UNIVERSITY OF CALIFORNIA,
IRVINE
&
SAN DIEGO STATE UNIVERSITY

Using SpArcFiRe to Help Automate GALFIT's Multi-Component Decomposition of Spiral Galaxies

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computational Science

by

Matthew Portman

2024

Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

I would like to acknowledge the support of many, many people who helped me through this arduous process and the several seemingly cataclysmic events that unfolded throughout my time completing this work.

First and foremost, I would like to thank Wayne Hayes and the `SpArcFiRe` group for many many things. Wayne assisted me greatly in figuring out how to handle this monster—his ideas, some good, some bad, formed the basis for several of the techniques we employ. If it weren't for Wayne, I'm sure I'd be off, stuck in `GALFIT` land, crying into my pillow at night wondering why-o-why these blobs of light just aren't turning out right. Beyond his assistance in figuring out the project, he has provided tireless support to me in my times of need. From dealing with the many administrative headaches of the joint program and the CS department's terrible confusion about my status, to making sure I had an appropriate amount of wine (a bottle) to properly label by-eye the (literally) thousands of models that form the foundation of my results, he has always been there when I needed him the most. And, despite Wayne's general insanity, I wouldn't trade him for the world.

As has the group—John English for his many suggestions and useful questions, Sahel Mesforoush for her assistance with `SpArcFiRe` for our rebuttal, Azra Zahin for her assistance in working on the pipeline, Cora Schallock for the best donuts Wayne could ever steal from me, and several others along the way who have chimed in with great ideas.

I would be remiss to acknowledge the support of Chien Peng (and his `GALFIT` website) in assisting us with anything and everything `GALFIT`. His help has been invaluable to this work and of course, none of it would have been possible without him.

Daniel Cheng provided the foundation for this research and helped me to get my feet wet with `GALFIT`.

Aaron Barth provided several insights about observational astronomy and the uses of this research that helped to kickstart the project as well as many pointers along the way.

Adam Miller, Bryan Scott, and my peers in the LSSTC-DA Fellowship Program offered many interesting perspectives to my research. The ideas discussed during the program were invaluable to this project and to much of what I would like for the project to be in the future.

I would also like to thank Fridolin Weber for all the support he gave me in the first years of my PhD. He was an excellent mentor to me and made my transition into this academic phase much, much easier.

I am very thankful to have had Shannon Alfaro as the professor I TA'd for during the majority of my time at UCI. She is an amazing instructor and I learned much about the teaching process, teaching strategies, and the general process of dealing with students. She (and Wayne) repeatedly helped me to navigate the perpetual issues with my TA assignment and I am incredibly grateful for her assistance.

I have also had the opportunity to perform research beyond the scope of this project, with Antonella Palmese at Carnegie-Mellon University doing multi-messenger astronomy through the University Researchers Association, and separately Peter Anninos and Rob Hoffman at Lawrence Livermore National Lab (LLNL) doing neutron star common envelope simulations. These opportunities have greatly expanded my perspective on this research and on the computational astronomy field in general.

Non-academically, I would like to thank my family upon whom I rely to keep me sane. Their love and support has meant the world to me.

I am deeply grateful to my girlfriend, Niloufar Shiri, without whom I'm not sure I would have stuck around. Her support, dedication, and wonderful music has been invaluable.

Thanks to my friends at UCI, SDSU, (from) LLNL, and those back in Texas (unless already thanked per LLNL) for the many wonderful experiences we've had together. Their support has been invaluable and as the saying goes, I couldn't have done it without them all.

This project benefited greatly from the image data provided by several sky surveys, particularly the Sloan Digital Sky Survey (SDSS). Funding for the SDSS and SDSS-II has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, the U.S. Department of Energy, the National Aeronautics and Space Administration, the Japanese Monbukagakusho, the Max Planck Society, and the Higher Education Funding Council for England. The SDSS Web Site is http://www.sdss.org/.

