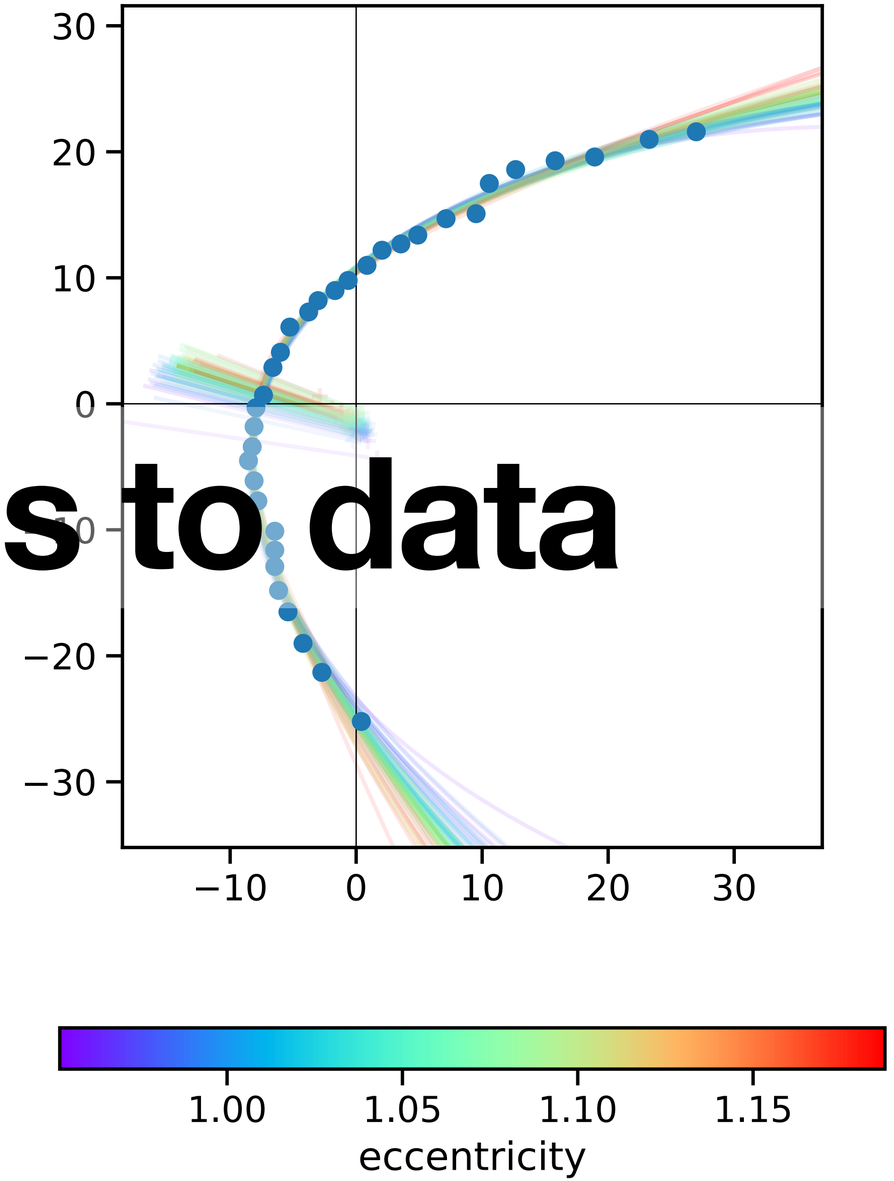
Fitting conic sections to data

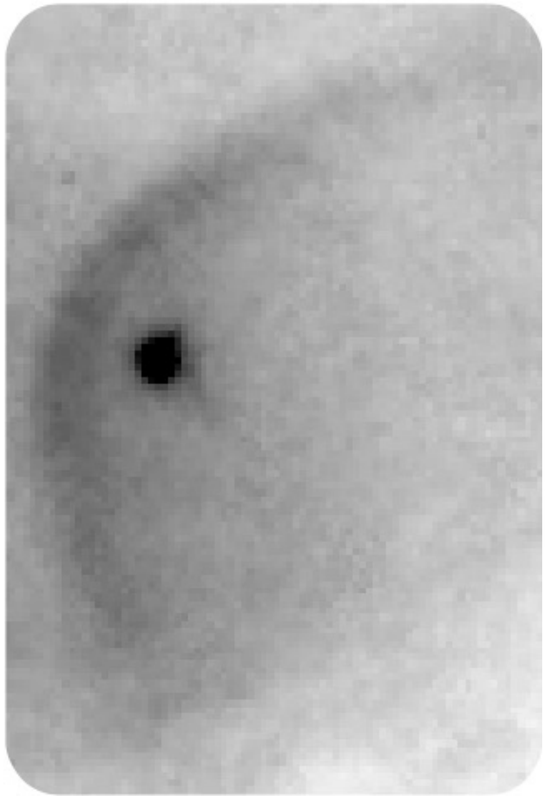
Parabolae, ellipses, and hyperbolae



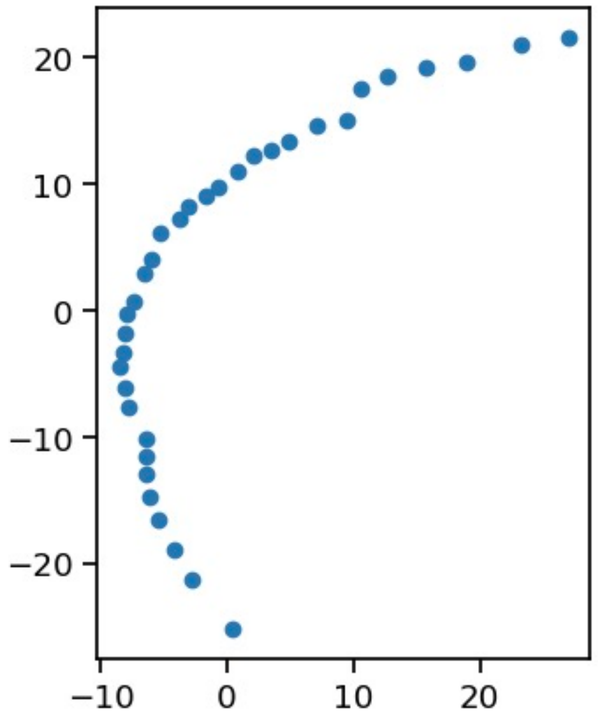
William Henney, IRyA, DAWGI 2024-02-28

What?

From this …

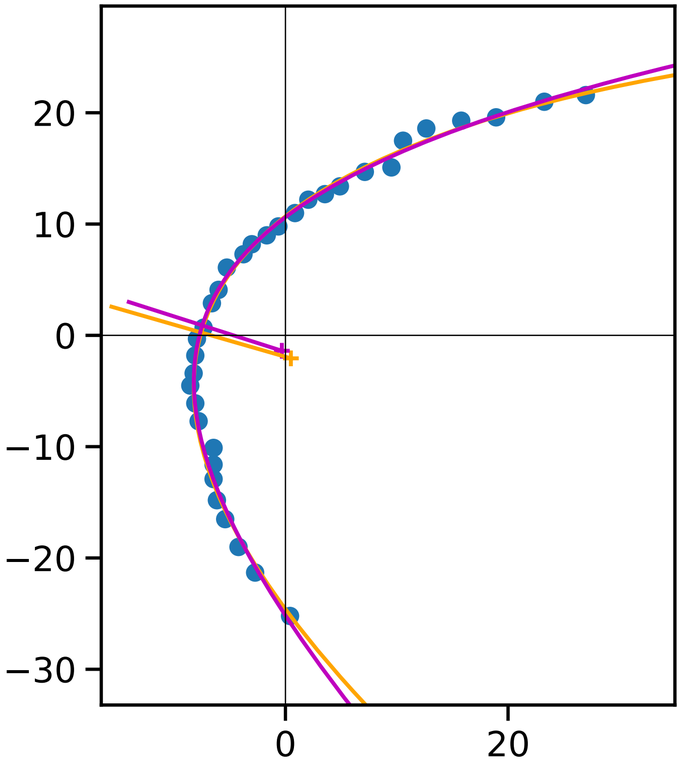


1. Tracing



… to this …

2. Fitting



… to this

Here I will concentrate on stage 2: Fitting

Why?

Curved emission arcs in H II regions

- Ionization fronts
- Bow shocks
- Bent jets

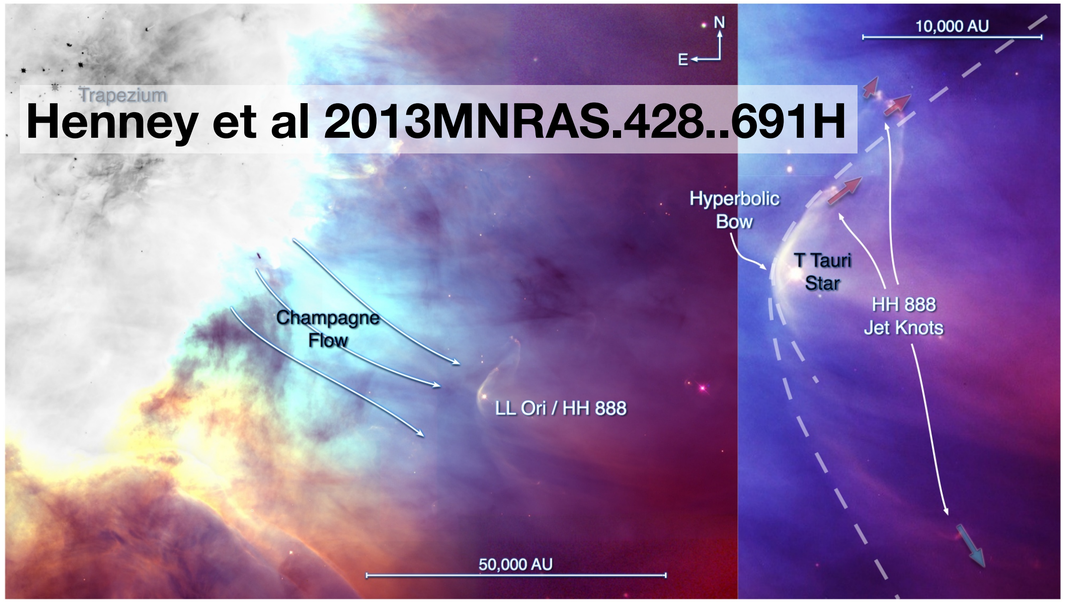
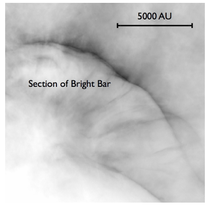
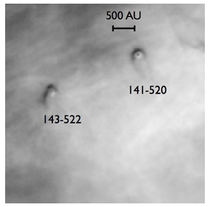
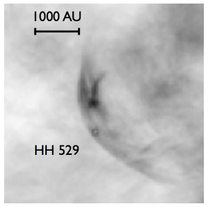
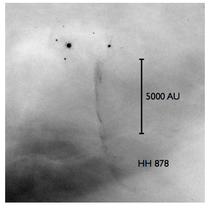
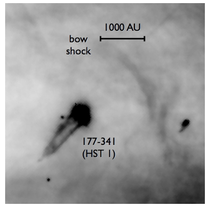
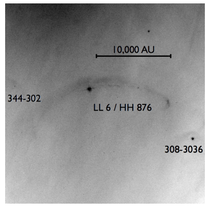


