An FSQ Analysis: How Error in Quad Center Position Affects Distance from Surface

Amanda Aho
May 19, 2016
University of Minnesota
School of Physics and Astronomy

Abstract

Gravitational lens modelling has been a popular way of probing into the mass distribution of distant galaxies. Modelling a mass distribution can become a difficult task, and there have been some recent methods to give us an understanding of lenses without having to do develop models. One such method may be a quick test of the distance of a quadruply lensed system from the Fundamental Surface of quads, as introduced in Woldesenbet & Williams 2012. The Fundamental Surface (FSQ) is a plane in the 3-dimensional space created by the relative angles between images. The surface is defined by a fit equation while images are determined by their arrival times. Quads with distances from the FSQ of less than 2° is believed to mean the system may be simply modelled by a twofold symmetric lens. Here we discuss how error in the location of the central lensing galaxy of a sample of 35 quads can affect this distance and its possible consequences. We show that a narrower distribution of possible lens positions is favorable for what has been found previously, while a wider distribution causes drastic changes in this value for each quad.

Introduction

Lens modelling has been a wide topic of discussion in the scientific community for the past several decades. Traditionally, looking for ways to model the mass distribution of a lens has been the key way to tell us about gravitational lenses. This can be done through careful analysis of the arrival times of images, their positions, fluxes, magnification, etc. In Woldesenbet & Williams 2012 (hereafter referred to as W&W 2012) they found that useful information about the lens can be found without first requiring a mass model. They stated that for a quadruply lensed system it is possible to tell whether or not a system is twofold symmetric by how far it is from a 2D surface called the Fundamental Surface of quads (FSQ). In their paper they investigated both a set of synthetic lenses and a sample of 40 observed quads. For information on these quads, refer to Table 2 of W&W 2012. They made the observation that if a quad does not fit within 2° of the FSQ then it cannot be modelled by a twofold symmetric lens[1]. In the case of the observed quads they found the quads sat in a ‘cloud’ formation around the FSQ with varying separation.

[1] Note that this 2° is the distance from the FSQ, not to be confused with the relative angle(s).

This paper is an extension of their work and is a study of the sample of 40 quads. Here we investigate how error in the location of the lensing galaxy can affect distance of a quad from the FSQ. A possible error in this position can tell us more about if a quad can be modelled by a simple symmetric lens.

Method

Much of this work consisted of creating a pipeline through which to test any number of new lens positions. The data was drawn from 40 known galaxy-lens quads used in W&W 2012. References for each quad are given in that paper and the positional information for each can be found either through their respective reference or on the CASTLeS website.

The first step of this process was to take the original quad data in (x, y) coordinates and shift each so that their new centers (lensing galaxy positions) were at (0, 0) which would make later adjustments easier. This was done using the simple equation $x_{new} = x - x_{center}$ and similarly for the y coordinate. A simple check of this method was done by plotting both the original and new coordinates and making sure that they matched[2]. These sets of shifted coordinates were the "original" data fed through the pipeline, as it did not require an extra step in the functions of the pipeline to account for different centers.

Then the pipeline was created using the Python programming language through the Spyder software. This software was chosen as it made it easier to access the NumPy computing package. The first job of the pipeline was to be able to calculate the relative angles θ_{12}, θ_{23}, and θ_{34}. Image positions were first converted into their respective polar coordinates. The radial position was not needed for this analysis and so was not calculated; only the θ positions were kept, which were found by

$$\theta = \tan^{-1}\frac{y}{x}$$

Positions in negative x coordinate were respected and adjusted as needed within the pipeline. Once θ position for each image in a quad was known, relative angles were calculated using

$$\theta_{ij} = \left|\theta_i - \theta_j\right|$$

If a relative angle was found to be more than 180° then its conjugate was calculated and kept. This method was chosen because it makes little sense to use a relative angle larger than 180°. Additionally, most of the relative angles given in W&W 2012 were less than 180°.