The SDSS is managed by the Astrophysical Research Consortium for the Participating Institutions. The Participating Institutions are the American Museum of Natural History, Astrophysical Institute Potsdam, University of Basel, University of Cambridge, Case Western Reserve University, University of Chicago, Drexel University, Fermilab, the Institute for Advanced Study, the Japan Participation Group, Johns Hopkins University, the Joint Institute for Nuclear Astrophysics, the Kavli Institute for Particle Astrophysics and Cosmology, the Korean Scientist Group, the Chinese Academy of Sciences (LAMOST), Los Alamos National Laboratory, the Max-Planck-Institute for Astronomy (MPIA), the Max-Planck-Institute for Astrophysics (MPA), New Mexico State University, Ohio State University, University of Pittsburgh, University of Portsmouth, Princeton University, the United States Naval Observatory, and the University of Washington.

Support and funding has been provided by: the Computational Science Research Center at San Diego State University, the University of California Irvine School of Information and Computer Sciences, National Science Foundation (NSF) grant DUE-1259951, and the LSST-DA Data Science Fellowship Program, which is funded by LSST-DA, the Brinson Foundation, and the Moore Foundation.

VITA

Matthew Portman

EDUCATION

Doctor of Philosophy in Computational Science	**2024**
University of California, Irvine & San Diego State University	*Irvine & San Diego, California*
Bachelor of Science in Physics	**2016**
University of Texas at Dallas	*Richardson, Texas*

RESEARCH EXPERIENCE

Graduate Research Assistant	**2019–2024**
University of California, Irvine	*Irvine, California*
URA Visiting Scholar & Collaborator	**2021–2023**
Lawrence Berkeley National Lab	*Berkeley, California*
Graduate Research Intern	**2022, 2023**
Lawrence Livermore National Lab	*Livermore, California*
Graduate Research Assistant	**2017–2019**
San Diego State University	*San Diego, California*
Undergraduate Research Assistant	**2015**
Rochester Institute of Technology	*Rochester, New York*

TEACHING EXPERIENCE

Teaching Assistant	**2019–2024**
University of California, Irvine	*Irvine, California*

REFEREED JOURNAL PUBLICATIONS

A re-assessment of SpArcFiRe's performance on toy spiral galaxies **2023**
Monthly Notices of the Royal Astronomical Society

ABSTRACT OF THE DISSERTATION

Using SpArcFiRe to Help Automate GALFIT's Multi-Component Decomposition of Spiral Galaxies

By

Matthew Portman

Doctor of Philosophy in Computational Science

University of California, Irvine & San Diego State University, 2024

Associate Professor Wayne Hayes, Chair (UCI)

Spiral galaxies constitute a significant fraction of galaxies observed in the local universe yet their characteristic structure is not well understood. Current methods of analysis rely on manual intervention and expertise, both of which present a significant barrier to the investigation of spiral structure at the scale of modern observational surveys. We present an automated modeling method that uses the simple, one-dimensional arc analysis from `SpArcFiRe` to generate an initial guess for `GALFIT` to produce two-dimensional photometric decompositions of spiral galaxies. Our automated method is encapsulated in a Python module entitled `"GalfitModule"` which contains a generalized framework to handle `GALFIT` components and parameters. `GalfitModule` employs this framework to translate `SpArcFiRe`'s output to input for `GALFIT`, runs `GALFIT` using multiple novel techniques with distributed processing, and performs an assessment of the success of our decompositions. Using this pipeline, we produce two and three component decompositions of several samples of spiral galaxies from the SDSS DR7 data release, as selected by the Galaxy Zoo team. We then assess the performance of our method, validating our results by-eye, and analyze the resultant parameterization of these in bulk. Our largest sample is 28912 galaxies, of which we estimate 54% (about 15,700) of the models accurately map the visible structure of the original observations. We identify trends in the Sérsic indices, magnitudes, and arm-to-total flux ratios, and compare these trends to previous decomposition studies, finding general agreement in the arm-to-total flux ratio. Of the other parameters, there is evidence that our models overfit the observations, causing disagreement. Finally, we present an extension to our method which evaluates the model's pitch angle as it varies along the length of the arm.

