Subject: Re: ellipse fitting?
Posted by hradilv.nospam on Mon, 29 Apr 2002 16:19:10 GMT
View Forum Message <> Reply to Message

I did the same thing that you are doing a couple of months ago!

The best alternatives are
1- use mpfitellipse or Dr. Fanning's code
2- call lapack routines need to emulate the eig() function. Namely, I used the function dggev.f :

http://www.netlib.org/lapack/double/dggev.f

```
*  DGGEV computes for a pair of N-by-N real nonsymmetric matrices
(A,B)
*  the generalized eigenvalues, and optionally, the left and/or right
*  generalized eigenvectors.
*
*  A generalized eigenvalue for a pair of matrices (A,B) is a scalar
*  lambda or a ratio alpha/beta = lambda, such that A - lambda*B is
*  singular. It is usually represented as the pair (alpha,beta), as
*  there is a reasonable interpretation for beta=0, and even for both
*  being zero.
*
*  The right eigenvector v(j) corresponding to the eigenvalue
lambda(j)
*  of (A,B) satisfies
*
*               A * v(j) = lambda(j) * B * v(j).
*
*  The left eigenvector u(j) corresponding to the eigenvalue lambda(j)
*  of (A,B) satisfies
*
*               u(j)**H * A  = lambda(j) * u(j)**H * B .
*
*  where u(j)**H is the conjugate-transpose of u(j).
```

On Sat, 27 Apr 2002 21:52:48 +0800, "tom" <tom2959@21cn.com> wrote:

> I found a matlab function for ellipse, but it is not easy for me translate
> to IDl. For example,
>
> % Solve eigensystem
>   [gevec, geval] = eig(S,C);
>
> are there any function like eig(S,C) in IDL?
>
> The matlab for ellips fitting is as following, who have a idl version?

```
>
>
>
>  ------------------------------------------------------------ ----------------
> ----------------
>
> function a = fitellipse(X,Y)
>
> % FITELLIPSE  Least-squares fit of ellipse to 2D points.
> %       A = FITELLIPSE(X,Y) returns the parameters of the best-fit
> %       ellipse to 2D points (X,Y).
> %       The returned vector A contains the center, radii, and orientation
> %       of the ellipse, stored as (Cx, Cy, Rx, Ry, theta_radians)
> %
> % Authors: Andrew Fitzgibbon, Maurizio Pilu, Bob Fisher
> % Reference: "Direct Least Squares Fitting of Ellipses", IEEE T-PAMI, 1999
> %
> % This is a more bulletproof version than that in the paper, incorporating
> % scaling to reduce roundoff error, correction of behaviour when the input
> % data are on a perfect hyperbola, and returns the geometric parameters
> % of the ellipse, rather than the coefficients of the quadratic form.
> %
> %  Example:  Run fitellipse without any arguments to get a demo
> if nargin == 0
>   % Create an ellipse
>   t = linspace(0,2);
>
>   Rx = 300
>   Ry = 200
>   Cx = 250
>   Cy = 150
>   Rotation = .4 % Radians
>
>   x = Rx * cos(t);
>   y = Ry * sin(t);
>   nx = x*cos(Rotation)-y*sin(Rotation) + Cx;
>   ny = x*sin(Rotation)+y*cos(Rotation) + Cy;
>   % Draw it
>   plot(nx,ny,'o');
>   % Fit it
>   fitellipse(nx,ny)
>   % Note it returns (Rotation - pi/2) and swapped radii, this is fine.
>   return
> end
>
> % normalize data
> mx = mean(X);
> my = mean(Y);
```

```
> sx = (max(X)-min(X))/2;
> sy = (max(Y)-min(Y))/2;
>
> x = (X-mx)/sx;
> y = (Y-my)/sy;
>
> % Force to column vectors
> x = x(:);
> y = y(:);
>
> % Build design matrix
> D = [ x.*x  x.*y  y.*y  x  y  ones(size(x)) ];
>
> % Build scatter matrix
> S = D'*D;
>
> % Build 6x6 constraint matrix
> C(6,6) = 0; C(1,3) = -2; C(2,2) = 1; C(3,1) = -2;
>
> % Solve eigensystem
> [gevec, geval] = eig(S,C);
>
> % Find the negative eigenvalue
> I = find(real(diag(geval)) < 1e-8 & ~isinf(diag(geval)));
>
> % Extract eigenvector corresponding to negative eigenvalue
> A = real(gevec(:,I));
>
> % unnormalize
> par = [
>   A(1)*sy*sy,   ...
>       A(2)*sx*sy,   ...
>       A(3)*sx*sx,   ...
>       -2*A(1)*sy*sy*mx - A(2)*sx*sy*my + A(4)*sx*sy*sy,   ...
>       -A(2)*sx*sy*mx - 2*A(3)*sx*sx*my + A(5)*sx*sx*sy,   ...
>       A(1)*sy*sy*mx*mx + A(2)*sx*sy*mx*my + A(3)*sx*sx*my*my   ...
>       - A(4)*sx*sy*sy*mx - A(5)*sx*sx*sy*my   ...
>       + A(6)*sx*sx*sy*sy   ...
>       ]';
>
> % Convert to geometric radii, and centers
>
> thetarad = 0.5*atan2(par(2),par(1) - par(3));
> cost = cos(thetarad);
> sint = sin(thetarad);
> sin_squared = sint.*sint;
> cos_squared = cost.*cost;
> cos_sin = sint .* cost;
```

```
>
> Ao = par(6);
> Au =   par(4) .* cost + par(5) .* sint;
> Av = - par(4) .* sint + par(5) .* cost;
> Auu = par(1) .* cos_squared + par(3) .* sin_squared + par(2) .* cos_sin;
> Avv = par(1) .* sin_squared + par(3) .* cos_squared - par(2) .* cos_sin;
>
> % ROTATED = [Ao Au Av Auu Avv]
>
> tuCentre = - Au./(2.*Auu);
> tvCentre = - Av./(2.*Avv);
> wCentre = Ao - Auu.*tuCentre.*tuCentre - Avv.*tvCentre.*tvCentre;
>
> uCentre = tuCentre .* cost - tvCentre .* sint;
> vCentre = tuCentre .* sint + tvCentre .* cost;
>
> Ru = -wCentre./Auu;
> Rv = -wCentre./Avv;
>
> Ru = sqrt(abs(Ru)).*sign(Ru);
> Rv = sqrt(abs(Rv)).*sign(Rv);
>
> a = [uCentre, vCentre, Ru, Rv, thetarad];
>
>
>
>
```