N-dimensional Ellipse Fitting with Weighting

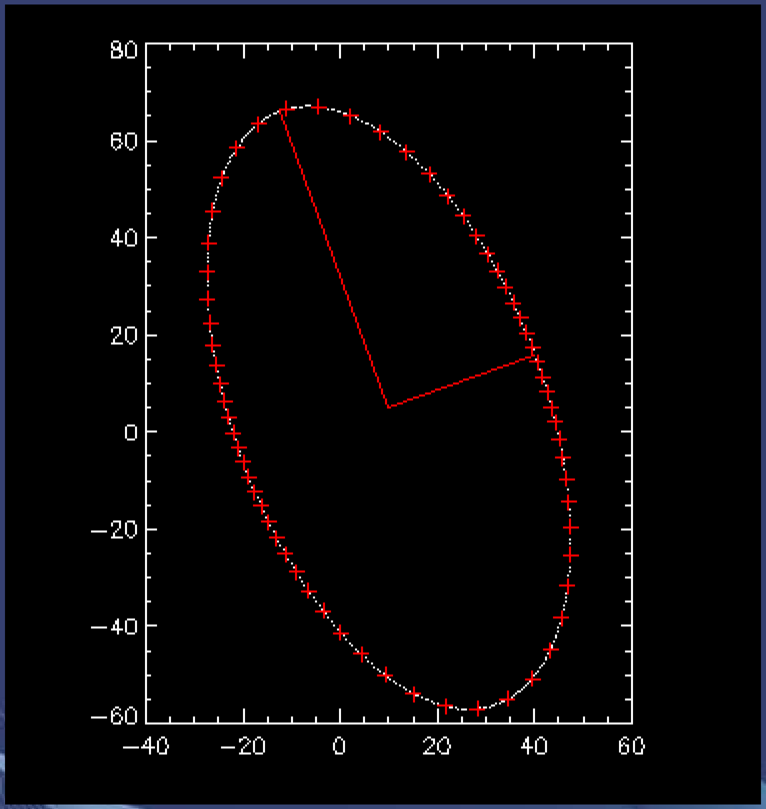
- Very similar to David Fanning's fit_ellipse routine that he first wrote in 2002 and updated in August 2008
- Uses the method of moments and the quadratic form of the ellipse/ellipsoid equation
  - Let a_i = amplitude at i^{th} point
  - $\underline{X}_i$ = vector to the i^{th} point
  - $\underline{X}_c$ = vector to the (weighted) center
  - Then $\underline{X}_c = \sum a_i \underline{X}_i / \sum a_i$ (First weighted moment)
  - Second moment is $P = \sum a_i \underline{dx}_i^T \underline{dx}_i / \sum a_i$ where $\underline{dx}_i = \underline{X}_i - \underline{X}_c$
- As you might expect if we find the eigenvectors and eigenvalues of P we can get the ellipse axes and rotation angle
- But take it one step further first

Ellipse fitting (cont)

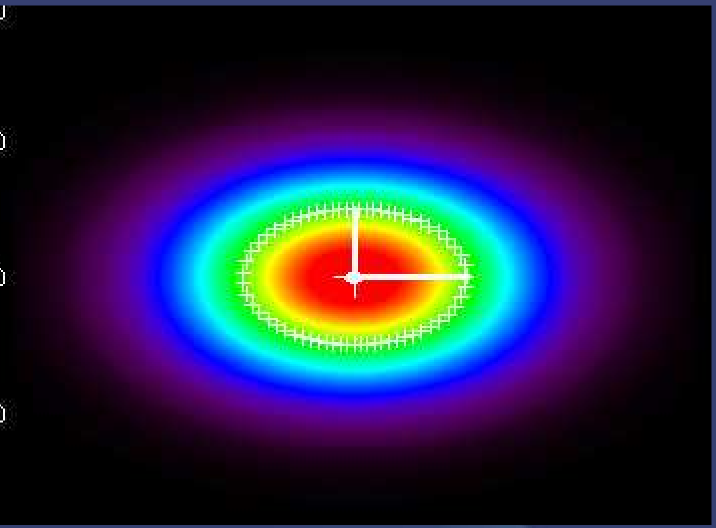
- Quadratic form of the ellipse equation is
  - $|\underline{X} - \underline{X}_c|^T P^{-1} | \underline{X} - \underline{X}_c | = k^2$
    - Independent of coordinate system
    - True for all dimensions
- If the data being fitted is gaussian then k is the sigma factor
  - i.e k=1 is 1 sigma, k=2 is 2 sigma etc.
- If the data is not gaussian then k is still useful
  - Calculate all of the k values from the equation above
  - Using the max ensures that you will encompass all of the points. Picking a value of k that is 90% of the max will create an ellipse that encompasses 90% of the points and so on