Chapter 1

Introduction

Spiral galaxies comprise about 70% of galaxies observed in the local universe (Sparke & Gallagher, 2007) yet the formation of the characteristic arms and their visible structure are not fully understood beyond strict formalisms that describe their dynamics (Lin & Shu, 1964; Binney & Tremaine, 2008). To make progress on this front, it is necessary to produce objective, quantitative measures of their structure to compare with theory and models obtained from increasingly detailed cosmological simulations such as Illustris (Vogelsberger et al., 2014; Pillepich et al., 2018), and more local simulations such as FIRE (Hopkins et al., 2014, 2018) and NIHAO (Wang et al., 2015). As the amount of data continues to increase, the development of automated procedures to evaluate galactic structure is paramount.

In the context of morphological analysis, two types of analyses are most common: parametric and non-parametric fitting. Non-parametric fitting, being highly dependent on Signal-to-Noise (S/N) and therefore resolution *and* morphology, is less than optimal for such large and varied data sets (Lisker, 2008; Peng et al., 2010). Parametric fitting, on the other hand, models the light distribution of a galaxy by using analytic functions to fit individual features or components of the galaxy which allows us to decompose the galaxy into individual structures by their surface brightness distributions. Parametric fitting was first introduced by de

Vaucouleurs (1948) and expanded by Freeman (1970). The practice has since continued and expanded from one-dimensional analysis (Kormendy, 1977; Kent, 1985) to two-dimensional analysis (Byun & Freeman, 1995; Schade et al., 1996; Wadadekar et al., 1999; Benson et al., 2002) which have all laid the foundation for modern, analytical two-dimensional galaxy fitting programs such as `GIM2D` (Simard et al., 2002), `BUDDA` (de Souza et al., 2004), `PROFIT` (Robotham et al., 2016), and `GALFIT`3.0 (henceforth simply referred to as `GALFIT`, Peng et al., 2010). All provide the user with the ability to perform bulge+disk (B+D) decompositions—i.e. decomposing a galaxy into individual sub-structures—on images of galaxies; however, `GALFIT` is the only program which can extend the analytical fitting process to model spiral arms as well.

It is possible to employ spiral arm only decompositions or to use these in conjunction with existing B+D decompositions. The Fourier transform decomposition technique in one (Grosbøl et al., 2004; Kendall et al., 2011) and two dimensions (Seigar et al., 2005; Davis et al., 2012; Mutlu-Pakdil et al., 2018; Yu & Ho, 2018; Díaz-García et al., 2019; Hewitt & Treuthardt, 2019; Yu & Ho, 2020) is widely used but it fails to capture the variation in pitch angle found in observations of real galaxies (Ringermacher & Mead, 2009; Savchenko & Reshetnikov, 2013; Savchenko et al., 2020). Davis et al. (2012) and Savchenko & Reshetnikov (2013) propose solutions to this issue by either summing logarithmic spirals or by segmenting the Fourier analysis along the arm but neither of these techniques appear to have been widely adopted. Chugunov et al. (2023) proposes an entirely different technique—one which is more similar to that of `GALFIT`'s implementation—that takes an existing B+D decomposition and manually applies their own physically-motivated spiral arm function. As with previous attempts at using `GALFIT` for spiral arm decomposition, their process is heavily labor-intensive and expertise driven. As for non-expertise driven decompositions, the Galaxy Zoo team implemented the *Galaxy Builder* process (Lingard et al., 2020, 2021) which applies a logarithmic spiral based on traces drawn by human volunteers. Logarithmic spirals are *defined* by a constant pitch angle so this method again falls prey to the pitch angle variation problem.

The spiral arms that GALFIT creates are not restricted by a constant pitch angle and can be applied to existing B+D decompositions but its implementation is best suited to modeling two or more arms at a time (see Section 2.2.2). Previous attempts to model spiral arms using GALFIT have been performed manually (Läsker et al., 2013; Gao & Ho, 2017) however the manual fitting process is labor-intensive and requires significant expertise—it is not feasible to employ at the scale of observational surveys.

