Subject: Re: find a plane in a 3D plot
Posted by pgrigis on Fri, 12 Sep 2008 13:45:10 GMT
View Forum Message <> Reply to Message

If a field line is embedded in a plane, then the vector product
of any two vectors tangent to it will point in the direction of
the perpendicular of the plane. If not, the vector products will
not be parallel.

Paolo

Nicola wrote:
> Dear folks,
> I'm facing a problem which I do not have any idea how to solve.
> Imagine that you have a 3 dimensional field as a function of time, say
> the three component of a magnetic field for example. They are stored
> in a fltarr(nsample,3).
> If plotted in a 3D box (by using for example plot_3dbox) they describe
> a closed orbit. I do know that they actually describe an ellipse (or
> something similar considering the fact that we are dealing with
> experimental data which not always correspond to theory!) and that
> this ellipse (or closed path) lies on a plane which have a certain
> inclination with respect to the three magnetic axis. I have to find a
> way to identify this plane and the direction perpendicular to this
> plane in the more accurate way as possible.
> Do you have any idea how to proceed? Any suggestion will be very very
> appreciated
> Thank you very much
> nicola

Subject: Re: find a plane in a 3D plot
Posted by Brian Larsen on Fri, 12 Sep 2008 13:59:50 GMT
View Forum Message <> Reply to Message

Nicola,

One way this is done (maybe the way) is to do an eigenvector
decomposition of the data. One answer you get from this analysis is
the maximum, minimum, and intermediate variance directions. For your
data the maximum should be along the long axis of the ellipse,
intermediate along the short axis, and minimum should be the third
direction assuming things are really moving in a plane (or close to
it).

The particulars of how to actually do this depend on the data, format,
and all that jazz but its not generally too hard. Have a look on
wikipedia for eigenvectors and maybe even principal component analysis

(this is useful but not always comprehensible, at least for me).

Cheers,

Brian

 -- --------------
Brian Larsen
Boston University
Center for Space Physics
http://people.bu.edu/balarsen/Home/IDL

Subject: Re: find a plane in a 3D plot
Posted by David Fanning on Fri, 12 Sep 2008 14:29:35 GMT
View Forum Message <> Reply to Message

Brian Larsen writes:

> The particulars of how to actually do this depend on the data, format,
> and all that jazz but its not generally too hard. Have a look on
> wikipedia for eigenvectors and maybe even principal component analysis
> (this is useful but not always comprehensible, at least for me).

A careful reading of the Lindsay Smith article finally sorted
me out on this topic. Best explanation I ever read. Now, it
seems, I make my living doing Principal Component Analysis. :-(

   http://www.dfanning.com/code_tips/pca.html

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: find a plane in a 3D plot
Posted by Wox on Fri, 12 Sep 2008 14:47:18 GMT
View Forum Message <> Reply to Message

On Fri, 12 Sep 2008 02:54:09 -0700 (PDT), Nicola
<nicola.vianello@gmail.com> wrote:

> I have to find a
> way to identify this plane and the direction perpendicular to this
> plane in the more accurate way as possible.

The code below is one way of doing things. The resulting plane is
defined with a normal vector and a point.

```
pro test
x=[1.,0,1,2,3,4]
y=[0.,1,1,2,3,4]
z=[2.,2,2,2,2,2]

; Orthogonal distance regression
; check e.g. http://mathforum.org/library/drmath/view/63765.html

; Centroid: orthogonal distance
; regression plane goes through it
n=n_elements(x)
data=transpose([[x],[y],[z]])
centroid=total(data,2)/n

data[0,*]-=centroid[0]
data[1,*]-=centroid[1]
data[2,*]-=centroid[2]

SVDC, data, W, U, V

smallest_singularvalue=min(W,ind)
plane_normal=reform(V[ind,*])

print,'Orthogonal distance regression plane'
print,'1. goes through: ',centroid
print,'2. has normal: ',plane_normal
end;pro test
```

Subject: Re: find a plane in a 3D plot
Posted by Nicola on Fri, 12 Sep 2008 16:20:43 GMT
View Forum Message <> Reply to Message

On Sep 12, 4:47 pm, Wox <nom...@hotmail.com> wrote:
> On Fri, 12 Sep 2008 02:54:09 -0700 (PDT), Nicola

>
> <nicola.viane...@gmail.com> wrote:
>> I have to find a
>> way to identify this plane and the direction perpendicular to this
>> plane in the more accurate way as possible.
>
> The code below is one way of doing things. The resulting plane is
> defined with a normal vector and a point.
>

```
> pro test
> x=[1.,0,1,2,3,4]
> y=[0.,1,1,2,3,4]
> z=[2.,2,2,2,2,2]
>
> ; Orthogonal distance regression
> ; check e.g.http://mathforum.org/library/drmath/view/63765.html
>
> ; Centroid: orthogonal distance
> ; regression plane goes through it
> n=n_elements(x)
> data=transpose([[x],[y],[z]])
> centroid=total(data,2)/n
>
> data[0,*]-=centroid[0]
> data[1,*]-=centroid[1]
> data[2,*]-=centroid[2]
>
> SVDC, data, W, U, V
>
> smallest_singularvalue=min(W,ind)
> plane_normal=reform(V[ind,*])
>
> print,'Orthogonal distance regression plane'
> print,'1. goes through: ',centroid
> print,'2. has normal: ',plane_normal
> end;pro test
```

Thank you all. This was something I was thinking about, essentially
similar to what is called Minimum variance Method which is a method
used for Cluster data satellite. Now I have a perhaps smallest
problem, which is the graphics..... How I put the found plane on the
same 3D box? I know perhaps this is a stupid question but my knowledge
of IDL 3D plotting is quite scarce....
regards to all of you for your quick reply
Nicola

Subject: Re: find a plane in a 3D plot
Posted by Wox on Mon, 22 Sep 2008 09:15:45 GMT
View Forum Message <> Reply to Message

On Fri, 12 Sep 2008 09:20:43 -0700 (PDT), Nicola
<nicola.vianello@gmail.com> wrote:

> How I put the found plane on the same 3D box?