Once relative angles were found it was time to calculate the distance of each quad from the FSQ. This was done using Equation 18 from W&W 2012:

$$\begin{aligned}\theta_{23,fit} = {} & -5.792 + 1.783\theta_{12} + 0.1648\theta_{12}^2 - 0.04591\theta_{12}^3 - 0.0001486\theta_{12}^4 + 1.784\theta_{34} \\ & - 0.7275\theta_{34}\theta_{12} + 0.0549\theta_{34}\theta_{12}^2 + 0.01487\theta_{34}\theta_{12}^3 + 0.1643\theta_{34}^2 + 0.05493\theta_{34}^2\theta_{12} \\ & - 0.03429\theta_{34}^2\theta_{12}^2 - 0.04579\theta_{34}^3 + 0.01487\theta_{34}^3\theta_{12} - 0.0001593\theta_{34}^4\end{aligned}$$

[2] "Matched" simply meaning distances from center were conserved.

where all coefficients, θ_{12}, $\theta_{23,fit}$, and θ_{34} are in radians. Finally, distance from the FSQ was found by subtracting this value from the one previously measured as θ_{23} (i.e., $\Delta\theta_{23} = \theta_{23,measured} - \theta_{23,fit}$). Then $\Delta\theta_{23}$ was plotted against θ_{23} for analysis.

As a cross-check, the pipeline was used to reproduce Figure 9 of W&W 2012. It was found that five of the points were giving incorrect values for $\Delta\theta_{23}$. After close examination it was found that this was due to the 180° cutoff. In W&W 2012 quads 17, 20, 25, 30, and 40 had values for either θ_{12} or θ_{34} higher than 180°, as can be seen in their Table 2. This pipeline only used relative angles less than 180° so this resulted in different values in the calculation of $\theta_{23,fit}$ and thus, in $\Delta\theta_{23}$. A better method to account for this could not be found at the time so the final results exclude these five quads for simplicity. A reconstruction of W&W 2012 Figure 9 is shown in Figure 1 of this paper.