Attempts to automate GALFIT have been made previously (see Vikram et al., 2010; Kim et al., 2016; Domínguez Sánchez et al., 2021, among others specific to DR7 cited below), but none have successfully automated the modeling of spiral galaxies due in no small part to GALFIT's need for a well-conditioned initial guess on a per-image basis. To solve this issue, we propose a completely automated method which translates the output from SpArcFiRe (Davis & Hayes, 2014), a computer vision based spiral arc finding program, to an initial guess as input for GALFIT. We present our method using data from SDSS DR7 to coincide with prior analyses performed by SpArcFiRe in Davis & Hayes (2014), Silva et al. (2018), and Hart et al. (2016). Several other automated 2D decomposition studies have been performed on galaxies from DR7—Simard et al. (2011) performed bulge+disk decompositions of the photometric sample of the galaxies using GIM2D; Kelvin (2012) performed single component fits on a subsample of spectroscopically selected galaxies in DR7; Lackner & Gunn (2012) fit a sample of 70,000 galaxies at low redshift with B+D decompositions; and Meert et al. (2014, 2015) performed single and multi (bulge+disk) component fits for spectroscopically selected galaxies in DR7. The latter studies, and a B+D decomposition study performed by Kim et al. (2016) on SDSS DR12 galaxies, served as a baseline for many of our assumptions and initial choices made when developing the pipeline to GALFIT.

Since this is the first automated study of spiral galaxies using GALFIT, we aim to use these models as an opportunity to explore GALFIT's framework as an estimate of spiral galaxy morphological structure. We would like to emphasize here that the goal of this research is to produce models of the arms of spiral galaxies–we do not intend to make any strong claims about the accuracy of the resulting flux distribution, especially in the region of the bulge.

The chapters are laid out as follows: Chapter 2 details the programs that form the basis of the pipeline, `SpArcFiRe` and `GALFIT` in the context of our automated modeling method; Chapter 3 describes the translation strategy between `SpArcFiRe` and `GALFIT` that we employ in order to automate the fitting process; Chapter B provides documentation for the `GalfitModule`, the Python module (in development) which governs the operation of the pipeline and which we intend to release as a tool to the community upon its completion; Chapter 4 describes the fitting procedure the pipeline employs, from the scripts that control and execute the procedure, to the computational and algorithmic strategies we employ, to the validation scheme we have developed; Chapter 6 presents our results on several subsets of galaxies from SDSS DR7/GZ1; Chapter 7 provides a discussion of the results and validation method as well as comparisons to both existing B+D decomposition surveys on DR7 and smaller scale spiral galaxy decomposition studies; Chapter 8 summarizes our results; and finally, Chapter 9 describes the work we have planned for the release and improvement of our method as well as potential avenues for future study using our method. Additionally, we have included several appendices (A – F) which should aid in understanding our process and results.

Chapter 2

Methods

2.1 `SpArcFiRe`

`SpArcFiRe` (Davis & Hayes, 2014) is a computer vision algorithm designed to take images of real spiral galaxies as input, and automatically extract objective information about their structure. It has been used i) to provide quantitative information about spiral galaxy morphology on the GZ1 galaxies (Davis & Hayes, 2014), ii) to illustrate and explain S-wise spiral galaxy winding bias in the results of GZ1 (Hayes et al., 2016), iii) to demonstrate its proficiency (robustness and validation) in identifying structure across GZ1 and SDSS images (Silva et al., 2018), and iiii) by the Galaxy Zoo team to measure pitch angles (Hart et al., 2017) of the spiral galaxies in GZ2 and to subsequently assess the correlation between bulge size and arm winding via pitch angle (Masters et al., 2019).

The first step `SpArcFiRe` takes is to use an iterative 2D Gaussian fit to precisely center and isolate the target galaxy. The image is then deprojected to produce a face-on view of the galaxy and contrast enhanced before an orientation is assigned to each pixel. Based on the local strength and direction of the orientation, pixels are hierarchically clustered into regions with consistent logarithmic spiral shape—a common choice for describing spirals in galaxies (see Binney & Tremaine, 2008). Each cluster is then associated with a logarithmic spiral curve as determined by a least-squares fit before a final merging is performed using nearby arcs with compatible shape. See Davis & Hayes (2012) for a more in depth description of the algorithm.