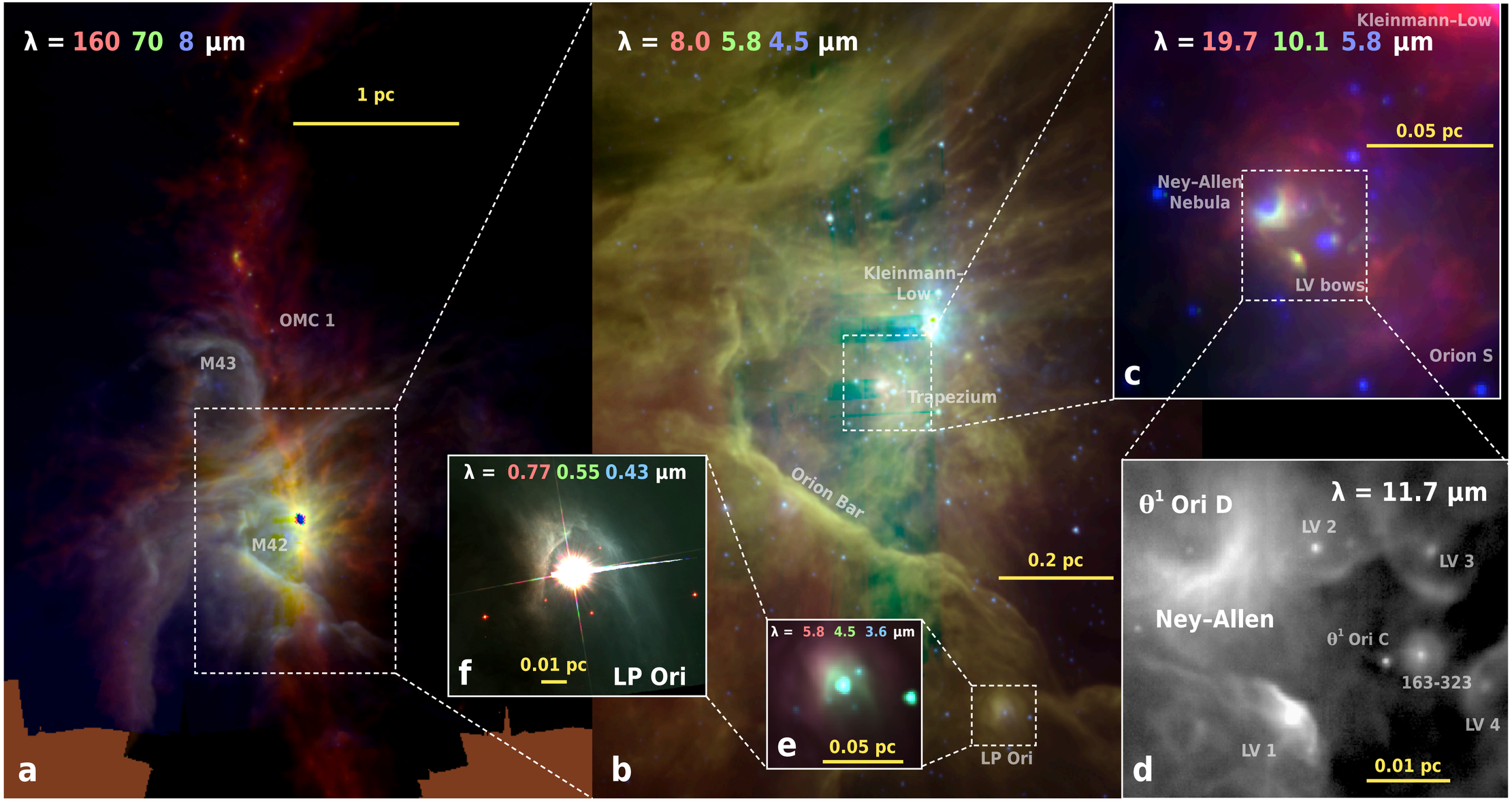
Table 1: Varieties of arc-like emission features in H II regions.

Phenomenon	Hα image	Physical Origin	Observational Signatures	Size, AU
Elephant trunks, pillars, globules, bright rims, bars		Both the clumpy structure of the surrounding molecular cloud and instabilities in the ionization front itself contribute to a rich variety of convex, D-critical ionization fronts, with associated ionized photoevaporation flows.	*Low proper motions and radial velocities. Sharp edge on inside of arc. Presence of low ionization species such as [O I], [S II] and [N II]. Ionization stratification.*	100 to 30,000
Proplyd crescents		Similar to above, but with a more complicated internal structure, involving a photodissociated flow from an embedded protoplanetary disk.	*Low proper motions and radial velocities. Suppression of low-critical-density lines. Larger linewidths (> 50 km s^{-1}) for high ionization lines such as [O III]. Central disk absorption.*	20 to 500
Herbig–Haro bow shocks		Working surface produced by a hypersonic collimated jet from a protostar or T Tauri star. May be due to interaction with ambient nebular gas or due to internal shocks in a variable-velocity jet.	*High proper motions perpendicular to arc, and high radial velocities (50–100 km s^{-1}). Sharp edge on outside of arc.*	1000 to 10,000
Bent jets		The jet beam may be deflected away from the center of the nebula due to ambient density gradients, ram pressure from a side wind, or the rocket effect on a photoevaporating neutral core.	*High proper motions parallel to arc, and high radial velocities (≈ 100 km s^{-1}). Knotty structure.*	3000 to 30,000
Proplyd wind-wind bow shocks		Due to the interaction of the ionized photoevaporation flow (30–40 km s^{-1}) from a proplyd crescent with the high-speed (> 1000 km s^{-1}) stellar wind from an O star.	*Low proper motions and radial velocities. Usually high ionization. Relatively closed bow shape. Emission confined to $\pm 45°$ from axis.*	500 to 2000
LL bow shocks		Due to the interaction of a relatively low speed ($\simeq 20$ km s^{-1}) champagne flow of ionized gas within the nebula and an outflow from a T Tauri star, which may also be a proplyd.	*Low proper motions and moderate radial velocities (≈ 15 km s^{-1}). Relatively broad shell. More open bow shape. Emission extends into conical wings.*	1000 to 10,000

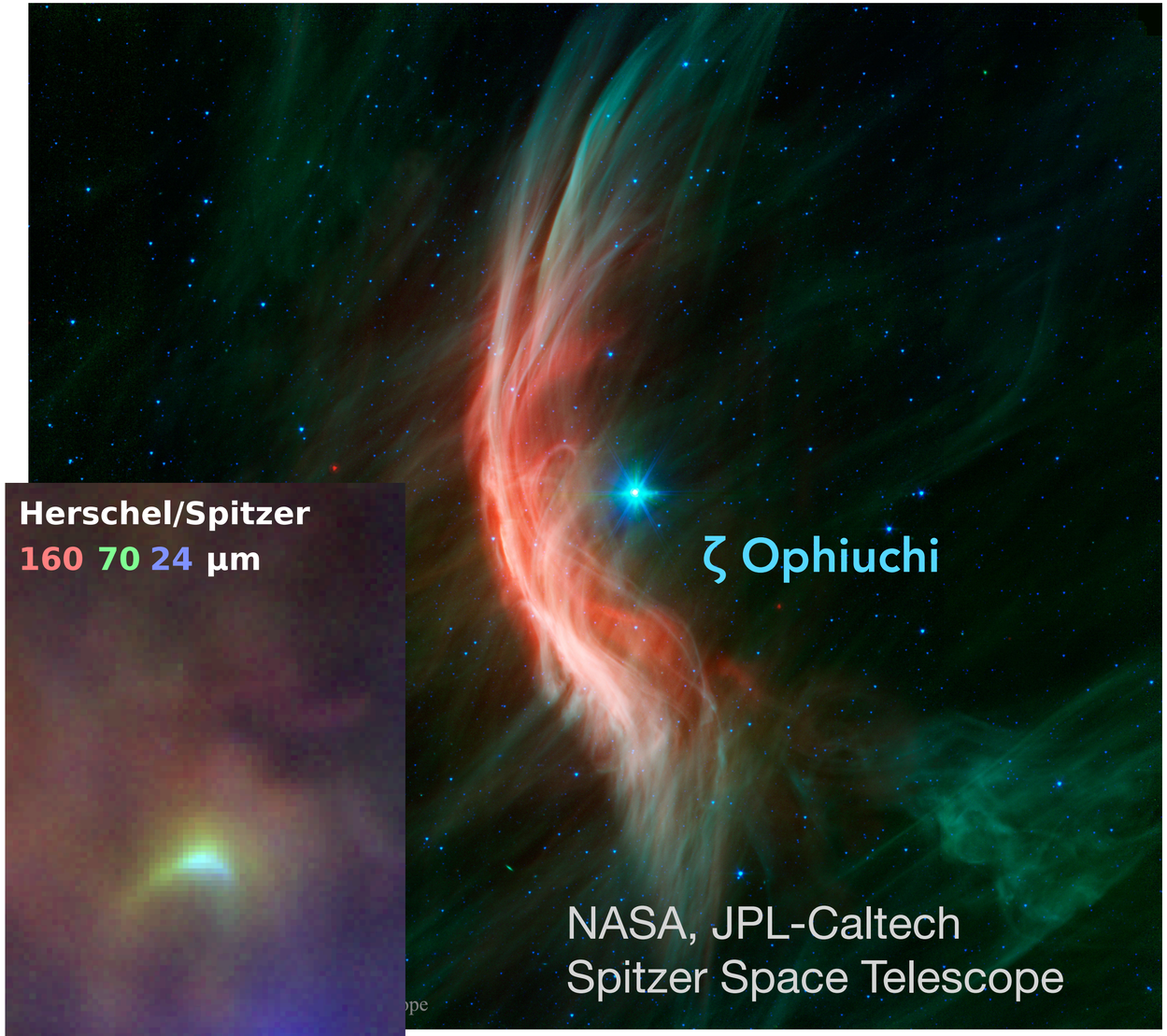
Henney 2013 SFN #249

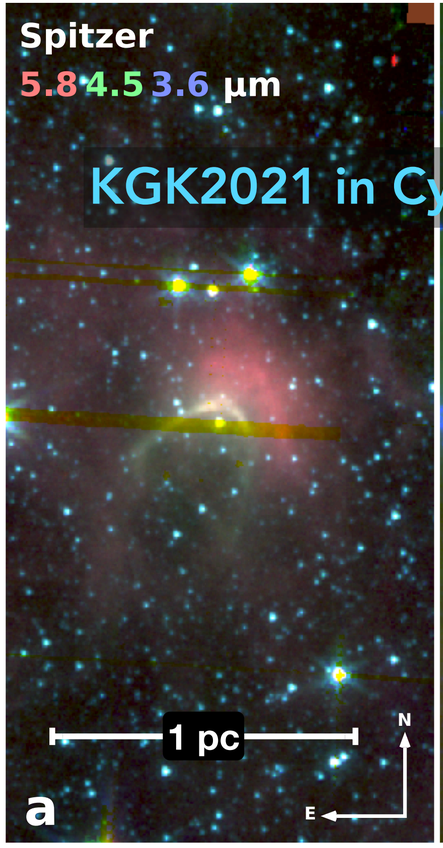
Also in infrared …

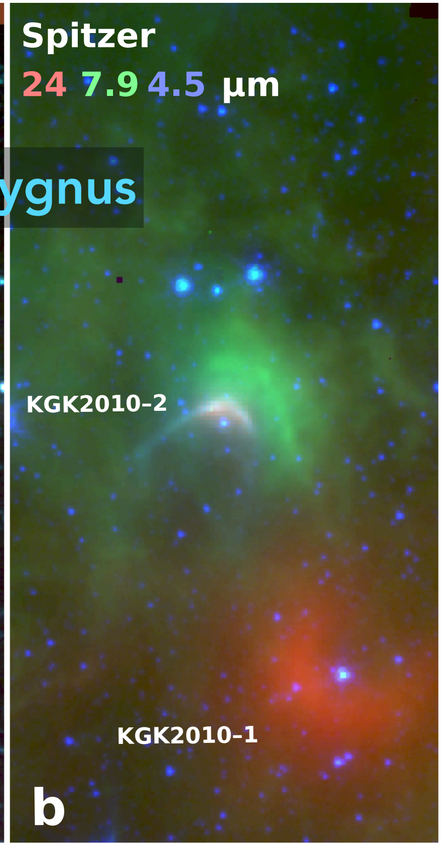
Henney & Arthur 2024 in prep (since 2019)

