Error Ellipse Computation

Introduction

This paper is intended to provide both an intuitive and mathematical insight into error ellipses, and ellipsoids.
An “error ellipse” is often used as a visual aid or performance measure to depict the accuracy or performance of an estimator, or other stochastic system. It implies that there are two (or more) variables involved, and that they are normally (Gaussian) distributed with known covariance.

It really is an ellipse.

Consider a 2-D zero-mean[1] random variable, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Since the r.v. is Guassian with covariance matrix $\mathbf{C}_\mathbf{x} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$, it has a probability density function (with $n = 2$),

$$p(\mathbf{x}) = \frac{1}{(2\pi)^n \sqrt{|\mathbf{C}_\mathbf{x}|}} e^{-\frac{1}{2}\mathbf{x}^T \mathbf{C}_\mathbf{x}^{-1}\mathbf{x}} \tag{1.1}$$

Note that

$$\mathbf{C}_\mathbf{x}^{-1} = \frac{1}{1-\rho^2}\begin{bmatrix} \frac{1}{\sigma_1^2} & \frac{-\rho}{\sigma_1\sigma_2} \\ \frac{-\rho}{\sigma_1\sigma_2} & \frac{1}{\sigma_2^2} \end{bmatrix}$$

Written out in terms of the two separate variables, we have,

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2\left(1-\rho^2\right)}\left[\frac{1}{\sigma_1^2}x_1^2 - \frac{2\rho}{\sigma_1\sigma_2}x_1x_2 + \frac{1}{\sigma_2^2}x_2^2 \right]\right) \tag{1.2}$$

While this form is cumbersome it serves to show that, for a constant value of $p^* = p(x_1, x_2)$, we have an ellipse, as shown below.

$$p^* = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2\left(1-\rho^2\right)}\left[\frac{1}{\sigma_1^2}x_1^2 - \frac{2\rho}{\sigma_1\sigma_2}x_1x_2 + \frac{1}{\sigma_2^2}x_2^2 \right]\right) \tag{1.3}$$

Which can be written as

$$k^2 = \frac{1}{\left(1-\rho^2\right)}\left[\frac{1}{\sigma_1^2}x_1^2 - \frac{2\rho}{\sigma_1\sigma_2}x_1x_2 + \frac{1}{\sigma_2^2}x_2^2 \right] \tag{1.4}$$

[1] If the random variance is not zero-mean, the mean, $\boldsymbol{\mu}$, can be removed by subtracting the mean from the variable.

Where $k^2 = -2\ln\left(p^* 2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}\right)$.

The selection of k determines the size of the ellipse. This will be discussed more, below.

A more practical way to describe the error ellipse is in terms of matrices. (1.4) is re-written as

$$k^2 = \mathbf{x}^T\mathbf{C}^{-1}\mathbf{x} \tag{1.5}$$

For a fixed value of k, the locus of points, $\left\{\mathbf{x} : k^2 = \mathbf{x}^T\mathbf{C}^{-1}\mathbf{x}\right\}$, which solves the equation is the error ellipse.

Stretching the circle

Recall the properties of multivariate transformation. If $\mathbf{x}$ is a random vector (of n variables) with covariance $\mathbf{C}$, and $\mathbf{A}$ is an $n\times n$ matrix, then $\mathbf{y} = \mathbf{Ax}$ is a random vector (of n variables) with covariance $\mathbf{ACA}^T$. (Assume $\mathbf{A}$ is invertible.)

Let's say that we desire $\mathbf{y}$'s components to be independent standard normal, e.g. $\mathbf{ACA}^T = \mathbf{I}$. What should $\mathbf{A}$ be set to? For this, we turn to Eigendecompsition[2], which allows any square matrix $\mathbf{C}$ to be expressed as

$$\mathbf{C} = \mathbf{VDV}^{-1} \tag{1.6}$$

Where $\mathbf{V}$ is an orthonormal square matrix (of eigenvectors), and $\mathbf{D}$ is a diagonal matrix (of eigenvalues). We will use the shorthand $\mathbf{D}^{\frac{1}{2}}$ to refer to a diagonal matrix contains the square roots of the eigenvalues such that $\mathbf{D}^{\frac{1}{2}}\mathbf{D}^{\frac{1}{2}} = \mathbf{D}$.