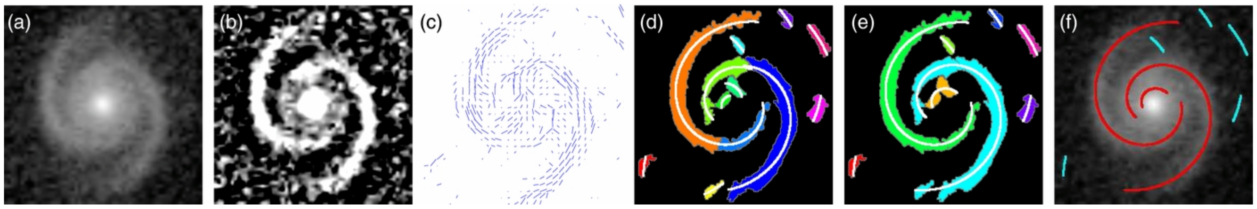


Figure 2.1: Steps `SpArcFiRe` takes to identify arcs in an image of a spiral galaxy. **a)** Auto-crop, centering, and de-projection. **b)** Contrast enhancement. **c)** Orientation field determination. **d)** Hierarchical clustering via consistent logarithmic spiral shape and orientation. **e)** Secondary merging to account for nearby shape. **f)** Final arc fit superimposed on *(a)*, red arcs denote S-wise winding while cyan arcs denote Z-wise. From this process, a CSV is produced which provides parameters from the fit: a length-sorted list of arms containing, for each arm, the initial and final polar co-ordinates, the signed pitch angle, and the area (in pixels) from the masks in figures *(d)* and *(e)* along with other image level information.

Figure 2.1 shows a visual representation of `SpArcFiRe`'s output at each step. Along with these images, `SpArcFiRe` also outputs information, to csv files, regarding approximate measurements of features of the galaxy, including bulge and disk radius, arm length in pixels, and arm beginning and end angle, all of which may be used to inform the input to `GALFIT` (see (Davis & Hayes, 2012) for a more in depth description of the program and (Davis & Hayes, 2014) for how it was tested). `SpArcFiRe`'s output is limited however, to a simplistic description of the galaxy's shape, i.e. the arm length, pitch angle, and area (in binary pixels). Although this information is useful on an image level, astronomers are unable to use it to compare theory with observation: it cannot, on its own, describe the *brightness* profile of a galaxy. More work is therefore necessary to extend `SpArcFiRe`'s functionality to better understand the complex physics of galactic morphology. For this, we can turn to `GALFIT` to model these galaxies in order to approximate their visible structure.

2.2 GALFIT

GALFIT (Peng et al., 2010) or P, 2010 is a parametric fitting program designed to model the light distribution of galaxies given a user-defined set of initial components and function parameters. It has seen frequent use in the astronomy community since its inception, having been used to model galaxies in many surveys including: GEMS (Bell et al., 2006), Spitzer (Sheth et al., 2010), CANDELS (Kartaltepe et al., 2015; van der Wel et al., 2012; Bouwens et al., 2015), the Carnegie-Irvine Galaxy Survey (Yu et al., 2018), and COSMOS (Dewsnap et al., 2023) among others.

In order to perform the galaxy decompositions, GALFIT takes the observation as input, convolves the image with the user-defined functions, and optimizes on the resulting model in order to minimize the difference between the model and the image (described in depth in 2.2.1).

2.2.1 GALFIT Components and Models

GALFIT uses the Levenberg-Marquardt technique to optimally fit user-specified functions to structures in an image. The algorithm works by minimizing χ^2 or the residuals between the data image and the model as given by the *normalized* equation,

$$\chi^2_\nu = \frac{1}{N_{\rm dof}} \sum_{x=1}^{nx} \sum_{y=1}^{ny} \frac{[f_{\rm data}(x,y) - f_{\rm model}(x,y)]^2}{\sigma(x,y)^2} \tag{2.1}$$

where x and y are image dimensions, $f_{\rm data}$ is the image flux at a pixel, $f_{\rm model}$ is the sum of the user-specified functions employed by the model, and σ is the Poisson error at each pixel. The minimization procedure is heavily dependent on the parameterization of the user-specified functions for some level of success, especially when tasked with fitting multiple components to an image.