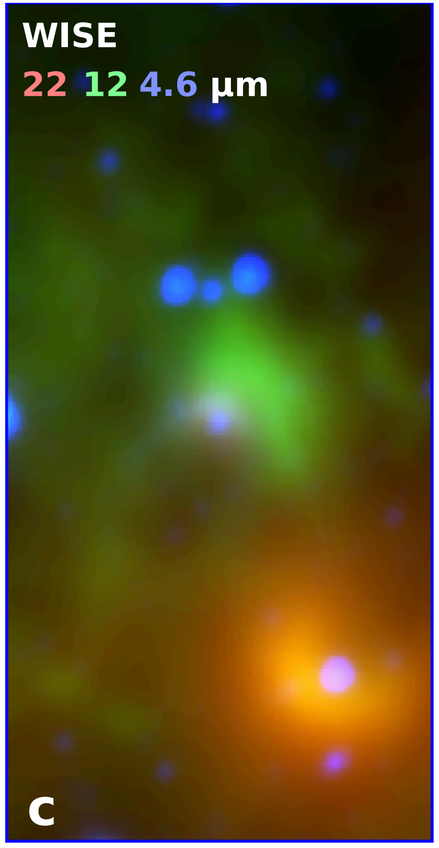


… and not just in Orion









Henney & Arthur
2019MNRAS.489.2142H

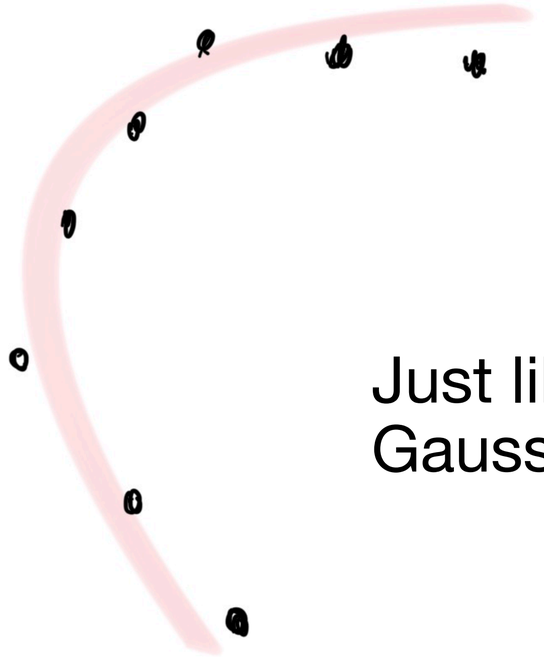
By fitting a conic section, we can measure:

- Position, (x, y)
- Size, r
- Shape, e
- Orientation, θ

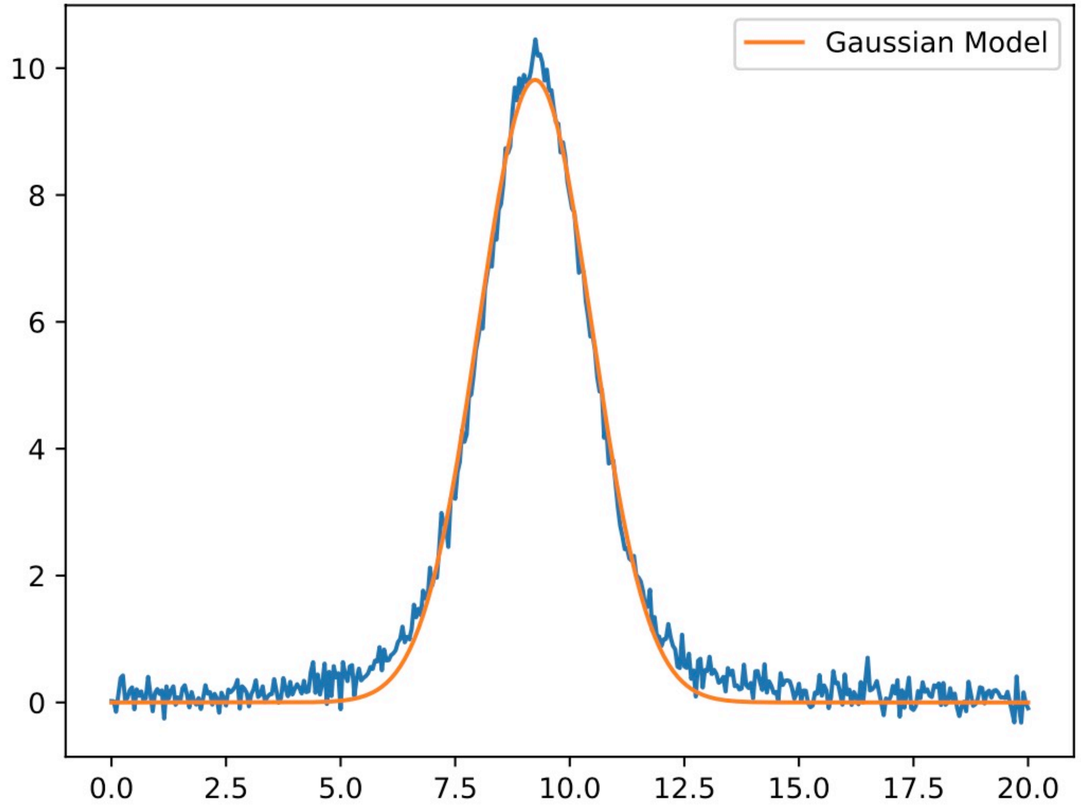
And we can even get error bars on those

But what if the shape is not a conic section?

Doesn’t matter – fitting a parabola is still useful



Just like the case of fitting Gaussians to line profiles



How?

Algebraic distance versus geometric distance

Different approaches to fitting conics (Zhang 1997)

- Conic sections are second order algebraic equations in *x* and *y*

3. Conic fitting problem

The problem is to fit a conic section to a set of n points $\{\mathbf{x}_i\} = \{(x_i, y_i)\}$ $(i = 1, \ldots, n)$. A conic can be described by the following equation:

$$Q(x, y) = Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0, \quad (1)$$

where A, B and C are not simultaneously zero. In practrue. A common practice is to directly minimize the *algebraic distance* $Q(x_i, y_i)$, i.e. to minimize the following function:

$$\mathscr{F} = \sum_{i=1}^{n} Q^2(x_i, y_i).$$

ELSEVIER

Image and Vision Computing 15 (1997) 59–76

COMPUTING

Parameter estimation techniques: a tutorial with application to conic fitting

Zhengyou Zhang*

INRIA, 2004 route des Lucioles, BP 93, F-06902 Sophia-Antipolis Cedex, France

Disadvantages of algebraic distance

$$Q(x, y) = A\,x^2 + B\,y^2 + C\,x\,y + 2\,D\,x + 2\,E\,y + F = 0$$

1. Need to avoid the trivial solution $A = B = C = D = E = F = 0$
   - Can use different normalizations: $F = 1$ or $A + C = 1$ with different trade-offs
2. High-curvature bias
   - Points in high-curvature regions contribute less to the fit
3. Not invariant under Euclidean transformations
   - Even simple translation produces complicated changes in A, B, C, …
   - No clear connection to the intuitive parameters: size, shape, orientation

Geometric Euclidean distance

Orthogonal distance from a point to a conic

The orthogonal distance d_i between a point $\mathbf{x}_i = (x_i, y_i)$ and a conic $Q(x, y)$ is the smallest Euclidean distance among all distances between $\mathbf{x}_i$ and points in the conic. The tangent at the corresponding point in the conic (denoted by $\mathbf{x}_t = (x_{ti}, y_{ti})$) is orthogonal to the line joining $\mathbf{x}_i$ and $\mathbf{x}_t$ (see Fig. 2). Given n points $\mathbf{x}_i$ $(i = 1, \ldots, n)$, the orthogonal distance fitting is to estimate the conic Q by minimizing the following function:

$$\mathscr{F}(\mathbf{p}) = \sum_{i=1}^{n} d_i^2. \qquad (8)$$

However, as the expression of d_i is very complicated (see below), an iterative optimization procedure must be carried out. Many techniques are readily available, includ-

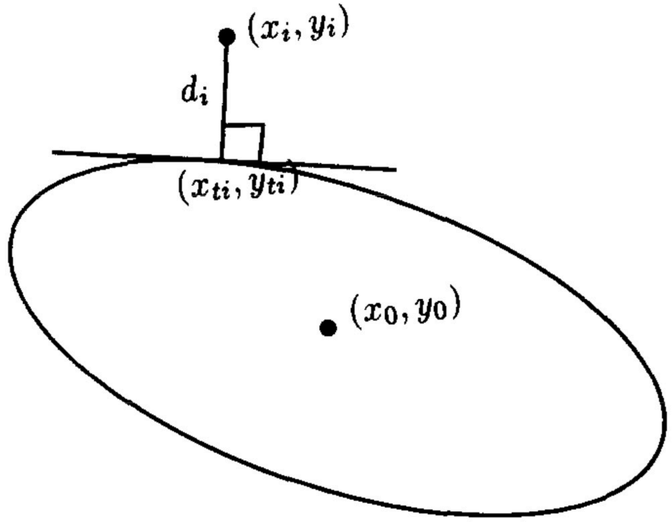


Fig. 2. Orthogonal distance of a point (x_i, y_i) to a conic. Point (x_{ti}, y_{ti}) is the point on the conic which is closest to point (x_i, y_i).

Zhang 1997 again

Isn’t there a simpler way?