The we have $\mathbf{AVD}^{\frac{1}{2}}\mathbf{D}^{\frac{1}{2}}\mathbf{V}^{-1}\mathbf{A}^T = \mathbf{I}$ which is satisfied if $\mathbf{D}^{\frac{1}{2}}\mathbf{V}^{-1}\mathbf{A}^T = \mathbf{I}$ (although other solutions may exist). Thus, setting

$$\mathbf{A} = \mathbf{D}^{-\frac{1}{2}}\mathbf{V} \tag{1.7}$$

allows $\mathbf{y} = \mathbf{Ax}$ to have a stand normal distribution, where $\mathbf{D}$ and $\mathbf{V}$ are the eigenvalues and eigenvectors of the covariance matrix, $\mathbf{C}$. Moreover, the inverse transform, $\mathbf{x} = \mathbf{A}^{-1}\mathbf{y}$, allows the original variables to be restored. ($\mathbf{A}^{-1} = \mathbf{D}^{\frac{1}{2}}\mathbf{V}^T$, using the property that $\mathbf{V}^{-1} = \mathbf{V}^T$.)

Working with $\mathbf{y}$ is much easier that working with $\mathbf{x}$ because is has standard normal distribution; the error ellipse is a circle, and so on. The selection of k for a desired confidence interval, p, becomes easier too, and can be computed in few different ways. The probability that a point lies with in a circle of a given radius has a Rayleigh

[2] Any symmetric decomposition can be used. For example, Cholesky decomposition may be used instead of Eigendecompsition, since the covariance matrix is assume to be square, symmetric, and positive definite. In this case, $\mathbf{C} = \mathbf{U}^T\mathbf{U}$, and then $\mathbf{AU}^T\mathbf{UA}^T = \mathbf{I}$, and $\mathbf{A} = \left(\mathbf{U}^T\right)^{-1}$, and $\mathbf{A}^{-1} = \mathbf{U}^T$. Other possibilities include Takagi's factorization.

distribution, that is, the random variable $d = \|\mathbf{y}\| = \sqrt{y_1^2 + y_2^2}$ has a Rayleigh distribution. The square of the distance, d^2 has a chi-square order-2 distribution.

$$P(d < R) = P\left(d^2 < R^2\right) = p \tag{1.8}$$

We know the desired confidence interval, p, and that d^2 has a chi-square(2) distribution, R^2 can be determined by computed the quantile of the chi-square(2) distribution at R^2, or $R^2 = \alpha_{\chi_2^2}^{-1}(p)$. R then defines the radius of the circle defining the confidence interval, p.

If $\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = R\begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$ is a point on the circle of radius R at any arbitrary θ, then $\mathbf{b} = \mathbf{A}^{-1}\mathbf{a}$ lies on the error ellipse.

Summarizing, $k^2 = R^2 = \mathbf{x}^T\mathbf{C}^{-1}\mathbf{x} \sim \chi^2(n)$, where n is the number of dimensions (1 for an interval, 2 for an error ellipse, 3 for an ellipsoid). For various confidence regions, assign k as required:

Confidence Region, $\mathrm{P}\left(\mathbf{x}^T\mathbf{C}^{-1}\mathbf{x} < k^2\right)$	k (approx.) $n = 1$	k (approx.) $n = 2$	k (approx.) $n = 3$
19.87%			1.000
39.34%		1.000	
50.0%	0.674	1.177	1.538
68.27%	1.000		
90.0%	1.645	2.146	2.500
95.0%	1.960	2.448	2.795
99.0%	2.576	3.035	3.368
99.9%	3.290	3.717	4.033
99.99%	3.889	4.292	4.598
p	$\sqrt{\alpha_{\chi^2[1]}^{-1}(p)}$	$\sqrt{\alpha_{\chi^2[2]}^{-1}(p)}$	$\sqrt{\alpha_{\chi^2[3]}^{-1}(p)}$

Summary of the error ellipse generation

1. Given a covariance matrix $\mathbf{C}$ of a normal bi-variatiate random variable, compute the eigenvalues, $\mathbf{D}$, and eigenvectors, $\mathbf{V}$.
2. Compute the transform matrix, $\mathbf{A} = \mathbf{D}^{-\frac{1}{2}}\mathbf{V}$, and its inverse, $\mathbf{A}^{-1} = \mathbf{D}^{\frac{1}{2}}\mathbf{V}^T$.
3. For a set of angles, $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots\}$, compute some points around a circle, $\mathbf{a} = \begin{bmatrix} \cos(\boldsymbol{\theta}) \\ \sin(\boldsymbol{\theta}) \end{bmatrix}$.
4. Transform the points on the circle to points on the error ellipse, $\mathbf{b} = \mathbf{A}^{-1}\mathbf{a}$.
5. Plot the points of $\mathbf{b}$. If desired, offset by the mean ($\boldsymbol{\mu}$) of the variables, if it is not zero-mean.