GALFIT assumes light profiles are axially symmetric, generalized ellipses and uses a modified ellipse equation to define the radial pixel coordinate,

$$r = \left(|x|^{c+2} + \left| \frac{y}{q} \right|^{c+2} \right)^{\frac{1}{c+2}} \tag{2.2}$$

where c determines the shape of the ellipse and q is the ratio of the minor and major axes. GALFIT then uses a user-provided input radial light profile (functions of r, see Equation (2.3)) to create a model image before optimizing via Levemberg-Marquardt and outputting the final results.

One of the most common radial light profiles used to study galaxy morphology, and the function we use to model the bulge and disk of galaxies in our method, is the Sérsic powerlaw which has the functional form (Sersic, 1968):

$$\Sigma(r) = \Sigma_e \exp\left\{ -\kappa(n) \left(\left[\frac{r}{r_e} \right]^{1/n} - 1 \right) \right\} \tag{2.3}$$

where $\Sigma(r)$ is the surface brightness at a distance r from the center of the galaxy, r_e is the effective radius or the radius within which half of the total light of a galaxy is contained, Σ_e is the surface brightness at the effective radius, and n is the Sérsic index which shapes the light profile of the galaxy (P, 2010). In practice, GALFIT limits the value of n between 0 and 20 for both practical reasons and theoretical consideration of κ. In the limit of large n, we define κ as

$$\kappa \approx 2n - 0.331290 \tag{2.4}$$

while at low n, the value of κ is interpolated (Chien Peng, personal communication, 2020).

The Sérsic implementations in GALFIT employ either an integrated magnitude m_{tot} or the surface brightness magnitude μ_e which corresponds with Σ_e. For this study, we use the integrated magnitude—all references to magnitude throughout the paper refer to the integrated magnitude unless otherwise specified. The integrated magnitude is defined as:

$$m_{\text{tot}} = -2.5 \log_{10}\left(\frac{F_{\text{tot}}}{t_{\text{exp}}}\right) + \text{mag zpt} \tag{2.5}$$

where F_{tot} is the total flux associated with the Sérsic component, t_{exp} is the exposure time used to take the image, and mag zpt is the zero-point magnitude associated with the instrumentation.

These parameters along with the axis ratio (q) and position angle of the underlying ellipse are free parameters the user may modify in order to fit galactic features. Other radial light profiles are available in `GALFIT` and may be used to model the bulge and disk, however, some—the de Vaucouleurs (de Vaucouleurs, 1948), Exponential, and Gaussian functions—are special cases of the Sérsic profile at values of $n = 4$, $n = 1$, $n = 0.5$ respectively.

2.2.2 Spiral Galaxies in `GALFIT`

In the case of spiral galaxies, the Sérsic function, which models the disk, undergoes coordinate rotation as controlled by a powerlaw spiral (α-tanh) function coupled to the hyberbolic tangent function to account for the possible existence of a bar extending to the center of the galaxy. The exact functional form is given in P, 2010 (and can also be found in Chapter 7.2) but a schematic function outlining the available free parameters has the form:

$$\theta(r) = \theta_{\text{out}} \tanh\left(r_{in}, r_{\text{out}}, \theta_{incl}, \theta_{PA}^{sky}; r\right) \left[\frac{1}{2}\left(\frac{r}{r_{\text{out}}} + 1\right)\right]^{\alpha} \tag{2.6}$$

where r is the radial coordinate system, θ_{out} is the rotation angle at r_{out}, and α is the powerlaw index. r_{in}, referred to as the inner spiral radius throughout this paper, is the mathematical construct of a bar but it can be used to represent the radius of a physical bar, i.e. the radius at which the rotation relative to the major axis of the underlying ellipse reaches approximately 20 degrees—an empirical choice by P, 2010. θ_{incl} is the line-of-sight inclination of the disk where 0 degrees is face on and 90 degrees is edge-on. Finally, θ_{PA}^{sky} is the sky position angle which rotates the spiral in the plane of the sky. For a more in-depth discussion of each of these parameters, see P, 2010.