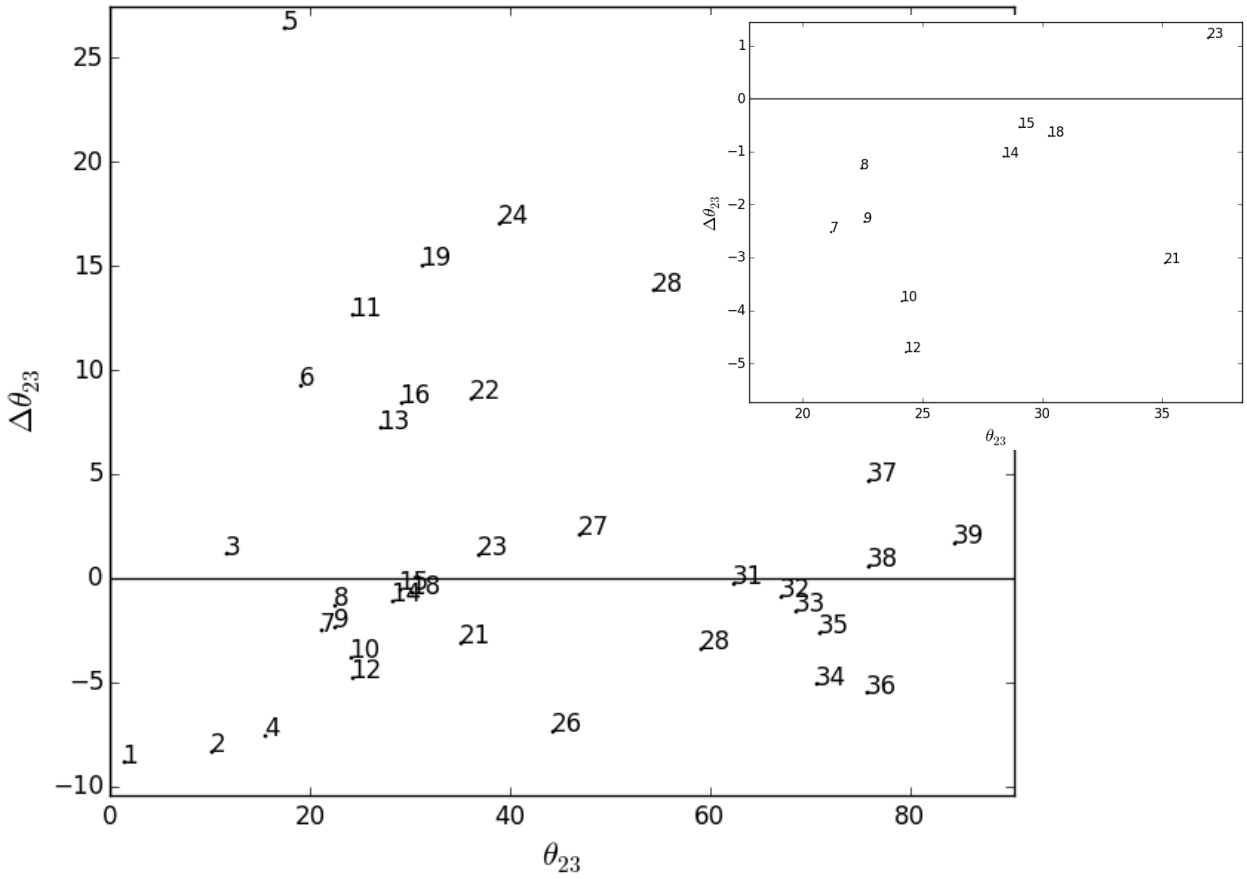


Figure 1: A reconstruction of Figure 9 from W&W 2012. Here quad numbers 17, 20, 25, 30, and 40 are excluded, as the 180° "cutoff" created inconsistencies with previous work. Angles are in degrees for consistency with W&W 2012.

Once the calculation for distance from FSQ was tested, the next piece was to include a random generation of quad centers. This was easily done using NumPy's pseudo-random number generator. The function generated 500 sets of new centers (physically the lensing gakaxy) on the x-y plane with a Gaussian distribution. A Gaussian was chosen as it is more likely that the correct center is near where it is believed, with a lower probability that the center would be

farther away. The width of the distribution did not exceed 1 arcsecond, as it is unlikely the error in the lens position would be so large. Different widths of the distribution were chosen and will be analyzed in the upcoming section.

Once the new lens positions were generated, the last missing piece was to apply the new lens to each quad. This was done similarly to the original shift to (0, 0): for each new lens the new (x, y) coordinate of each image was re-calculated using $x_{new} = x - x_{center}$, and same for the y coordinate.

To summarize, the steps of the pipeline as they appeared in it are as follows:

1. The program "read in" the original data with center positions already shifted to (0,0)
2. 500 new lenses (centers) were generated in (x, y) coordinate following a Gaussian distribution
3. Each center was applied to each quad: images were shifted again to re-center onto (0,0)
4. Relative angles were calculated ("measured")
5. Distance from FSQ was calculated
6. Quads were plotted on the θ_{23}-$\Delta\theta_{23}$ plane

A copy of this code can be found in the Appendix at the end of this paper.

Results

The most telling contributor to the results of this experiment was the width of the Gaussian distribution. At low distribution widths it is easiest to see how error in position of the lensing galaxy affects distance from FSQ. For example, Figure 2a shows these results for a width of 0.01 arcsec on the lens position. Each cluster of points is an individual quad but each point has a different center. As the width of the distribution increases it is harder to tell individual quads apart. Furthermore, they tend to spread out and seem to move upwards as distribution increases. As in Figure 3 for a distribution width of 0.03 arcsec the clusters bulge out and have a wider spread. When we get up to a width of 0.1 arcsec, as in Figure 4, it is nearly impossible to tell individual quads apart and at 0.3 arcsec (Figure 5) it's a lost effort completely.