Ellipse fitting (cont)

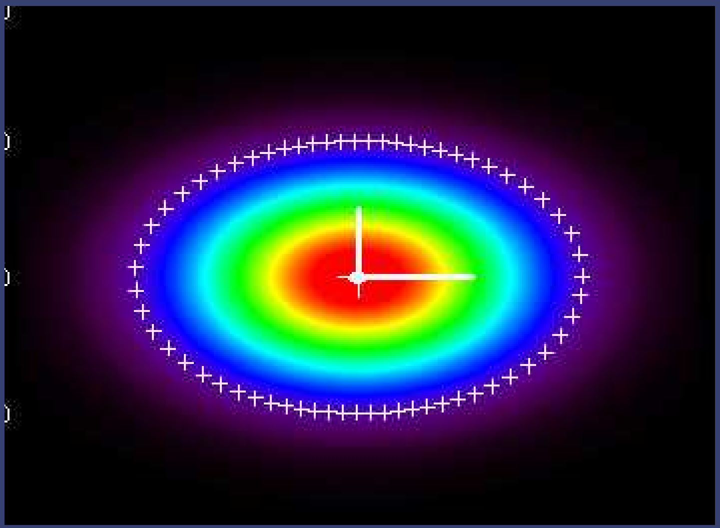
- Once you have chosen a k you can draw the ellipse
- In 2D we let $\underline{X} = r\,\underline{u}$ where $\underline{u} = [\cos(\theta), \sin(\theta)]$ then
  - $r^2 |\,\underline{u}\,|^T P^{-1} |\,\underline{u}\,| = k^2$
  - Or $r = k / (|\,\underline{u}\,|^T P^{-1} |\,\underline{u}\,|)^{1/2}$
  - So $X(\theta) = r\cos(\theta) + \underline{X}_c$ and $Y(\theta) = r\sin(\theta) + \underline{Y}_c$
- In 3D $\underline{u} = [\cos(\theta)*\cos(\varphi), \sin(\theta)*\cos(\varphi), \sin(\varphi)]$



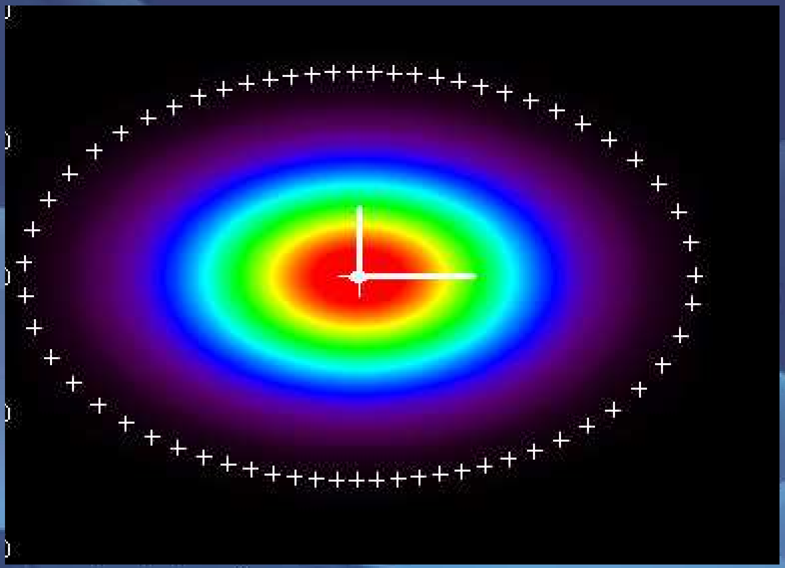
- No Weighting
- White dots are the original data
- Red +'s are the fitted
- Semi-major and semi-minor are from the eigenvectors
  - evals = eigenql(P, eigenvectors = evec)
  - semiMajor = sqrt(evals[0])
  - semiMinor = sqrt(evals[1])
  - angle = atan(evec[1,0],evec[0,0])



K = 1



K = 2



K = 3