The fit of the arms can further be refined with the addition of Fourier Modes which modify the azimuthal shape by perturbing the coordinate system from a rectilinear grid. This choice ensures the preservation of the radial light profile while allowing us to modify the underlying ellipse in conjunction with the powerlaw. The Fourier perturbation on an ellipsoid is defined as follows:

$$r(x,y) = r_o(x,y)\left\{1 + \sum_{m=1}^{N} a_m \cos[m(\theta + \phi_m)]\right\} \tag{2.7}$$

where $r_o(x,y)$ is the radial coordinate of the ellipse and a_m is the Fourier amplitude for mode m. The phase angle, ϕ_m, is the angle of mode m relative to the position angle of the generalized ellipse - $0°$ in the direction of the semi-major axis and increasing counter-clockwise. The number of modes N is determined by the user and allows for non-specified modes to be skipped. Finally, θ is defined as

$$\theta = \arctan \frac{y - y_c}{q(x - x_c)}$$

where the right hand side is derived from Equation (2.2) but modified for generality with respect to the coordinate axes; (x_c, y_c) is the centroid of the ellipse (please refer to P, 2010 for more information). The Fourier modes, in effect, retain the structure and meaning (wherein prior intuition still applies) of the radial profile function thereby allowing for meaningful physical analysis (P, 2010).

By using SpArcFiRe, we can automate the procedure of producing an initial estimate for the parameters which describe the model as input for GALFIT. This is the basis of our pipeline. We discuss several other methods we employ which aim to improve the chances of producing a 'successful' model in Section 4 below.

Equations (2.1)–(2.7) form an incredibly complex system of parameters that makes the task of translating SpArcFiRe's output into suitable input nontrivial. Little in GALFIT's mathematical description directly correlates to parameters output by SpArcFiRe and both utilize different coordinate systems when executing their analysis. By first performing manual fits using easily resolved galaxies, we can develop an intuition for implementing an automated procedure and later utilize the intuition to inform parameter choices when translating between the two programs.

Chapter 3

Input to GALFIT

Both SpArcFiRe and GALFIT, as standalone programs, can be made to work in an automated fashion. However, translating between the two presents a challenge in overcoming the necessary conventions of both. Immediate considerations include translating between chosen coordinate systems, properly scaling SpArcFiRe's output, and determining what output should be converted as input to GALFIT. It should be noted that our automation pipeline is implemented on *top* of SpArcFiRe's current file hierarchy. Furthermore, the pipeline is designed to work completely independent of running SpArcFiRe, allowing the user to save computation time and to generate results on any data set previously generated by SpArcFiRe such as those from Davis & Hayes (2014) and Hayes et al. (2016)–we do not intend to use the pipeline to modify SpArcFiRe's capabilities. We therefore focus much of the following sections on ensuring GALFIT's success as it is highly dependent on an appropriate initial parameterization. Furthermore, we aim, with this research, to automate the process of modelling spiral arms with GALFIT with the expectation that the bulge and non-arm disk fit may be less accurate given the nature of multi-component decompositions. Many of the choices discussed below reflect this goal.

3.1 Header Information and Pre-Processing

The GALFIT header holds information for several control parameters needed to perform the image convolution–we will describe them in brief as informed by P, 2010 here. Items A and B indicate the input and output filenames. Items C–G and I are optional and allow the user to specify: C) a sigma image or noise map of the data, which we allow GALFIT to generate (see P, 2010 for information on how this is done), D) a PSF image to be convolved with the model, E) the PSF fine sampling factor which is used when the PSF has a finer pixel scale than the data, F) a pixel mask, G) a constraint file which indicates limits for fitting parameters, and I) the size of the fitting region/box used to convolve the PSF with each component. Item H is the image region (of the input image) to fit which crops the image prior to fitting. Items J and K, the magnitude zero-point and plate scale, refer to the instrumentation and are set according to the values found on the DR7 Website.[1] Item O allows for an interaction window and Item P indicates to GALFIT whether to perform the convolution or not. Several of the above items require pre-processing, including the input image and PSF from SDSS image fields, as detailed in Section 6.1, and the pixel mask and region to fit.