Error ellipsoid

For three dimensions, the concept is the same, with few differences. The differences are:

1. $\mathbf{C}$ is a 3x3 covariance matrix, instead of 2x2.
2. The variable d^2 is distributed chi-square order-3 instead of order-2, which affects the computation of R^2, based on a desired confidence interval.
3. Choosing points on the unit sphere can be tricky; see Matlab function `ellipsoid` for more information.

Caveats

If the covariance matrix for a (normally distributed) data set is not known, it can be estimated, as the "sample covariance matrix." However, this is in itself a $n \times n$ random variable, having a Wishart Distribution. Further more, if the data is not zero-mean and the mean is unknown, it too can be estimated and removed. If either case, the error ellipse described in this paper becomes only an approximation, increasing in accuracy as the number of data points increases.

Relationship among ellipse properties

Consider the un-rotated ellipse,

$$\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} = k^2 .$$

Its companion matrix form is

$$\begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \end{bmatrix} = k^2 .$$

To remain consistent with prior notation, we let $\mathbf{D} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$, and therefore

$$\mathbf{D}^{\frac{1}{2}} = \begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix} .$$

Now consider a rotation of angle θ, and define the rotation matrix

$$\mathbf{V} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix},$$

Such that

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{V} \begin{bmatrix} x \\ y \end{bmatrix}.$$

Note that $\mathbf{V}$ is orthogonal, i.e. $\mathbf{V}^{-1} = \mathbf{V}^T$.
The rotated ellipse equation (in matrix form) is:

$$\begin{bmatrix} x \\ y \end{bmatrix}^T \mathbf{V}^T \mathbf{D}^{-1} \mathbf{V} \begin{bmatrix} x \\ y \end{bmatrix} = k^2 .$$

This provides a method of create covariance matrix having "major" and "minor" ellipse axes of σ_x and σ_y with a (counterclockwise) rotation angle of θ, by defining

$$\mathbf{T} = \mathbf{D}^{-\frac{1}{2}}\mathbf{V} = \begin{bmatrix} \sigma_x^{-1} & 0 \\ 0 & \sigma_y^{-1} \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} \dfrac{\cos\theta}{\sigma_x} & \dfrac{\sin\theta}{\sigma_x} \\ -\dfrac{\sin\theta}{\sigma_y} & \dfrac{\cos\theta}{\sigma_y} \end{bmatrix}$$

And

$$\mathbf{C}^{-1} = \mathbf{T}^T\mathbf{T}.$$

This is easily invertible, defining

$$\mathbf{T}^{-1} = \mathbf{V}^T\mathbf{D}^{\frac{1}{2}} = \begin{bmatrix} \sigma_x \cos\theta & -\sigma_y \sin\theta \\ \sigma_x \sin\theta & \sigma_y \cos\theta \end{bmatrix}$$

And

$$\mathbf{C} = \mathbf{T}^{-1}\left(\mathbf{T}^{-1}\right)^T .$$

To test is a point (x, y) is inside the matrix, we apply the test

$$\begin{bmatrix} x \\ y \end{bmatrix}^T \mathbf{T}^{-1}\left(\mathbf{T}^{-1}\right)^T \begin{bmatrix} x \\ y \end{bmatrix} \le k^2$$

If we define

$$\begin{bmatrix} u \\ v \end{bmatrix}^T = \begin{bmatrix} x \\ y \end{bmatrix}^T \mathbf{T}^{-1} = \begin{bmatrix} \dfrac{x\cos\theta + y\sigma_x \sin\theta}{\sigma_x} \\ \dfrac{-x\sin\theta + y\cos\theta}{\sigma_y} \end{bmatrix}$$

Then

$$\begin{bmatrix} u \\ v \end{bmatrix}^T \begin{bmatrix} u \\ v \end{bmatrix} \le k^2$$

$$\left(\frac{x\cos(\theta) + y\sin(\theta)}{\sigma_x}\right)^2 + \left(\frac{-x\sin(\theta) + y\cos(\theta)}{\sigma_y}\right)^2 \le k^2$$