Well, I'm not sure there is an easy way, but this is what I would do:

1. The plane you have is defined by its normal n and a point p0:
n.(p-p0)=0 (. = scalar product)

2. Consider a box around your data:
p1 = 12 starting points of the lines that make the box
p2 = 12 end points of the lines that make the box

3. Find all intersections between the box-lines and the plane:
t = [n.(p0-p1)]/[n.(p2-p1)]
=> (numerator eq 0) AND (denominator eq 0) =>line in plane => t=1
=> (numerator ne 0) AND (denominator eq 0) => no intersection

intersections = p1 + t(p2-p1)
=> keep only those with 0<=t<=1

see http://local.wasp.uwa.edu.au/~pbourke/geometry/planeline/

4. Sort the resulting intersections and make an IDLgrPolygon object
with them and set alpha=0.5 for transparency. Add also an
IDLgrPolyline with the same intersection points to emphasize the plane
boarder. Make 12 IDLgrPolylines for the box.

5. Make an IDLgrPolyline with your datapoints (x,y,z) and set
linestyle=6 and use a symbol.

6. Plotting: see below. It's rather complicated, but it might be
worthwhile to learn about object graphics. In this case you need it
because of the transparent plane. If you want to use direct graphics,
check http://www.dfanning.com/tips/transparent.html

```
; ----Create App----
base=widget_base(/column)
draw=widget_draw(base,xsize=500,ysize=500,GRAPHICS_LEVEL=2)
WIDGET_CONTROL, base, /REALIZE
widget_control,draw,get_value=oWindow
```

```
; ----Create view----
myview=[-0.5,-0.5,2,2]
oView = OBJ_NEW('IDLgrView',PROJECTION=1,$
EYE=3,$ ; coord as for zclip (doesn't change anything for PROJ=1)
ZCLIP=[2.,-2.],$; normal. coord. relative to surface plot range (x, y
or z)
VIEWPLANE_RECT=myview ,$; normal. coord. relative to surface plot x
and y range
LOCATION=[0,0],dimensions=[1,1],$; Occupy whole window if units=3
units=3); 3: normalised relative to the graphics destination's rect

; ----Create model----
oTop = OBJ_NEW('IDLgrModel')
oGroup = OBJ_NEW('IDLgrModel')
oTop->Add, oGroup

; ----Create axis----
otitlefont= OBJ_NEW('IDLgrFont','times',size=8)
oticfont = OBJ_NEW('IDLgrFont','times',size=7)
if not keyword_set(xtitle) then xtitle='X'
if not keyword_set(ytitle) then ytitle='Y'
if not keyword_set(ztitle) then ztitle='Z'
oXtitle=OBJ_NEW('IDLgrText',xtitle,FONT=otitlefont,
Recompute_Dimensions=2)
oYtitle=OBJ_NEW('IDLgrText',ytitle,FONT=otitlefont,
Recompute_Dimensions=2)
oZtitle=OBJ_NEW('IDLgrText',ztitle,FONT=otitlefont,
Recompute_Dimensions=2)

ticklen=bytarr(3)
oXAxis = OBJ_NEW('IDLgrAxis', 0, RANGE=xrange,TICKLEN=ticklen[0],$
title=oXtitle,/exact,TEXTPOS=0)
oYAxis = OBJ_NEW('IDLgrAxis', 1, RANGE=yrange,TICKLEN=ticklen[1],$
title=oYtitle,/exact,TEXTPOS=0)
oZAxis = OBJ_NEW('IDLgrAxis', 2, RANGE=zrange,TICKLEN=ticklen[2],$
title=oZtitle,/exact,TEXTPOS=0)

oXAxis->GetProperty, TICKTEXT = xtick_text
oYAxis->GetProperty, TICKTEXT = ytick_text
oZAxis->GetProperty, TICKTEXT = ztick_text
xtick_text->SetProperty, font=oticfont, Recompute_Dimensions=2
ytick_text->SetProperty, font=oticfont, Recompute_Dimensions=2
ztick_text->SetProperty, font=oticfont, Recompute_Dimensions=2

oGroup->Add, oXAxis
oGroup->Add, oYAxis
oGroup->Add, oZAxis
```

```
; ----Add your objects here----
oGroup->Add, oPlane ; this is a polygon
oGroup->Add, oPlaneBoarder ; polyline
oGroup->Add, oBox ; polyline
oGroup->Add, oDatapoints ; polyline

; ----Create some lights----
oLight1 = OBJ_NEW('IDLgrLight', LOCATION=[2,2,2], TYPE=1)
oTop->Add, oLight1
oLight2 = OBJ_NEW('IDLgrLight', TYPE=0, INTENSITY=0.5)
oTop->Add, oLight2

; ----Place the model in the view----
oView->Add, oTop

; ----Rotate to standard view for first draw----
SURFR,ax=30,az=-30
oGroup -> SetProperty, TRANSFORM =!p.t

; ----Create a holder object for easy destruction----
oHolder = OBJ_NEW('IDL_Container')
oHolder->Add, oView
oHolder->Add, oXtitle
oHolder->Add, oYtitle
oHolder->Add, oZtitle
oHolder->Add, otitlefont
oHolder->Add, oticfont

oWindow->draw, oView

; ----Destroy everything with you don't need it anymore----
obj_destroy,oHolder
```