There is also a somewhat asymptotic trend on the right side of these graphs as distribution increases. This is very prevalent in Figure 5 where the distribution is almost 1/3 of an arcsecond. A trend like this may be attributed to an already "extreme" value for θ_{23} for the right-most quads. It is also due to different changes in the relative angles θ_{12} and θ_{34} as new centers are applied. As error in lens position gets larger, for a new center far away, more images may get to one side of it and relative angles shrink. Then the first term of the equation dominates so values shift upwards. If this is not the case, then the lens position gets close to one image and relative angles may become extreme, causing serious changes in distance form FSQ. Whatever the specific reason for each quad, the wider the distribution is, the more vertical the spread becomes on the left side, while it flattens out horizontally on the right.

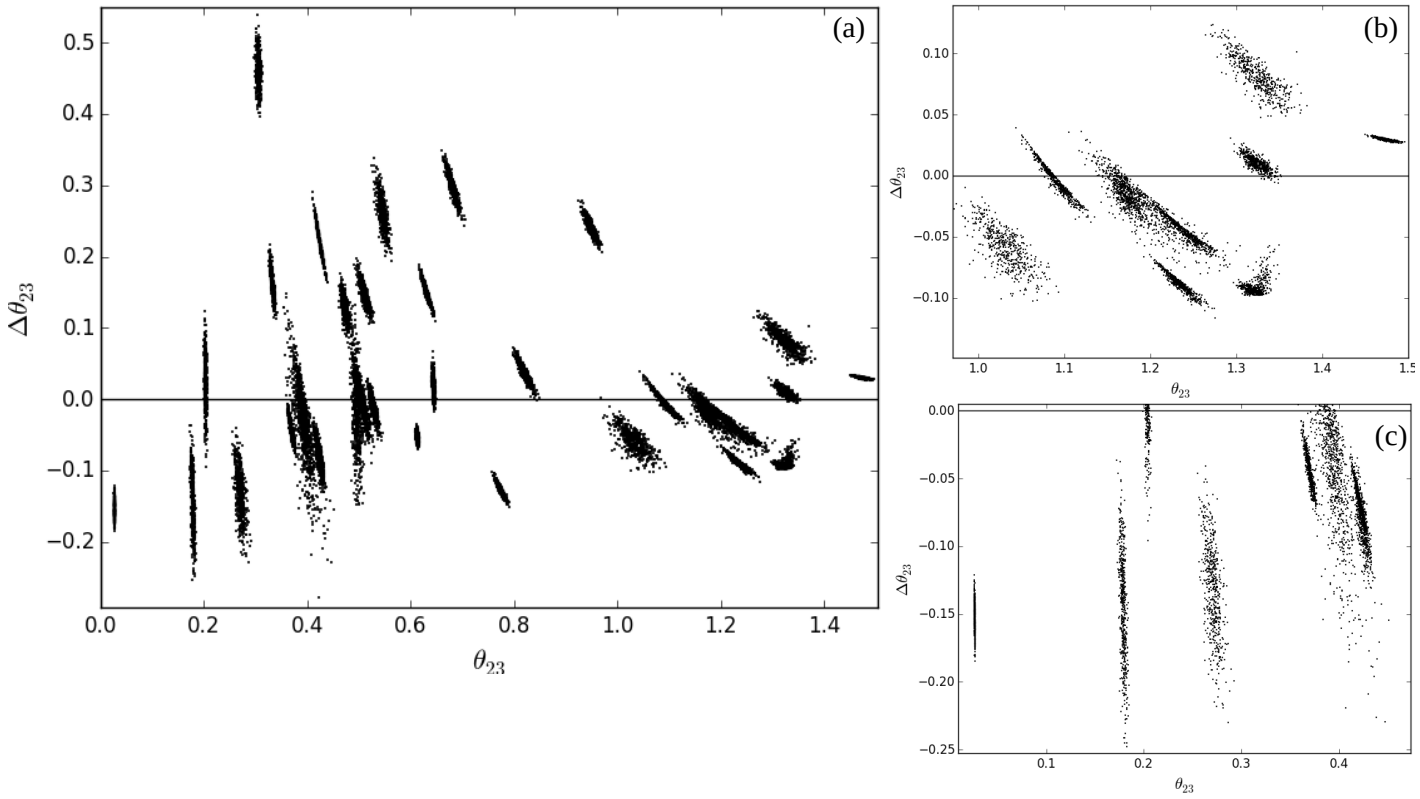
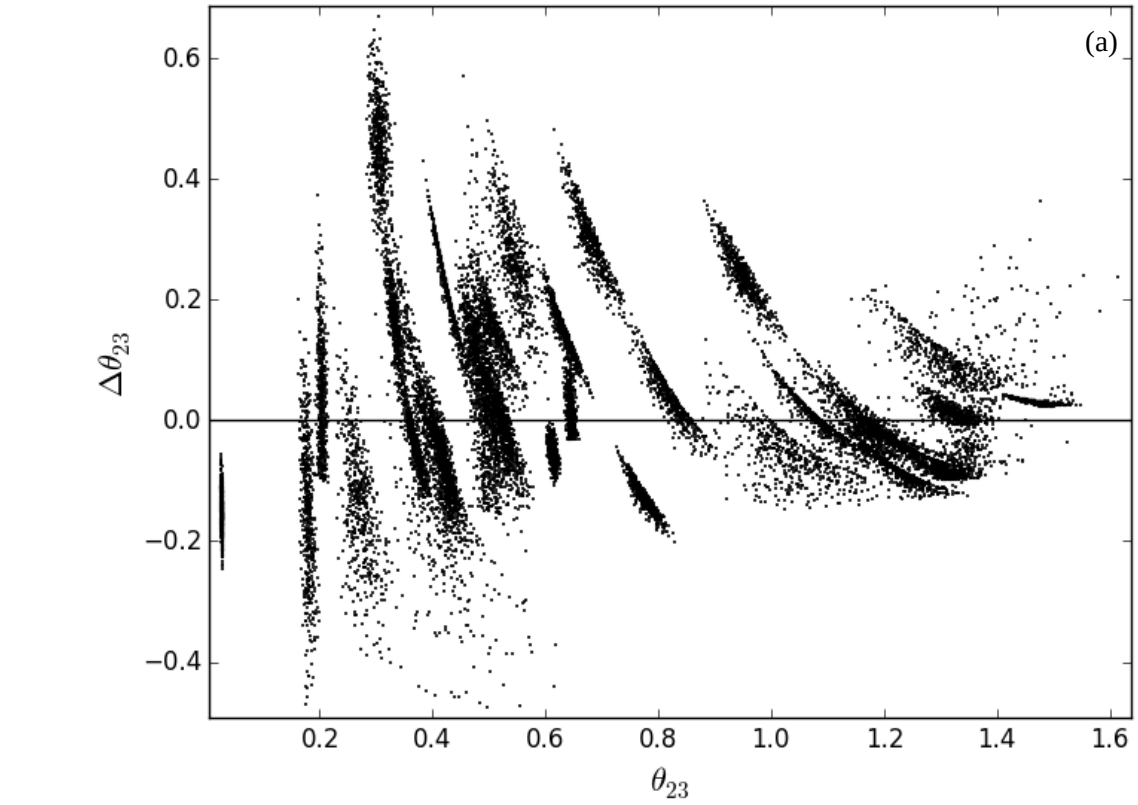


Figure 2: (a) Distribution of FSQ values for each quad with a distribution width in new lens position of 0.01 arcsec. Axes are in radians. Each cloud resembles one quad and each point is a different center. Note how quads are easy to distinguish from one another. **(b)** A zoom in on the "far right" region. Pictured are quads 29 and 31-39. **(c)** A zoom in on region to the left. Pictured are quads 1-4, 7-10, and 12. The last five are originally very on top of each other so it is more difficult to tell which is which here.



