with 20k vectors. Although unfinished, you are welcome to
this code as well. The meat is there, you'll just need to add the dressing.

In your case, none of these improvements will get you to the point where you
will be able to interact in real time. It takes a long time to draw nearly
80K vertices. I think that you could go with any of them.

But back to your question. You still seem to be looking for an answer.
What do you need that Mark's barb plot or my vector can't provide?

-Rick

Subject: Re: Displaying 3-D vector fields
Posted by Struan Gray on Tue, 19 Nov 2002 10:21:43 GMT
View Forum Message <> Reply to Message

Rick Towler, rtowler@u.washington.edu writes:

> Ideally, as Mark has implemented in his barb plot, you'll have one model and
> one atom. Ignoring any parent models, IDL calculates a transform based on
> the one model and one atom transform (3x3 coord_conv transform) and applies
> it to all the vertices. Very efficient, but limiting.
>
> On the other end of the spectrum is a scene with 20k of my vector objects.
> Assuming we put our 20k models into a parent model, we need to do 40k matrix
> multiplies then apply each of those 20k transforms to the 4 verts that make
> up the vector. Not a model of efficiency but infinitely flexible.

We had a thread related to this a couple of years ago. If you search google.groups for "Object graphic 3d Scatterplot" (and/or my name) you will turn it up. There are a couple of other options.

The first is to use individual models for every object, but to use an alias. If your barb or arrow can be scaled without looking ugly (i.e. you don't mind the head scaling with the body) you can have a single barb object, and add it as an alias to 20k individual model objects. Each model object can have a different scaling, so your barbs can vary in size. This is actually a bit slower to plot than just creating a barb object for each point, but it is faster to create the model, and a lot faster if you want to animate it by, say, changing the barb colour for successive plots.

The second is to use the symbol keyword to a polyline plot. Even if you don't plot any lines, this turns out to be faster than creating multiple objects. I assume you strip out some overhead by having some of the loops through the object hiearchy done in native code.

The really cool trick though is to recognise that a 20k plot on any real reproduction medium is only going to have a limited subset of the 20k possible symbols. In the thread, the example was that if you use a palette to colour the symbols, there are only 256 possible colours, and so only 256 possible symbols, not 20k. In your case, I suspect you could select a small number of barb lengths and nobody would ever be able to tell.

This is of course, just a sneaky way of getting HISTOGRAM into the conversation: you create the desired characteristics for your barbs, and then use HISTOGRAM to distill them down into a subset. REVERSE_INDICES will tell you which data point uses which of the symbols in the subset. You can then use either the polyline+symbol technique, or make individual models for each point and add the relevant symbol as an alias.

This may seem tedious, but for 20k objects every little helps.

Struan

Subject: Re: Displaying 3-D vector fields
Posted by jim.blackwell on Tue, 19 Nov 2002 14:48:53 GMT

View Forum Message <> Reply to Message

"Rick Towler" <rtowler@u.washington.edu> wrote in message
news:<arbgn1$1pbo$1@nntp6.u.washington.edu>...
> "Jim" <jim.blackwell@gsfc.nasa.gov> wrote
>> "Rick Towler" <rtowler@u.washington.edu> wrote
>>> "Jim" <jim.blackwell@gsfc.nasa.gov> wrote
>>>
>>>> what scales the size of the arrow (the length of the arrow relative to
>>>> the shaft)?
>>>
>>> The vector is scaled according to the sqrt() of the sum of the squares
>   of
>>> the components of the magnitude. The *entire* vector is scaled, head
>   and
>>> all since I am using the model's transformation matrix to do the work
>   for
>>> me. The vector is not drawn if the magnitude is 0.
>>>
>>>
>>>> Is the default vector that is drawn of unit length ?
>>>
>>> Yes, the default vector is 1 unit long. It's default magnitude is
>   [0,0,-1]
>>> which makes it parallel to the z axis pointing away from the viewer.
>   It's
>>> default location is [0,0,0].
>>>
>>> -Rick
>>
>> I see. Say have you looked at the IDL program "vector_field" ? Seems

>> to do something similar in that it creates vectors, however I don't
>> know offhand how to implement this in IDL object graphics to see what
>> it looks like ?
>
> No. I haven't looked at it. I was satisfied with my vector as is. Well,
> sort of.
>
> After reading Mark's comments I was curious as to the impact of having large
> numbers of IDLgrModel objects in a scene. In our case it comes down to the
> cost of calculating the numerous transforms and internal procedural
> overhead.
>
> Ideally, as Mark has implemented in his barb plot, you'll have one model and
> one atom. Ignoring any parent models, IDL calculates a transform based on
> the one model and one atom transform (3x3 coord_conv transform) and applies
> it to all the vertices. Very efficient, but limiting.
>
> On the other end of the spectrum is a scene with 20k of my vector objects.
> Assuming we put our 20k models into a parent model, we need to do 40k matrix
> multiplies then apply each of those 20k transforms to the 4 verts that make
> up the vector. Not a model of efficiency but infinitely flexible.
>
> Somewhere in the middle is a single model which contains 20k IDLgrPolyline
> objects. I modified my original vector object and created a 3d vector field
> object following this design. I end up calculating the transform and
> applying it to the vertex data manually when the location or magnitude data
> is changed. What little I did play with it showed around a 30-40% increase
> in redraw speed with 20k vectors. Although unfinished, you are welcome to
> this code as well. The meat is there, you'll just need to add the dressing.
>
> In your case, none of these improvements will get you to the point where you
> will be able to interact in real time. It takes a long time to draw nearly
> 80K vertices. I think that you could go with any of them.
>

> But back to your question. You still seem to be looking for an answer.
> What do you need that Mark's barb plot or my vector can't provide?
>
> -Rick

Rick,

Mainly I just wanted to see what the differences were between the barb
plot and yours. My main problem is that 18K vectors is just too many
and I need some way to resample the grid so the grid is of lower
resolution.

Also, I wanted to be able to color the vectors based on their value.

Jim