Yes there is a simpler way

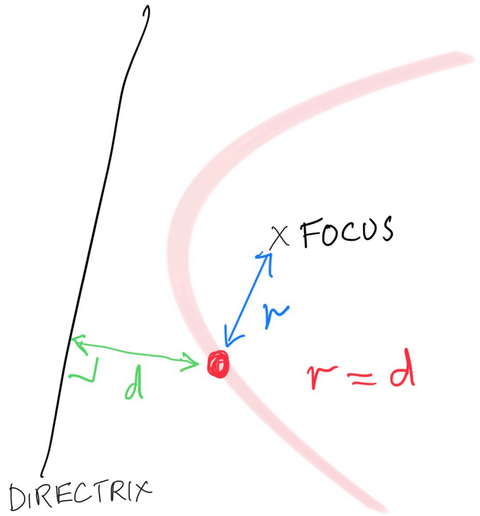
If we go back in time nearly 1800 years

All info from Wikipedia

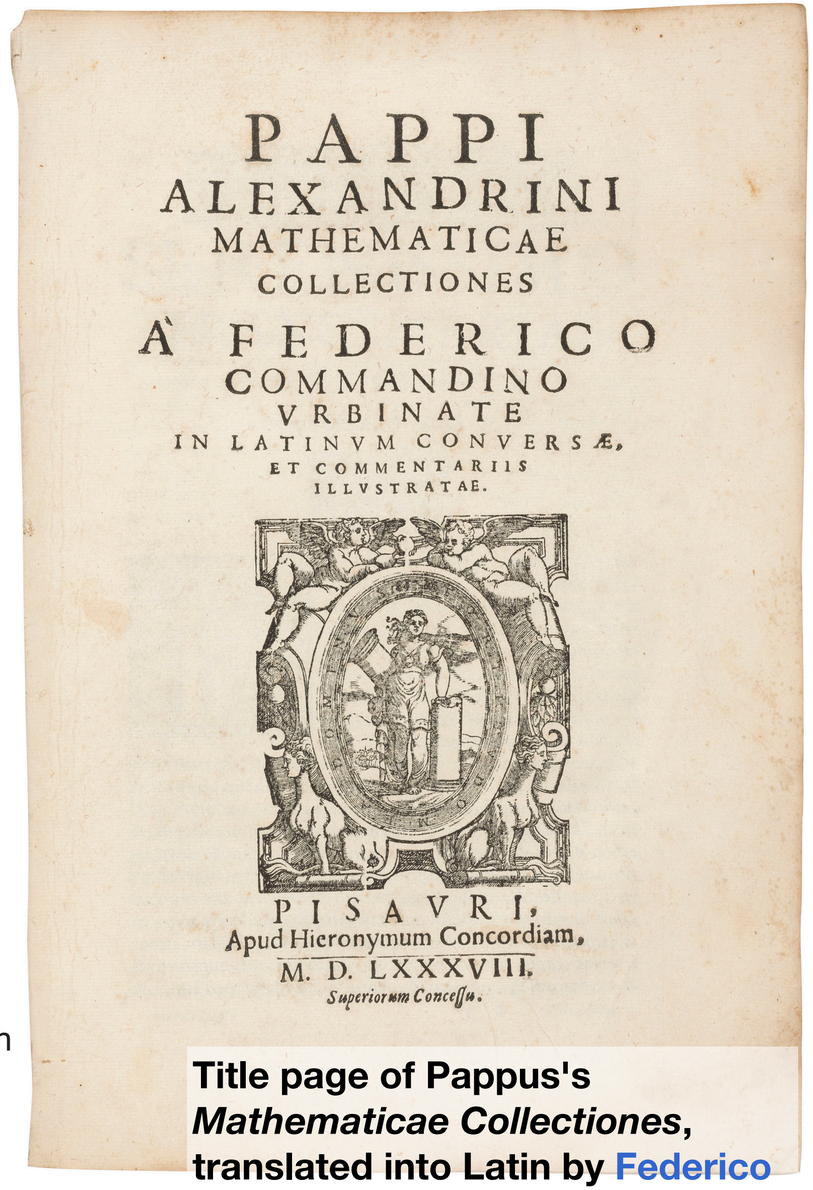
A parabola is a set of points, such that for any point P of the set the distance $|PF|$ to a fixed point F, the *focus*, is equal to the distance $|Pl|$ to a fixed line l, the *directrix*:

$$\{P : |PF| = |Pl|\}.$$

The focus–directrix property of the parabola and other conic sections is due to Pappus.



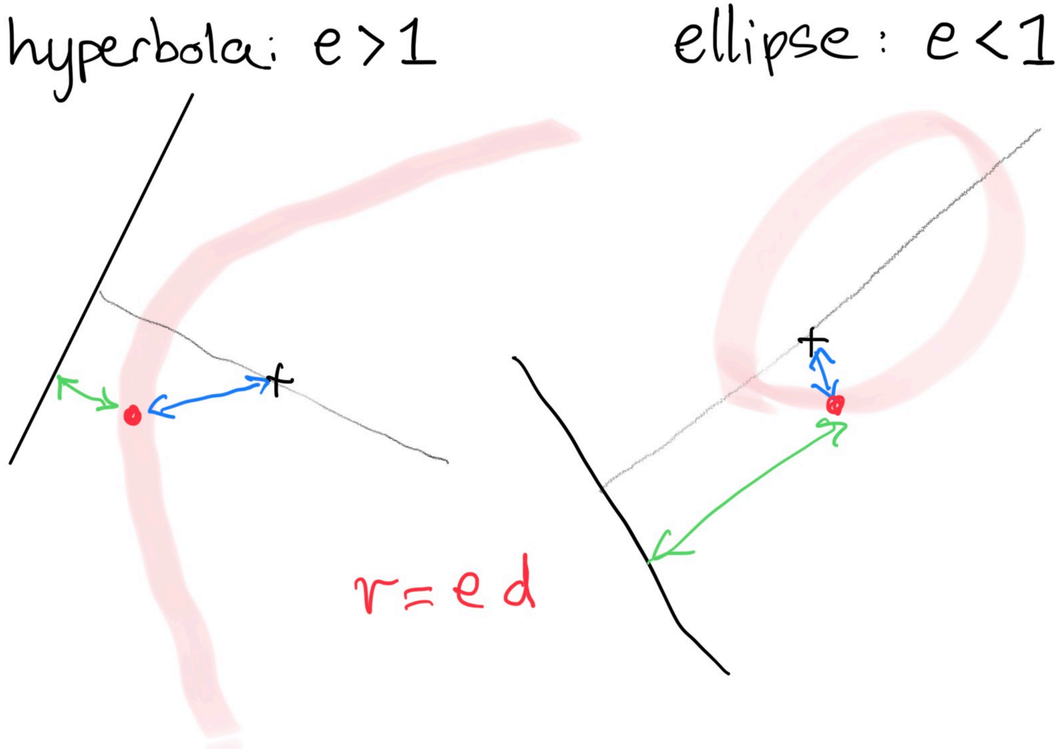
Pappus of Alexandria (/ˈpæpəs/; Greek: Πάππος ὁ Ἀλεξανδρεύς; c. 290 – c. 350 AD) was a Greek mathematician of Late Antiquity known for his *Synagoge* (Συναγωγή) or *Collection* (c. 340),[1] and for Pappus's hexagon theorem in projective geometry. Almost nothing is known about his life except for what can be found in his own writings, many of which are lost. Pappus apparently lived in Alexandria were he worked as a mathematics teacher to higher level students, such one named Hermodorus.[2]


PAPPI
ALEXANDRINI
MATHEMATICAE
COLLECTIONES
A FEDERICO
COMMANDINO
VRBINATE
IN LATINVM CONVERSÆ,
ET COMMENTARIIS
ILLVSTRATAE.

PISAVRI,
Apud Hieronymum Concordiam,
M. D. LXXXVIII.
Superiorum Concessu.

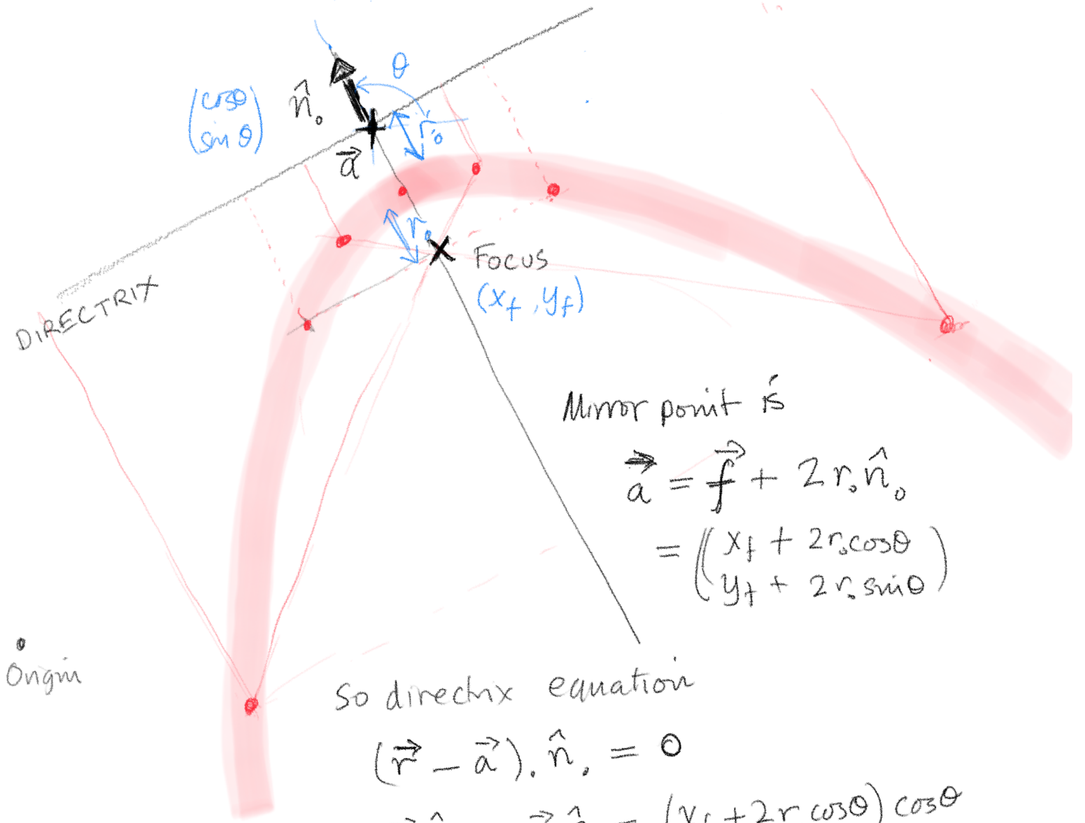
Title page of Pappus's *Mathematicae Collectiones*, translated into Latin by Federico Commandino (1588).

This can be generalized to other forms of conics



Use $\Sigma(r_k - e\, d_k)^2$ as our distance metric

Solve for x, y, r, θ, e



If this is so clever, why did nobody else do it?

Yeah, they did …

… but not until 2018 …

… and only for parabolas

Pattern Recognition 84 (2018) 301–316

A fast robust geometric fitting method for parabolic curves

Ezequiel López-Rubio[a], Karl Thurnhofer-Hemsi[a,*], Elidia Beatriz Blázquez-Parra[b], Óscar David de Cózar-Macías[b], M. Carmen Ladrón-de-Guevara-Muñoz[b]

[a] Department of Computer Languages and Computer Science, University of Málaga, Bulevar Louis Pasteur, 35, Málaga 29071, Spain
[b] Department of Graphical Engineering, Design and Projects, University of Málaga, C/ Doctor Ortiz Ramos, Málaga 29017, Spain

Therefore we can rewrite (7) as:

$$E_d(x,y) = E_f(x,y)$$

$$E_d(x,y) = \sqrt{\frac{(ax+by+c)^2}{a^2+b^2}}$$

$$E_f(x,y) = \sqrt{(x-u)^2+(y-v)^2}$$

Now we may consider the minimization of the following cost function:

$$\mathcal{E}(\mathbf{p}) = \frac{1}{N}\sum_{i\in R_d}\left(E_f(x_i,y_i) - E_d(x_i,y_i)\right) +$$

$$\frac{1}{N}\sum_{i\in R_f}\left(E_d(x_i,y_i) - E_f(x_i,y_i)\right) - \lambda E_d(u,v) \qquad (12)$$

Very similar to my idea: evidence that I am on the right track

Implementation: my `confit` python package

https://github.com/div-B-equals-0/confit/

Uses `lmfit` library under the hood:

- Unified interface to multiple minimization and fitting algorithms
- Easy to set bounds/constraints on parameters or freeze them
- Determination of confidence levels via MCMC