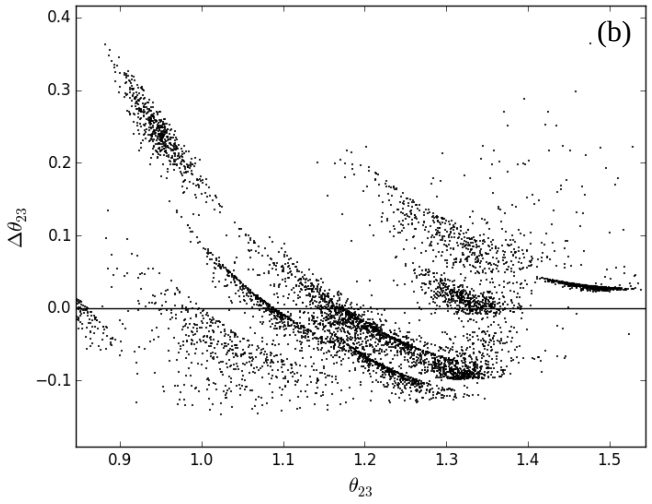


Figure 3: (a) Distribution of distance from FSQ with a Gaussian width of 0.03 arcsec. Note how the points are more widely dispersed and there is slightly less structure to some. **(b)** A zoom in on the far right region. Here it is still possible to distinguish between separate quads but their points are much more scattered.

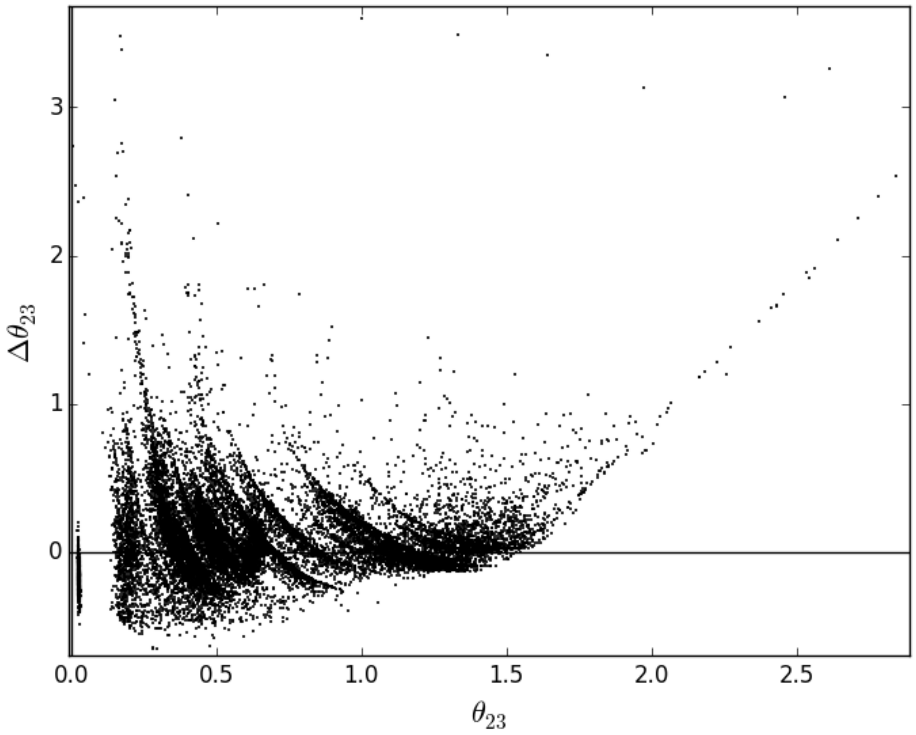


Figure 4: Distribution of distances from FSQ using a distribution width of 0.1 arcsec. Here it is nearly impossible to tell different quads from each other and calculated values have started to creep upwards.