Matlab code

```
function h=error_ellipse(varargin)
% ERROR_ELLIPSE - plot an error ellipse, or ellipsoid, defining confidence
region
%    ERROR_ELLIPSE(C22) - Given a 2x2 covariance matrix, plot the
%    associated error ellipse, at the origin. It returns a graphics handle
%    of the ellipse that was drawn.
%
%    ERROR_ELLIPSE(C33) - Given a 3x3 covariance matrix, plot the
%    associated error ellipsoid, at the origin, as well as its projections
%    onto the three axes. Returns a vector of 4 graphics handles, for the
%    three ellipses (in the X-Y, Y-Z, and Z-X planes, respectively) and for
%    the ellipsoid.
%
%    ERROR_ELLIPSE(C,MU) - Plot the ellipse, or ellipsoid, centered at MU,
%    a vector whose length should match that of C (which is 2x2 or 3x3).
%
%    ERROR_ELLIPSE(...,'Property1',Value1,'Name2',Value2,...) sets the
%    values of specified properties, including:
%      'C' - Alternate method of specifying the covariance matrix
%      'mu' - Alternate method of specifying the ellipse (-oid) center
%      'conf' - A value betwen 0 and 1 specifying the confidence interval.
%        the default is 0.5 which is the 50% error ellipse.
%      'scale' - Allow the plot the be scaled to difference units.
%      'style' - A plotting style used to format ellipses.
%      'clip' - specifies a clipping radius. Portions of the ellipse, -oid,
%        outside the radius will not be shown.
%
%    NOTES: C must be positive definite for this function to work properly.

default_properties = struct(...
  'C', [], ... % The covaraince matrix (required)
  'mu', [], ... % Center of ellipse (optional)
  'conf', 0.5, ... % Percent confidence/100
  'scale', 1, ... % Scale factor, e.g. 1e-3 to plot m as km
  'style', '', ...  % Plot style
  'clip', inf); % Clipping radius

if length(varargin) >= 1 & isnumeric(varargin{1})
  default_properties.C = varargin{1};
  varargin(1) = [];
end

if length(varargin) >= 1 & isnumeric(varargin{1})
  default_properties.mu = varargin{1};
  varargin(1) = [];
end

if length(varargin) >= 1 & isnumeric(varargin{1})
  default_properties.conf = varargin{1};
  varargin(1) = [];
end

if length(varargin) >= 1 & isnumeric(varargin{1})
  default_properties.scale = varargin{1};
  varargin(1) = [];
end

if length(varargin) >= 1 & ~ischar(varargin{1})
  error('Invalid parameter/value pair arguments.')
end

prop = getopt(default_properties, varargin{:});
C = prop.C;

if isempty(prop.mu)
  mu = zeros(length(C),1);
else
  mu = prop.mu;
end
```

```
conf = prop.conf;
scale = prop.scale;
style = prop.style;

if conf <= 0 | conf >= 1
  error('conf parameter must be in range 0 to 1, exclusive')
end

[r,c] = size(C);
if r ~= c | (r ~= 2 & r ~= 3)
  error(['Don''t know what to do with ',num2str(r),'x',num2str(c),' matrix'])
end

x0=mu(1);
y0=mu(2);

% Compute quantile for the desired percentile
k = sqrt(qchisq(conf,r)); % r is the number of dimensions (degrees of
freedom)

hold_state = get(gca,'nextplot');

if r==3 & c==3
  z0=mu(3);

  % Make the matrix has positive eigenvalues - else it's not a valid
covariance matrix!
  if any(eig(C) <=0)
    error('The covariance matrix must be positive definite (it has non-
positive eigenvalues)')
  end

  % C is 3x3; extract the 2x2 matricies, and plot the associated error
  % ellipses. They are drawn in space, around the ellipsoid; it may be
  % preferable to draw them on the axes.
  Cxy = C(1:2,1:2);
  Cyz = C(2:3,2:3);
  Czx = C([3 1],[3 1]);

  [x,y,z] = getpoints(Cxy,prop.clip);
  h1=plot3(x0+k*x,y0+k*y,z0+k*z,prop.style);hold on
  [y,z,x] = getpoints(Cyz,prop.clip);
  h2=plot3(x0+k*x,y0+k*y,z0+k*z,prop.style);hold on
  [z,x,y] = getpoints(Czx,prop.clip);
  h3=plot3(x0+k*x,y0+k*y,z0+k*z,prop.style);hold on


  [eigvec,eigval] = eig(C);

  [X,Y,Z] = ellipsoid(0,0,0,1,1,1);
  XYZ = [X(:),Y(:),Z(:)]*sqrt(eigval)*eigvec';

  X(:) = scale*(k*XYZ(:,1)+x0);
  Y(:) = scale*(k*XYZ(:,2)+y0);
  Z(:) = scale*(k*XYZ(:,3)+z0);
  h4=surf(X,Y,Z);
  colormap gray
  alpha(0.3)
  camlight
  if nargout
    h=[h1 h2 h3 h4];
  end
elseif r==2 & c==2