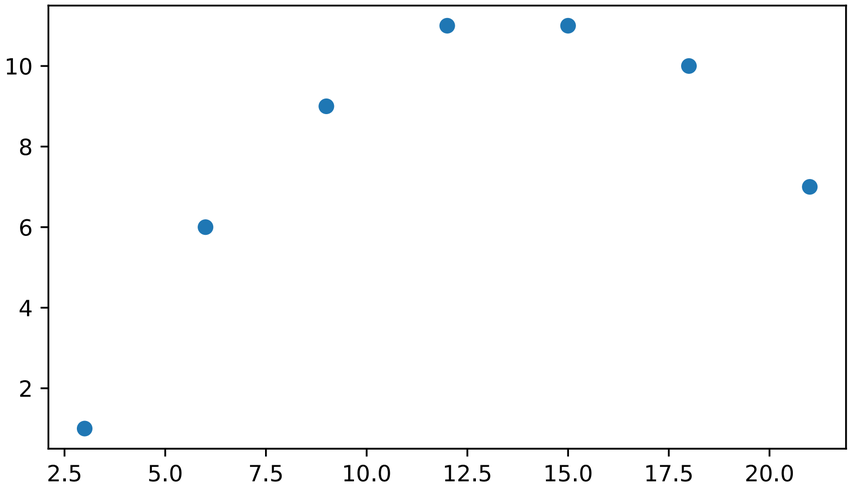
```
    # Set limits on parameters
    params["r0"].set(min=0.0)
    params["theta0"].set(min=0.0, max=360.0)
    params["eccentricity"].set(min=0.0)
    if only_parabola:
        params["eccentricity"].set(vary=False)
```

```
def fit_conic_to_xy(
    xdata,
    ydata,
    eps_data=None,
    only_parabola=True,
    restrict_xy=False,
    restrict_theta=False,
):
    """Fit a conic section curve to discrete (x, y) data points."""
```

```
    # Create Minimizer object
    minner = lmfit.Minimizer(
        residual, params, fcn_args=(xdata, ydata), fcn_kws={"eps": eps_data}
    )
    # do the fit
    result = minner.minimize(method="leastsq")
    return result
```

Objective function

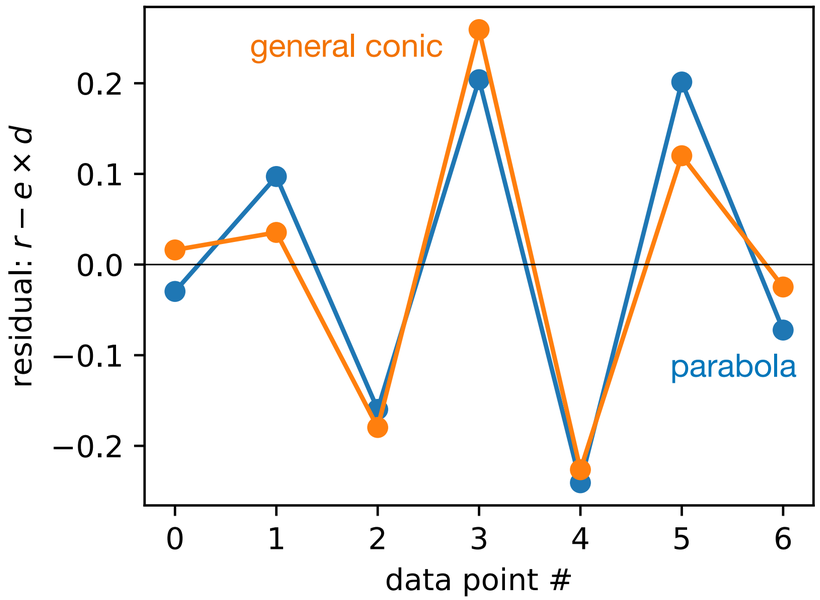
Examples

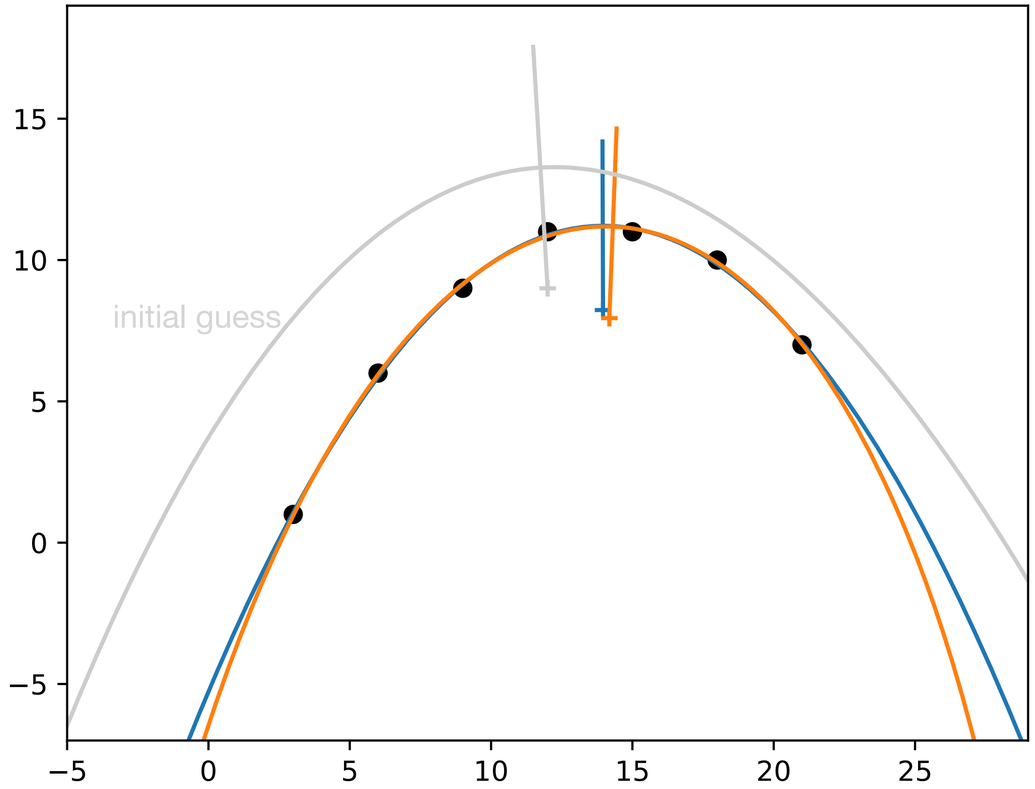


Test data

Symmetric arrangement of 7 points, which I then stretch and distort to make it more interesting. Using fewer than 7 points is not recommended, although it is possible to ge spectacularly small residuals that way!

```
xpts, ypts = np.array([1, 2, 3, 4, 5, 6, 7]), np.array([0, 4, 6, 7, 6, 4, 0])
ypts += xpts
xpts *= 3
```





The fit seems good, but what about uncertainties in the parameters?

Parameters

name	value	standard error	relative error	initial value	min	max	vary
x0	13.9604323	0.31535709	(2.26%)	12.0	-inf	inf	True
y0	8.23028621	0.21719266	(2.64%)	9.0	-inf	inf	True
r0	2.98270890	0.15103907	(5.06%)	4.279303547652316	0.00000000	inf	True
theta0	90.1860042	2.64713202	(2.94%)	93.41881035763075	0.00000000	360.000000	True
eccentricity	1.00000000	0.00000000	(0.00%)	1.0	0.00000000	inf	False

Correlations (unreported values are < 0.100)

Parameter1	Parameter 2	Correlation
y0	r0	-0.9428
x0	theta0	-0.9271
r0	theta0	+0.5817
y0	theta0	-0.5134
x0	r0	-0.4388
x0	y0	+0.3861

parabola

θ = 90 ± 3

name	value	standard error	relative error	initial value	min	max	vary
x0	14.1836372	0.62232238	(4.39%)	12.0	-inf	inf	True
y0	7.94433898	0.70227002	(8.84%)	9.0	-inf	inf	True
r0	3.24215163	0.61612568	(19.00%)	4.279303547652316	0.00000000	inf	True
theta0	87.8031870	6.56584850	(7.48%)	93.41881035763075	0.00000000	360.000000	True
eccentricity	0.93363817	0.14810053	(15.86%)	1.0	0.00000000	inf	True

Correlations (unreported values are < 0.100)

Parameter1	Parameter 2	Correlation
y0	r0	-0.9908
x0	theta0	-0.9701
r0	eccentricity	-0.9506
y0	eccentricity	+0.9279
theta0	eccentricity	+0.8303
x0	eccentricity	-0.7571
r0	theta0	-0.6780
y0	theta0	+0.6587
x0	r0	+0.6146
x0	y0	-0.5994

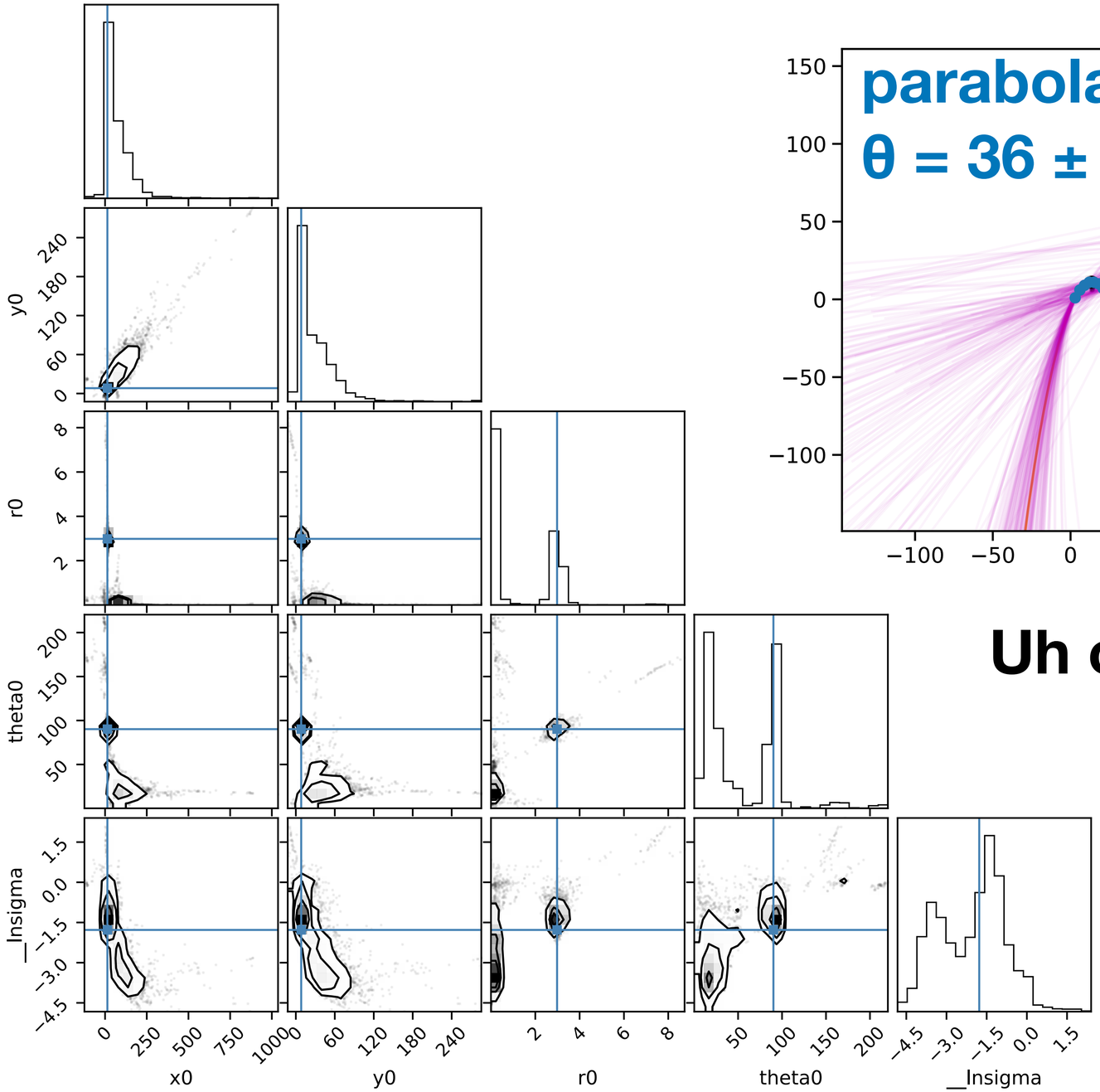
general conic

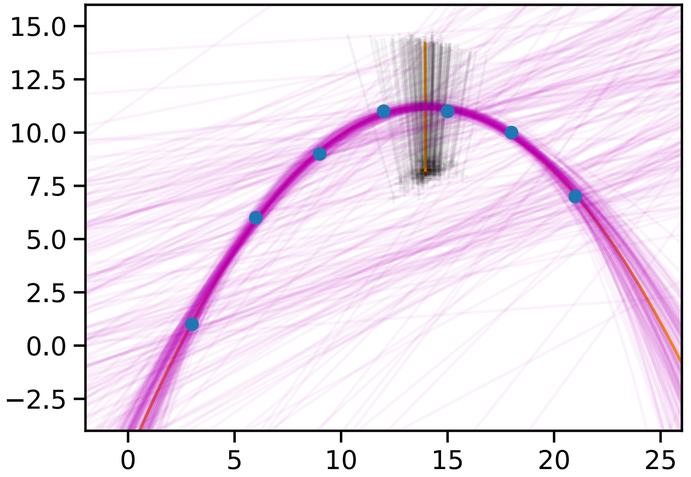
θ = 88 ± 7

Orientations seem consistent

Uncertainties in parameters come from covariance matrix.