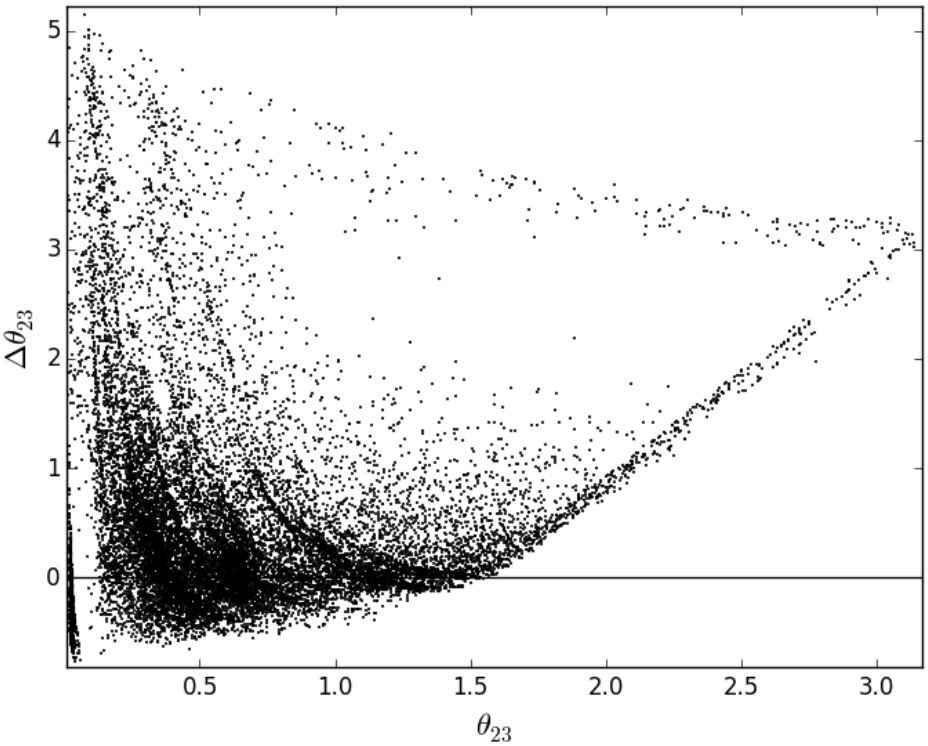


Figure 5: Distribution of distances from FSQ using a distribution width of 0.3 arcsec. Here it is impossible to tell separate quads apart. The asymptotic nature of the quads is very prevalent on the right side. Many values have crept upwards and are more widespread.

Conclusion

As discussed in W&W 2012, the distance of a quad from the Fundamental Surface can tell us some about whether or not that quad may be modelled using a twofold symmetric lens. This paper used a pipeline for 35 observed quads to investigate how error in the position of the lensing galaxy of each quad affected its distance from the FSQ. It was found that a wider distribution of new positions from their original led to different relative angles θ_{12} and θ_{34} and therefore larger and more varied values for $\Delta\theta_{23}$. Lower values for the distribution are favored as it gives similar values as from W&W 2012 which could mean those lenses may be modelled by twofold symmetric lenses. Smaller distributions also favor what is already known about those 35 lenses and may help model some of the systems better.

Some future work may stem from Woldesenbet & Williams 2015, in which the authors discussed Type II lenses and their modelling parameters. Follow-up work would consist of generating a set of synthetic Type II lenses, going through this same pipeline, and comparing the distribution to observations. This could tell us more about how Type II lenses can be modelled, as well as whether or not the observed quads of W&W 2012 can be modelled with Type IIs.

References

Woldesenbet, A.G., Williams, L.L.R. "The Fundamental Surface of Quad Lenses". 2012. MNRAS, 420, 2944.

Woldesenbet, A.G., Williams, L.L.R. "Model-free analysis of quads and substructured galaxies." 2015. MNRAS, 454, 862.

Raybaut, P. The Scientific Python Development Environment (Spyder 2.3.5.2). 2015.

