Re: Fitting a 3D circle to a lobsided set of points

Source: http://sci.tech−archive.net/Archive/sci.math.num−analysis/2008−11/msg00301.html

---

- *From*: Gordon Sande <g.sande@xxxxxxxxxxxxxxxx>
- *Date*: Wed, 26 Nov 2008 14:00:19 GMT

On 2008−11−26 09:28:12 −0400, spellucci@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (Peter Spellucci) said:

> In article <9c19126b−24b3−4bf7−94c7−850bfc633ea6@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>, Dave the Funkatron <dave.rudolf@xxxxxxxx> writes:
>
> > On Nov 25, 4:50 am, spellu...@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (Peter Spellucci) wrote:
> >
> > > In article <84a6d604−e243−425c−be7d−610c8ccd6...@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>, Dave the Funkatron <dave.rud...@xxxxxxxx> writes:
> > >
> > > > Hi all,
> > > >
> > > > Suppose I have a set of 3D points that are roughly in one quadrant of a circle, and I want to find the plane that they are on, as well as
> > >
> > > they are on a plane or on the sphere ???? or inside the ?? orthnat of the sphere?
> >
> > They are roughly on a plane (though with noise and such), so the points would be on the perimeter of a circle. However, they will not be evenly distributed around the circle, but maybe only covering 90 degrees of the circle.
>
> if you can fit by a plane, then there are many spheres which satisfy your

description, so i guess you mean "on" the sphere:

Yup, but there are not an infinite number of circles. If I constrain the sphere to be on that plane, it narrows things down a bit.

you are right, I misunderstood this. your problem is a circle, but in 3D.

well

sum_{i=1 to N} (sqrt((x_i
−xmid)^2+(y_i−ymid)^2+(z_i−zmid)^2)−radius)^2
= min subject to xmid,ymid,zmid,radius.

well, nonlinear, −> levenberg−marquardt or gauss−newton
−> elsunc
(this is software onhttp://plato.asu.edu/sub/nonlsq.html)

Right, my concern again with local minimizers is that they settle on a local minimum, and a global minimizer might be too expensive. I was just wondering what other options were out there. Seems like levenberg−marquardt is well−suited for least−squares type problems, but just want to be sure.

since the problem can be linearized in such a reasonable manner,
you can be rather sure to find the global minimizer by local methods if you take the result from this specialized linearization (not by Tayplor's theorem but by weighting!) as initial guess

but you need an initial guess

I could use the centroid of the points as a guess for the centre and the average distance from that centre for the radius.

well

multiply the summand by (sqrt(....) + radius)^2 getting

sum_{i=1 to N} ((x_i
−xmid)^2+(y_i−ymid)^2+(z_i−zmid)^2 − radius^2)^2 =min

simplify ...

sum_{i=1 to N} (a_i + b_i*xmid + c_i*ymid + d_i*zmid + t
)^2 = min

where
a_i = x_i^2+y_i^2+z_i^2 , b_i =−2*x_i , c_i = −2*y_i ,
d_i = −2*z_i
and t=xmid^2+ymid^2+zmid^2−radius^2

Sorry, I'm not following this step. What does this simplification gain me?

I used (distance + radius)^2 as weight, in the hope they all are almost the same, and this problem in xmid, ymid, zmid and t is a linear least squares problem, one call to svd does the job

this is now linear least sqaures in the four unknowns from which you also can
recover the radius

if you meant "inside the sphere" then it becomes a little bit harder:

Nope, I mean on the perimeter. Thanks though.

I misunderstood the problem, nevertheless the trick above (fit by sphere , initial guess by linearized problem using weighting) will help you
simply forget the third dimension.
as you wrote you have already the plane (by svd, this is orthogonal distance fitting and o.k.)
the following trick will give you the circle, but it is "suboptimal" since it is a fit on fit (plane already accepted):
the plane will be given to you in its so called
"Hesse−normal form"

a^t *[x;y;z] + d =0 a^t*a =1 (from the svd)

your data are [x_i;y_i;z_i] (matlab notation, a column)
the idea : transform this plane into a plane in (x,y,0) coordinates ,
apply the same transform to the data, forget the z−coordinate and fit a circle to the transformed (x_i,y_i)−data in the same manner as written above for 3D spheres.
how to obtain this?
well : d is the (negative) signed distance of the plane from [0;0;0];
hence add d*a from the plane resp your data:

a^t*([x;y;z]+d*a) =0 <=> a^t*([xt;yt;zt])=0 a plane through zero.

Next, reflect the plane properly such that in falls into the (xt;yt;0) −plane:
this can be done by transforming the normal a into [0;0;1]
hence this obtained by a householder reflector

U=I−2/(u^t*u)*u*u^t

where u=[a_1;a_2;sign(a_3)*(abs(a3)+1)] (since a has length 1 already)

apply this reflector also on the transformed data
[x_i;y_i;zI]+d*a
you obtain transformed data for which the z component now is the distance of your original points from the original fitted plane. forget the z−coordinate and fit the (x,y) coordinates by a circle (same method as on top, but in 2D)

since this is a two stage fit method, it will not be optimal, but reasonable if the noise is not to wild.
you then may take the result for a direct fit, but this is strongly nonlinear:
the parametrization of a general 2D−circle in 3D space:

[x;y;z] = [xmid;ymid;zmid] + U*r*[cos(phi);sin(phi);0]

where U is a Householder reflector (u unknown!) which plays the same role as before, whereas [xmid;ymid;zmid] plays the role of shifting the origin to zero.
hth
peter

My recollection (unchecked!) is that ODRPack has an example of fitting a circle to a small number of points that are on only part of its arc. It this is the case it provides a nicely packaged and complete solution. I do not believe it had the complication of the points being on a plane in 3 space. ODRPack is in netlib and comes with a expository paper. Orthogonal Distance Regression is a tool used by folks doing calibration curves. The ODRPack folks were at the the US National Bureau of Standards (NBS −> NIST some time ago) where calibration curves are all in a days work.

.