  % Make the matrix has positive eigenvalues - else it's not a valid
covariance matrix!
  if any(eig(C) <=0)
    error('The covariance matrix must be positive definite (it has non-
positive eigenvalues)')
  end

  [x,y,z] = getpoints(C,prop.clip);
  h1=plot(scale*(x0+k*x),scale*(y0+k*y),prop.style);
  set(h1,'zdata',z+1)
  if nargout
    h=h1;
  end
else
  error('C (covaraince matrix) must be specified as a 2x2 or 3x3 matrix)')
end
%axis equal

set(gca,'nextplot',hold_state);

%----------------------------------------------------------------
% getpoints - Generate x and y points that define an ellipse, given a 2x2
%   covariance matrix, C. z, if requested, is all zeros with same shape as
%   x and y.
function [x,y,z] = getpoints(C,clipping_radius)

n=100; % Number of points around ellipse
p=0:pi/n:2*pi; % angles around a circle

[eigvec,eigval] = eig(C); % Compute eigen-stuff
xy = [cos(p'),sin(p')] * sqrt(eigval) * eigvec'; % Transformation
x = xy(:,1);
y = xy(:,2);
z = zeros(size(x));

% Clip data to a bounding radius
if nargin >= 2
  r = sqrt(sum(xy.^2,2)); % Euclidian distance (distance from center)
  x(r > clipping_radius) = nan;
  y(r > clipping_radius) = nan;
  z(r > clipping_radius) = nan;
end

%----------------------------------------------------------------
function x=qchisq(P,n)
```

```
% QCHISQ(P,N) - quantile of the chi-square distribution.
if nargin<2
  n=1;
end

s0 = P==0;
s1 = P==1;
s = P>0 & P<1;
x = 0.5*ones(size(P));
x(s0) = -inf;
x(s1) = inf;
x(~(s0|s1|s))=nan;

for ii=1:14
  dx = -(pchisq(x(s),n)-P(s))./dchisq(x(s),n);
  x(s) = x(s)+dx;
  if all(abs(dx) < 1e-6)
    break;
  end
end

%---------------------------------------------------------------
function F=pchisq(x,n)
% PCHISQ(X,N) - Probability function of the chi-square distribution.
if nargin<2
  n=1;
end
F=zeros(size(x));

if rem(n,2) == 0
  s = x>0;
  k = 0;
  for jj = 0:n/2-1;
    k = k + (x(s)/2).^jj/factorial(jj);
  end
  F(s) = 1-exp(-x(s)/2).*k;
else
  for ii=1:numel(x)
    if x(ii) > 0
      F(ii) = quadl(@dchisq,0,x(ii),1e-6,0,n);
    else
      F(ii) = 0;
    end
  end
end

%---------------------------------------------------------------
function f=dchisq(x,n)
% DCHISQ(X,N) - Density function of the chi-square distribution.
if nargin<2
  n=1;
end
f=zeros(size(x));
s = x>=0;
f(s) = x(s).^(n/2-1).*exp(-x(s)/2)./(2^(n/2)*gamma(n/2));
```

```
%----------------------------------------------------------------
function properties = getopt(properties,varargin)
%GETOPT - Process paired optional arguments as 'prop1',val1,'prop2',val2,...
%
%   getopt(properties,varargin) returns a modified properties structure,
%   given an initial properties structure, and a list of paired arguments.
%   Each argumnet pair should be of the form property_name,val where
%   property_name is the name of one of the field in properties, and val is
%   the value to be assigned to that structure field.
%
%   No validation of the values is performed.
%
% EXAMPLE:
%   properties =
struct('zoom',1.0,'aspect',1.0,'gamma',1.0,'file',[],'bg',[]);
%   properties = getopt(properties,'aspect',0.76,'file','mydata.dat')
% would return:
%   properties =
%         zoom: 1
%       aspect: 0.7600
%        gamma: 1
%         file: 'mydata.dat'
%           bg: []
%
% Typical usage in a function:
%   properties = getopt(properties,varargin{:})

% Process the properties (optional input arguments)
prop_names = fieldnames(properties);
TargetField = [];
for ii=1:length(varargin)
  arg = varargin{ii};
  if isempty(TargetField)
    if ~ischar(arg)
      error('Propery names must be character strings');
    end
    f = find(strcmp(prop_names, arg));
    if length(f) == 0
      error('%s ',['invalid property ''',arg,'''; must be one
of:'],prop_names{:});
    end
    TargetField = arg;
  else
    % properties.(TargetField) = arg; % Ver 6.5 and later only
    properties = setfield(properties, TargetField, arg); % Ver 6.1 friendly
    TargetField = '';
  end
end
if ~isempty(TargetField)
  error('Property names and values must be specified in pairs.');
end
```