Appendix

Found here is the code used for the pipeline. It does not include commands for writing out data to save, as it was extraneous.

```
import numpy
import math
import matplotlib.pyplot as plt

# function to create new centers/lens positions
def create_centers(n):
    dist = 0.01       # distribution size, 0.01 arcsec
    points = []
    i = 0
    while i < n:
        x = numpy.random.normal(0, dist)
        y = numpy.random.normal(0, dist)
        points += [[x,y]]
```

```
        i += 1
    return points

def shift(quad,x,y):  # function to shift quad for a new center
    normal = []
    i = 0
    while i < len(quad):
        new_x = quad[i] - x
        new_y = quad[i+1] - y
        normal += [new_x,new_y]
        i += 2
    return normal

def calc_angles(coords): # converts set of coordinates to polar and
                         #calculates relative angles
    angles = []
    i = 0
    while i < len(coords):
        x = coords[i]
        y = coords[i+1]
        theta_rad = math.atan(y/x)
        if x < 0:
            theta_deg = math.degrees(theta_rad) + 180
        elif (x > 0) and (y < 0):
            theta_deg = math.degrees(theta_rad) + 360
        else:
            theta_deg = math.degrees(theta_rad)
        angles += [theta_deg]
        i += 2
    dif_12 = abs(angles[0]-angles[1])
    if dif_12 > 180:
        dif_12 = 360 - dif_12
    dif_23 = abs(angles[1]-angles[2])
    if dif_23 > 180:
        dif_23 = 360 - dif_23
    dif_34 = abs(angles[2]-angles[3])
    if dif_34 > 180:
        dif_34 = 360 - dif_34
    angle_list = [math.radians(dif_12), math.radians(dif_23),
              math.radians(dif_34)]
    return angle_list

def calc_23(xf,yf):   # calculates the value of theta_23 using fit eqn
    p00 = -5.792
    p01 = 1.784
    p02 = 0.1643
```

```
    p03 = -0.04579
    p04 = -0.0001593
    p10 = 1.783
    p11 = -0.7275
    p12 = 0.05493
    p13 = 0.01487
    p20 = 0.1648
    p21 = 0.0549
    p22 = -0.03429
    p23 = 0
    p30 = -0.04591
    p31 = 0.01487
    p32 = 0
    p40 = -0.0001486
    p41 = 0
    p50 = 0
    zf = p00 + (p10 * xf) + (p01 * yf) + (p20 * xf**2) + (p11 * xf *
           yf) + (p02 * yf**2) + (p30 * xf**3) + (p21 * xf**2 * yf) +
           (p12 * xf * yf**2) + (p03 * yf**3) + (p40 * xf**4) + (p31 *
           xf**3 * yf) + (p22 * xf**2 * yf**2) + (p13 * xf * yf**3) +
           (p04 * yf**4) + (p50 * xf**5) + (p41 * xf**4 * yf) + (p32 *
           xf**3 * yf**2) + (p23 * xf**2 * yf**3)
    return zf

data = []
ofile = open('quadsdata40_truncated.txt','r')
lines = ofile.readlines()
for line in lines:
    line = line.rstrip('\n')
    llist = line.split('\t')
    data += [llist]
i = 0
while i < len(data):
    j = 0
    while j < len(data[i]):
        data[i][j] = float(data[i][j])
        j += 1
    i += 1
centers = create_centers(500)
difs = []
for quad in data:
    i = 0
    while i < len(centers):
        new = shift(quad, centers[i][0], centers[i][1])
        angles = calc_angles(new)
        difs += [angles]
```

```
        i += 1
theta_measured = []
delta_theta = []
j = 0
while j < len(difs):
    zf = calc_23(difs[j][0], difs[j][2])
    d = difs[j][1] - zf
    delta_theta += [d]
    theta_measured += [difs[j][1]]
    j += 1
plt.scatter(theta_measured, delta_theta, c = 'k', s = 0.3)
plt.axhline(0, color = 'k')
plt.axvline(0, color = 'k')
plt.xlabel('$\\theta_{23}$', fontsize=16)
plt.ylabel('$\Delta \\theta_{23}$', fontsize=16)
plt.show()
```