This is often good enough, but we should use MCMC to get more robust confidence intervals



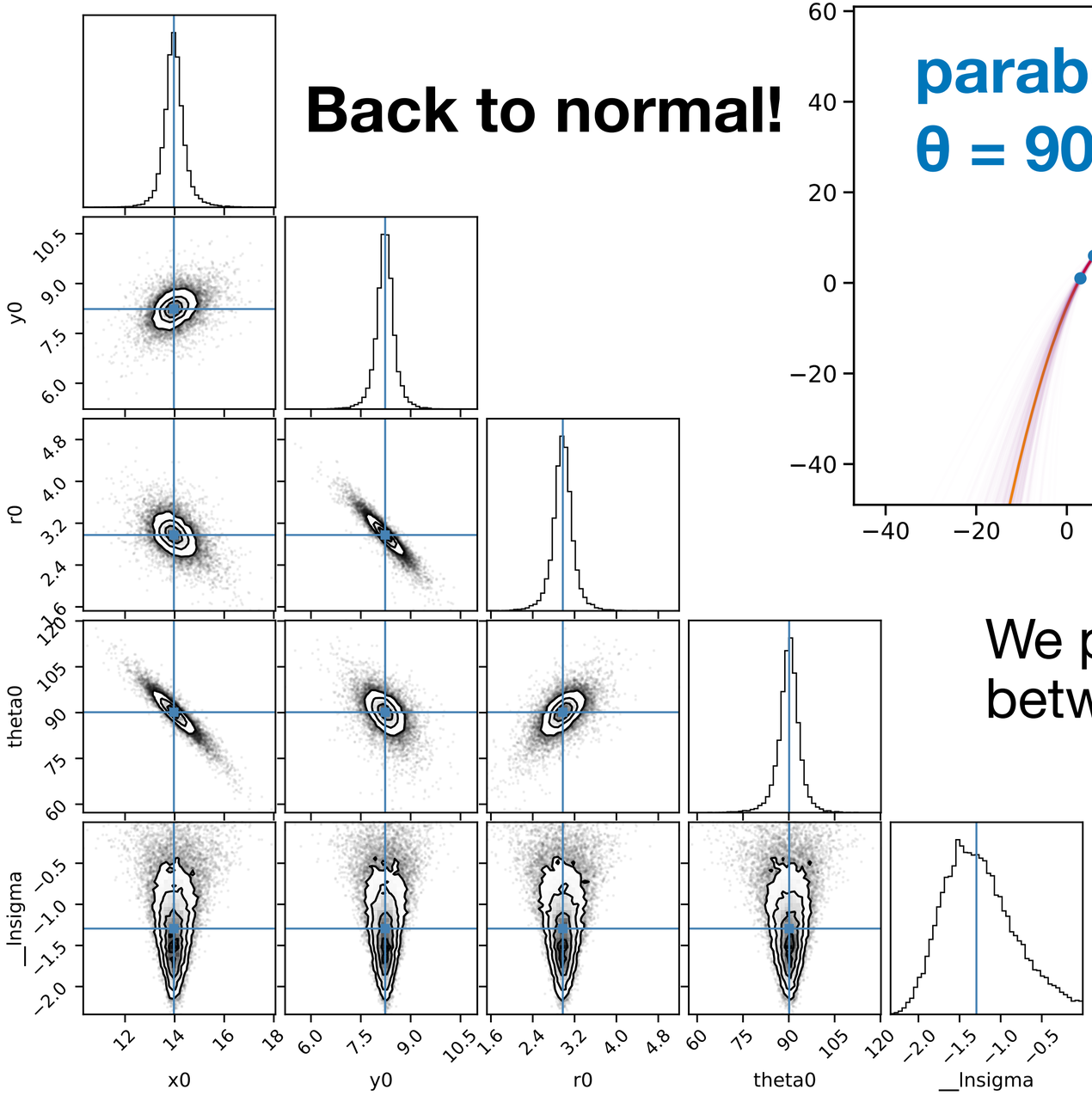


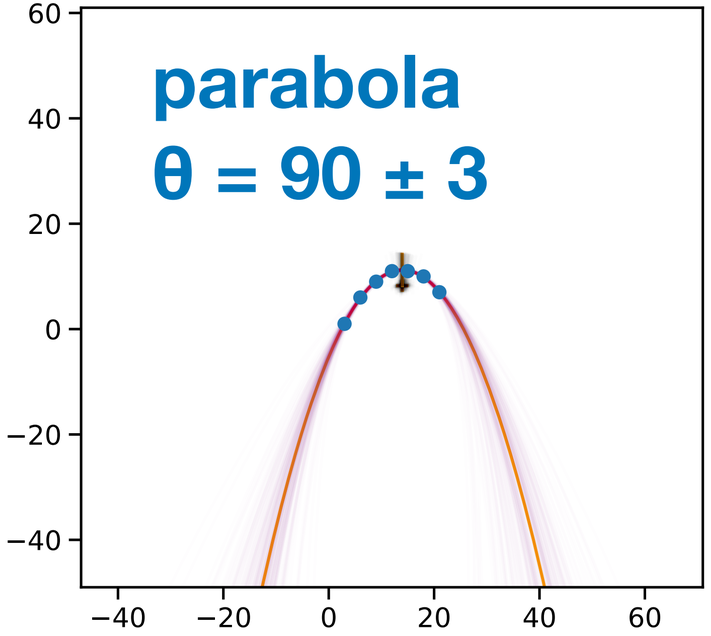
Uh oh … what have we found?

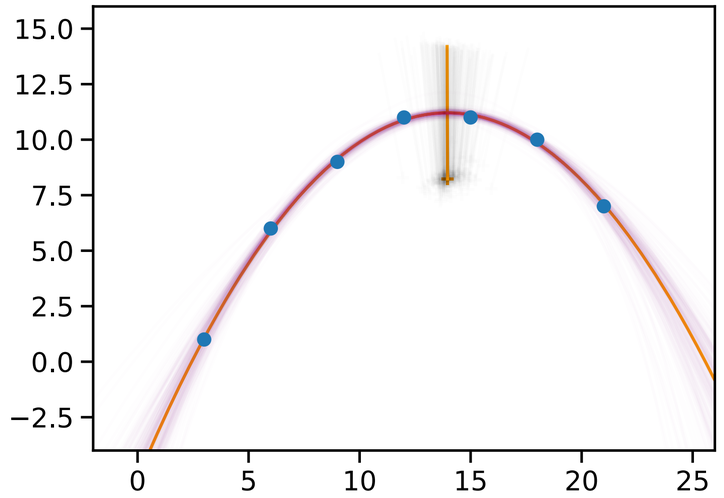
MCMC has found an additional set of parameters that give “good fits”

This clearly makes no sense, but how can we fix it?

Back to normal!







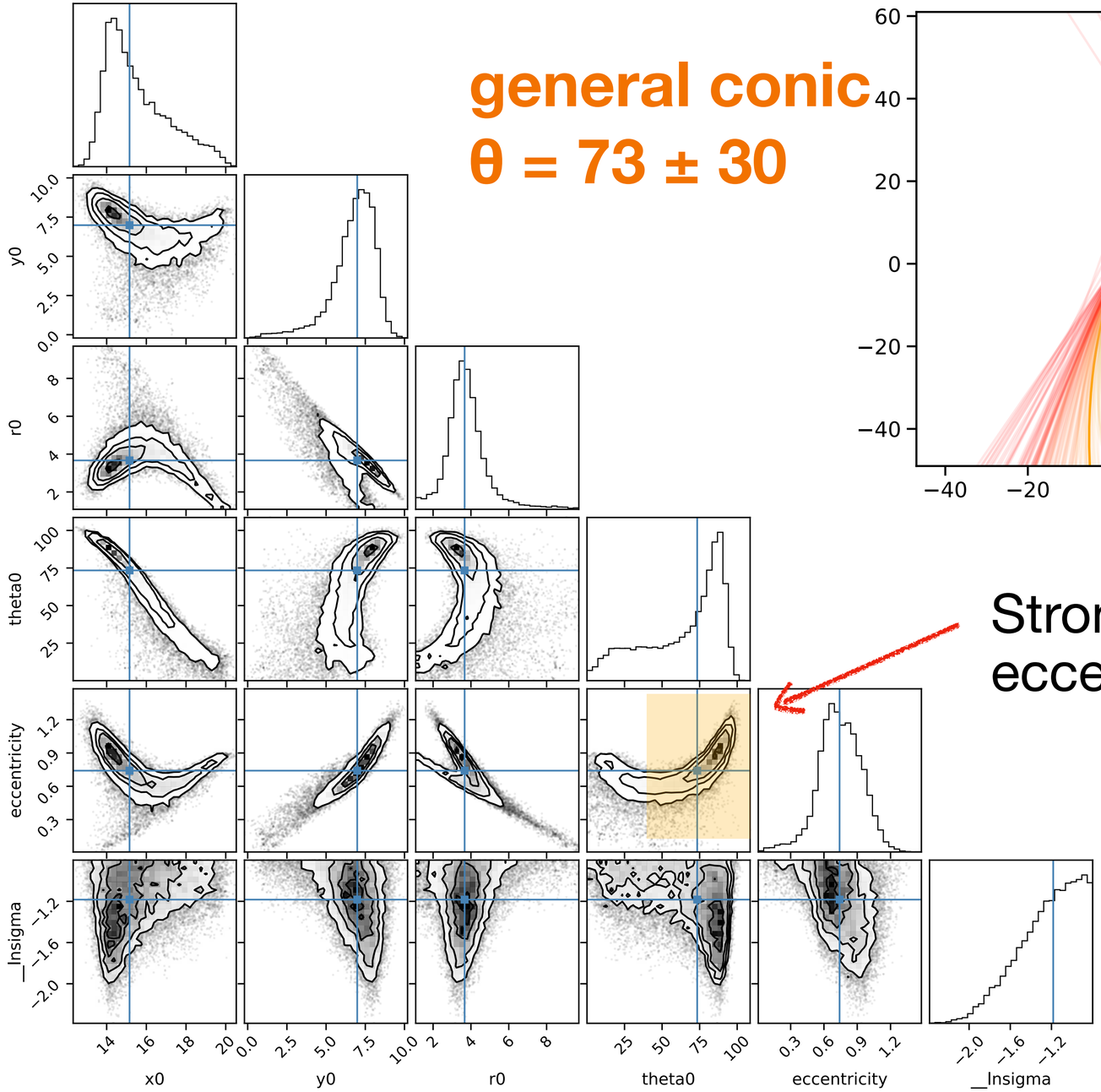
We put limits on the size parameter:
between 0.5 and 2 times the initial guess

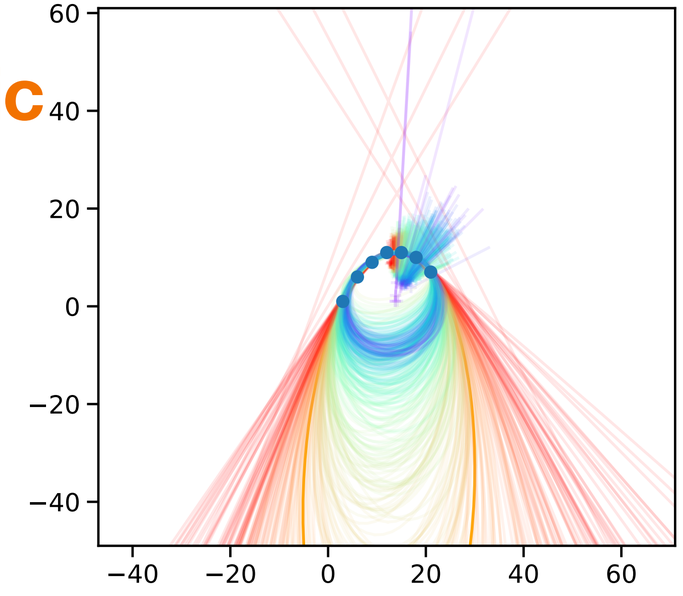
```
new_params = result_p.params.copy()
rscale = new_params["r0"].value
new_params["r0"].set(min=rscale / 2, max=rscale * 2)
```

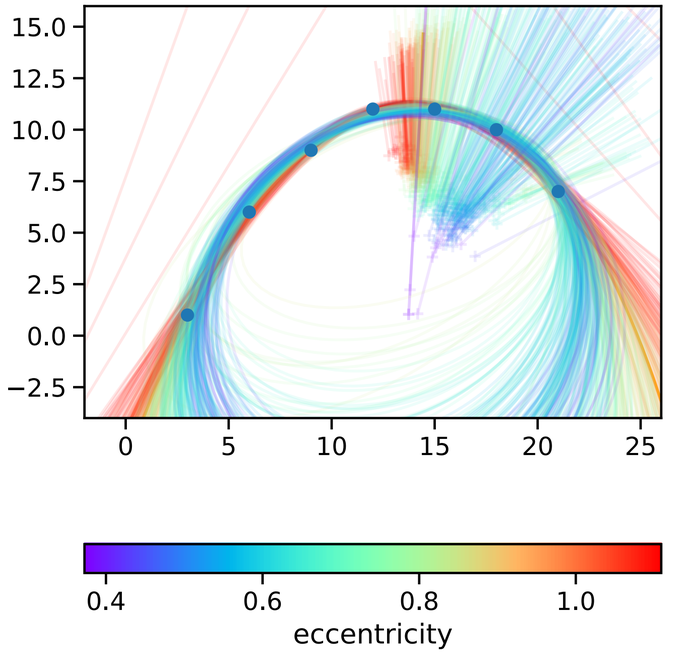
Now we have nice elliptical shapes
for the confidence regions

general conic

θ = 73 ± 30







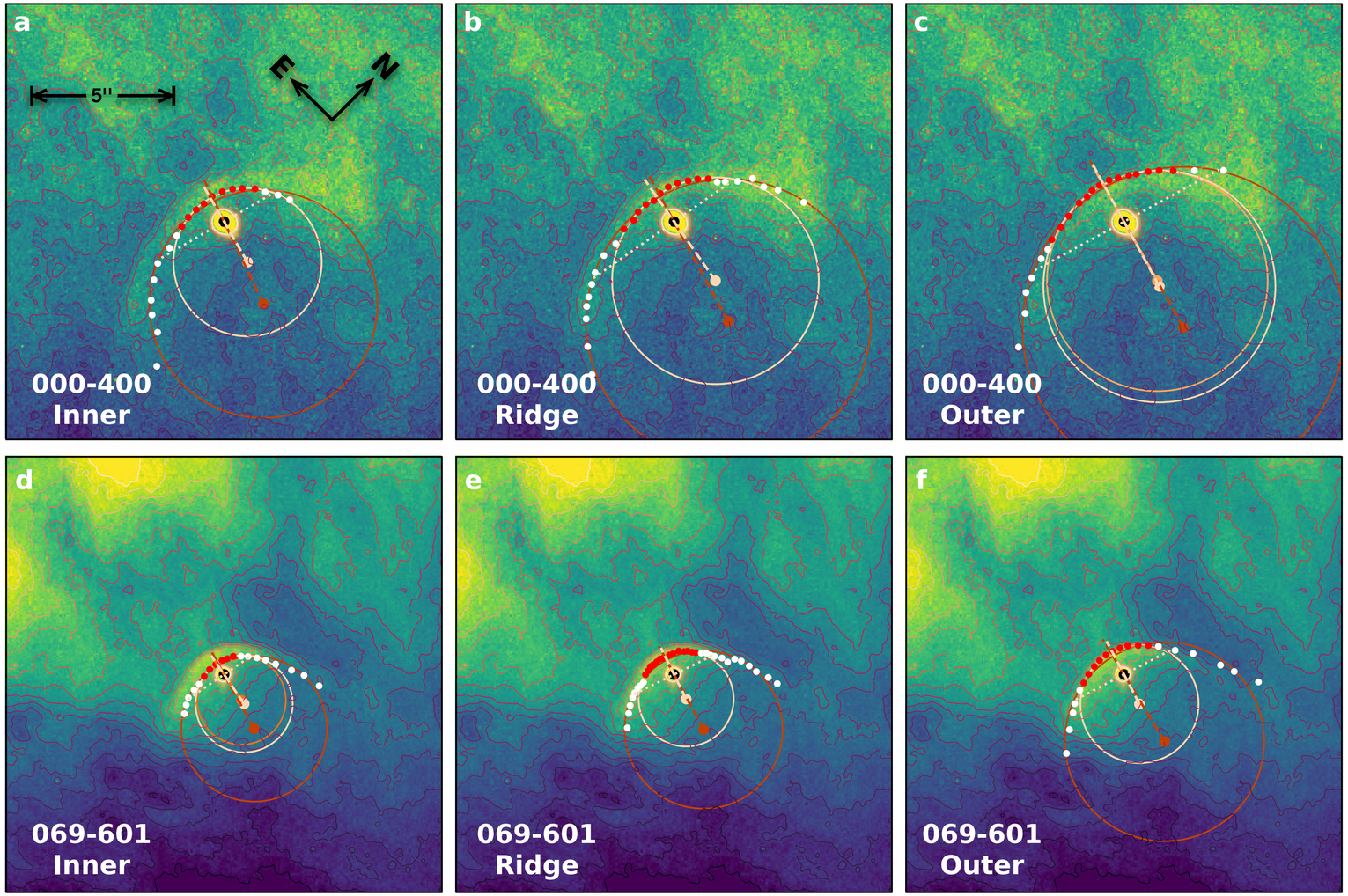
Strong correlation between eccentricity and angle

With freely varying eccentricity we still have issues

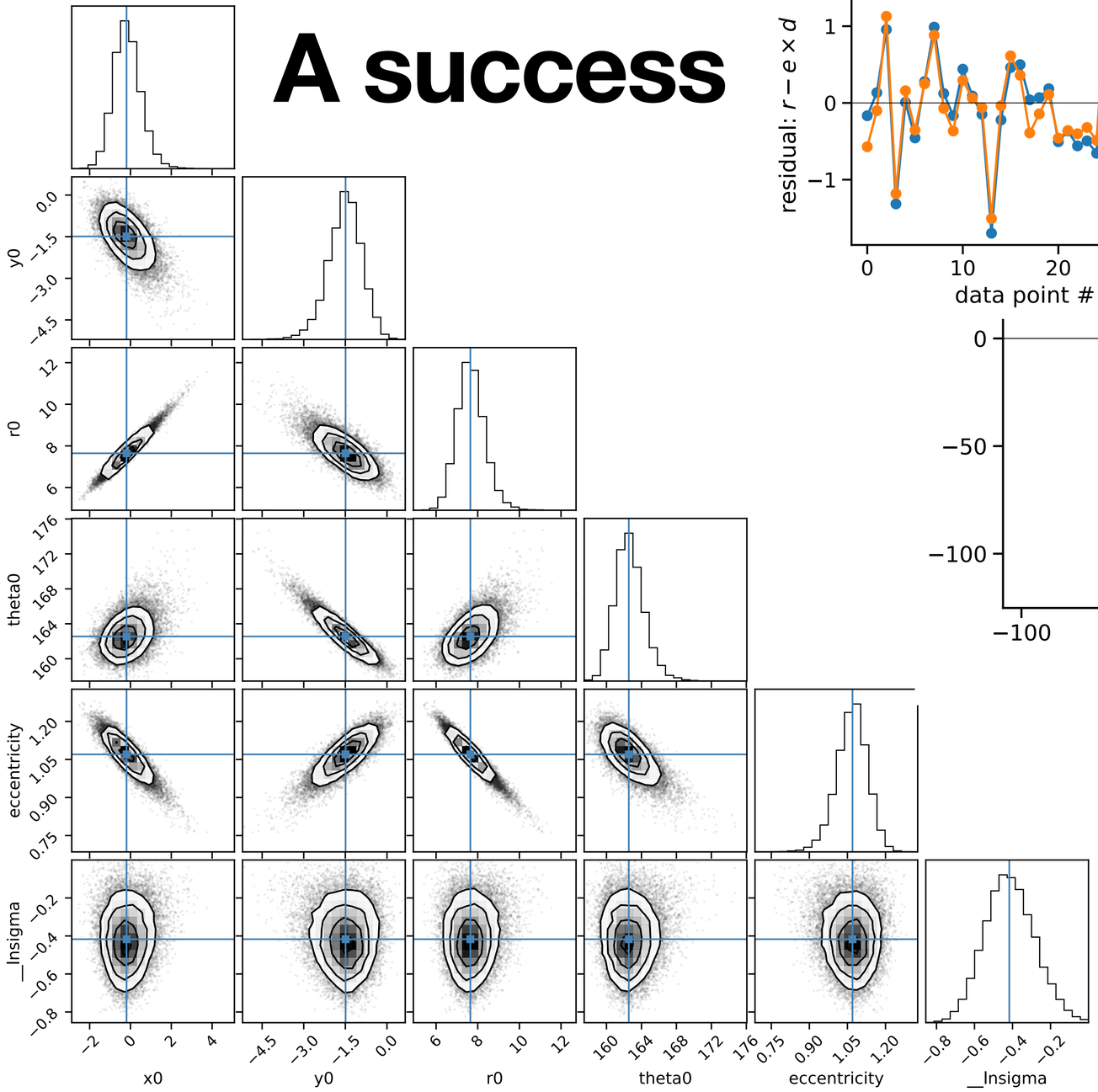
But this is mostly due too the small number of data points

Test on real data

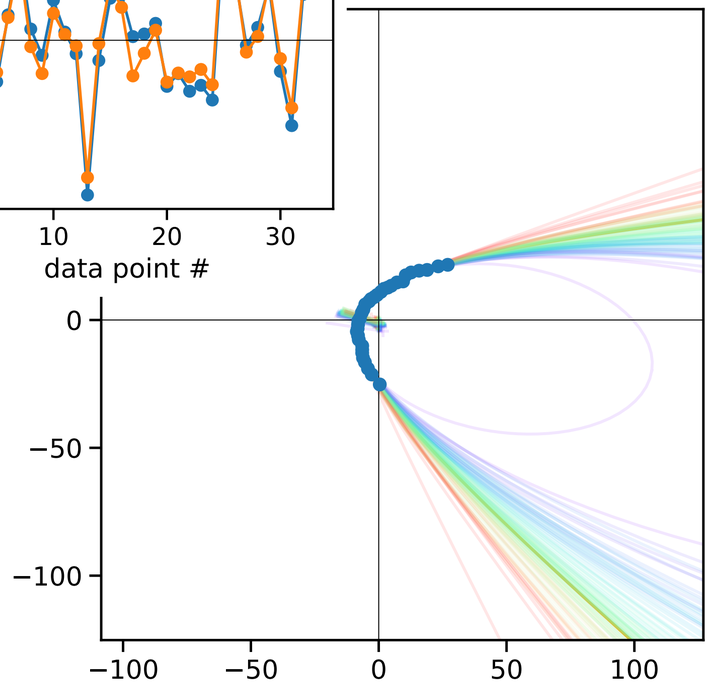
Tarango-Yong & Henney 2018MNRAS.477.2431T

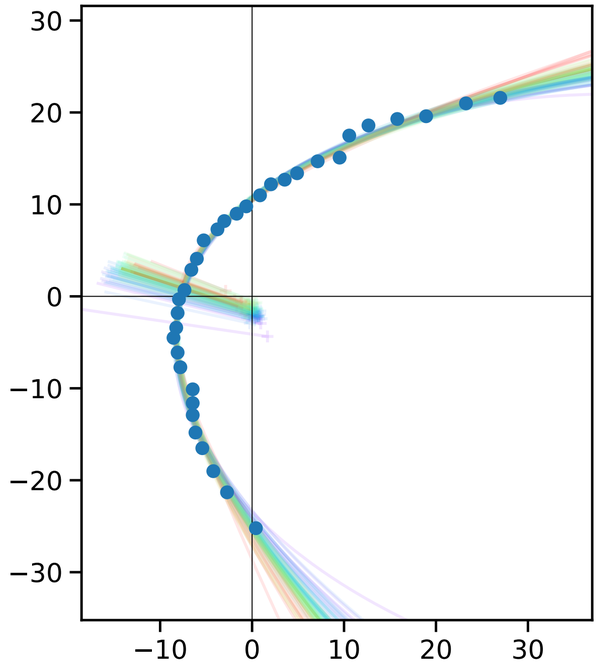


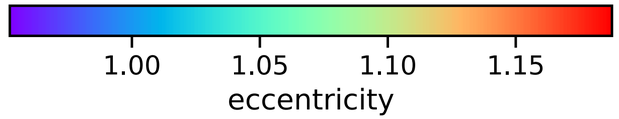
A success

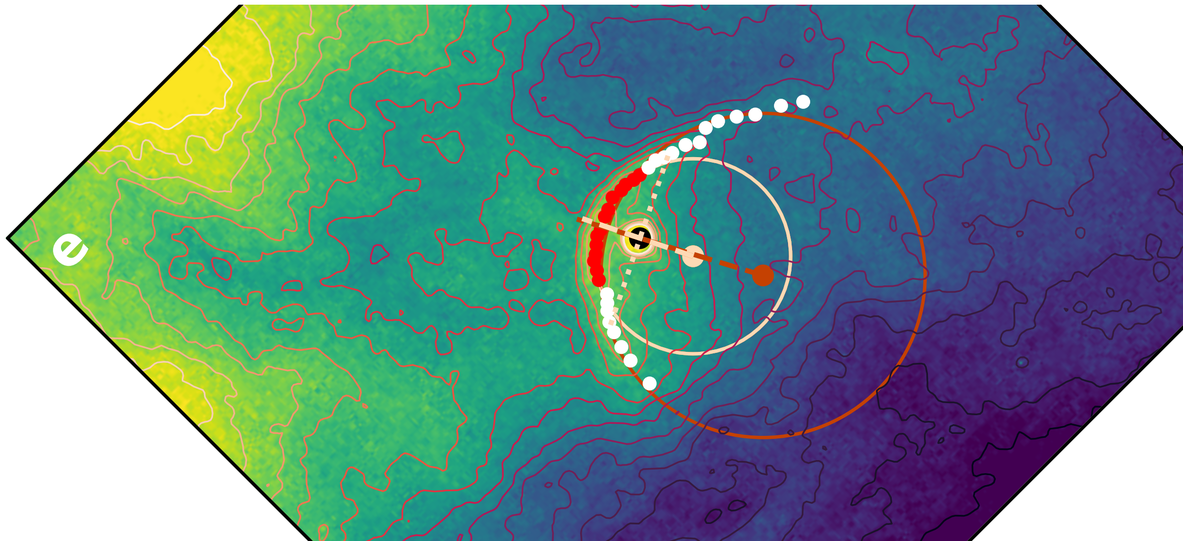

x0
y0
r0
theta0
eccentricity
__lnsigma

residual: $r - e \times d$
data point #

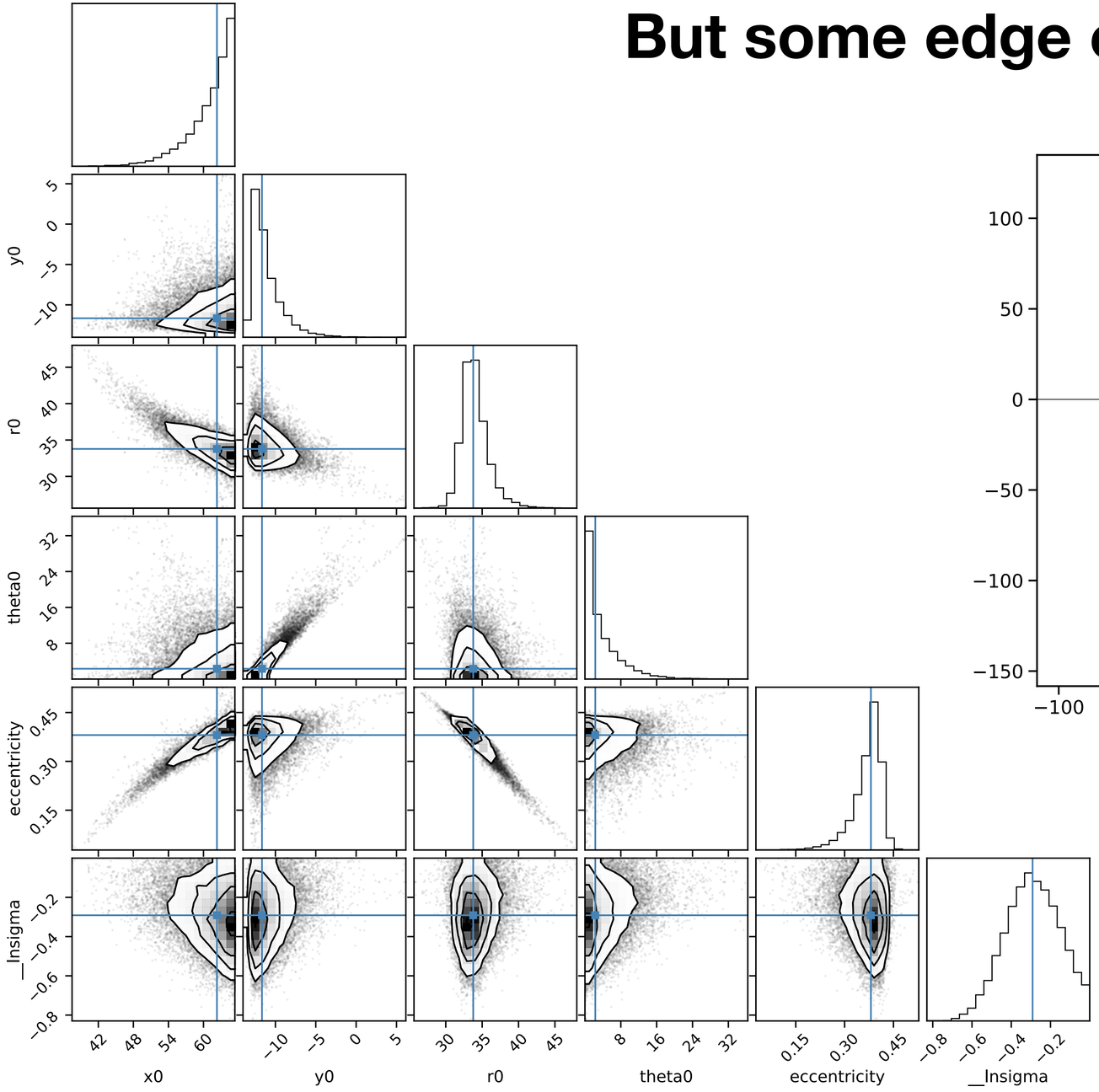


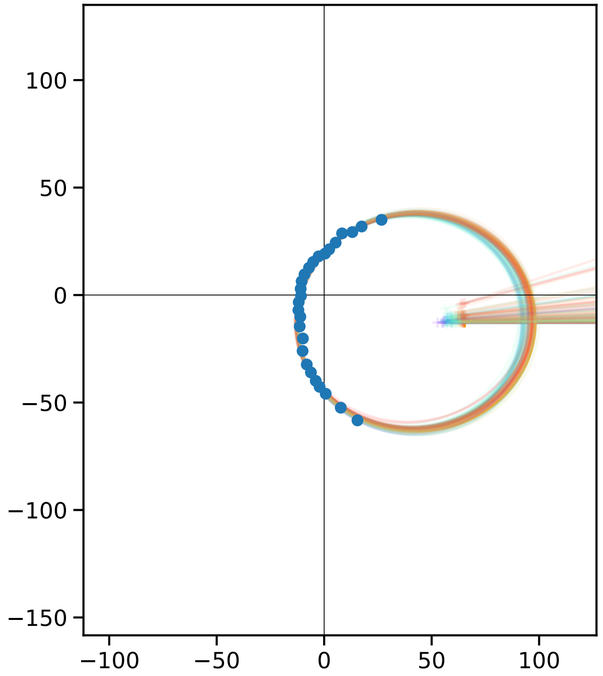


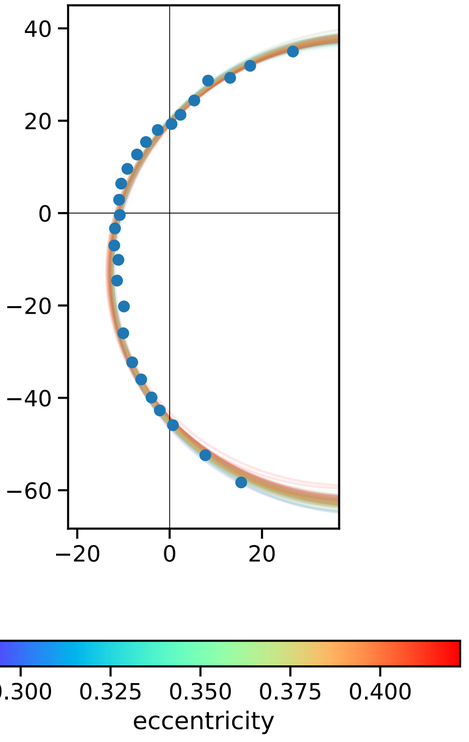

eccentricity


e

But some edge cases still need to be finessed







For low-eccentricity ellipses, the algorithm tends to want to fit the wrong side of the ellipse

Conclusions

- **Fitting conic sections can be useful, easy, and fun**
- **My python package *confit* is available for doing this**
- **Includes sample notebooks for all the fits included in this talk**
- **Thank you!**

https://github.com/div-B-equals-0/confit