To create pixel masks for the images, we use SourceExtractor (Bertin & Arnouts, 1996) by the same method that SpArcFiRe applies during its pre-processing stage. This process is summarized in Davis & Hayes (2014) and detailed in Davis (2014) but by re-using the same method, we can retain consistency between SpArcFiRe's analysis and GALFIT's, an important consideration when the images are dynamically centered and cropped.

To determine a fitting region on the images, we turn to SpArcFiRe's analysis, specifically its ellipse fitting routine. When SpArcFiRe processes a galaxy, it iteratively fits a 2D Gaussian ellipse to the galactic disk to estimate its size and to identify the center position of the galaxy

[1]Further discussion, and the value used for the magnitude zero-point, can be found at https://classic.sdss.org/dr7/algorithms/fluxcal.php

in the image (see Figure 1 of Davis & Hayes, 2014). Under the assumption that galaxy disks are roughly circular (Sparke & Gallagher, 2007), `SpArcFiRe` determines the 3D inclination of the galaxy and transforms it to be face-on by stretching the ellipse into a circle and cropping the image to a square with the circle inscribed (Davis, 2014). We extend the box out to two times the circle's radius in order to assist `GALFIT` in determining an appropriate value for the background sky as suggested in the 'Technical FAQ' section of `GALFIT`'s website. We discuss approximating the background sky more thoroughly in Section 3.3.2. Note that none of these transformations are applied to the original input image–when we process a galaxy through the pipeline, we use the observation as-is.

Finally, we opt to avoid the inclusion of a PSF file as we have not yet developed a proper automated method for its preparation according to the recommendations on `GALFIT`'s website. Given our scientific goals, we do not believe this to be an issue with our analysis of the results. We additionally avoid using a constraint file in order to allow increased flexibility and algorithmic stability per recommendation 11 on `GALFIT`'s website. We have previously attempted to incorporate a constraint file but found little impact on the results for our scientific goals.

3.2 Morphological Parameterization

In an effort to minimize and standardize the vast and potentially degenerate parameter space available when fitting galaxies, we choose to model the structure of the galaxies with as few components as possible. In our case, and in extending the conventions set forth by previous automated studies (such as Meert et al., 2014; Kim et al., 2016; Domínguez Sánchez et al., 2021), we opt to model the bulge, disk, and arms of spiral galaxies by two – three components, each represented by a Sérsic profile for additional flexibility when performing the optimization step. For the arms, `GALFIT` can apply a power law coordinate rotation, described in Section 2.2.2, to a chosen profile in order to mimic the spiral winding of the arms. When using two components, bulge and disk, we apply the coordinate rotation to the disk component. When using three components, we choose to initialize a Sérsic profile with a nearly identical parameterization to that of the disk and apply the coordinate rotation to create the arms—we therefore do not specify the parameterization for this profile separate from the disk itself except where necessary. In addition to the bulge, disk, and spiral arm components, we use two Fourier modes (F1, F3), coupled to the power law function, to refine the shape of the arms, and a final sky component to fit the background. These methods are described as two and three component fits (NC2 and NC3 respectively) throughout the paper, where the coordinate rotation and fourier transform are implicitly applied to the 'last' Sérsic component and taking the sky component as also implicit. Although both the NC2 and NC3 structures provide an intuitive approach to modeling spiral galaxies, it is not necessarily the case that the components are physically meaningful on their own (Peng et al., 2010; Meert et al., 2013, 2014). Furthermore, `GALFIT` will sometimes modify one component to overcome the shortcomings of another to better minimize the χ^2_ν which ultimately creates a model of a galaxy that is not separable by the bulge-disk-arm convention. This is described as 'overfitting' in Meert et al. (2013, 2014). We will discuss the implications that arise from these issues in Section 7.

Figure A.2 shows an example input text file which we refer to (generally) as a 'feedme' file—feedme files are the input `GALFIT` uses to perform the convolution. Component 1 represents the bulge, component 2, the disk, and component 3, the disk used by the power law rotation function—the spiral power law function and Fourier modes are not output as distinct 'components' but are represented by the 'R' and 'F' parameter labels respectively. Finally, component 4 corresponds to the background sky. Two numbers describe each parameter: the value, and a binary (0/1) number which indicates to `GALFIT` whether to optimize on the parameter or to hold it fixed to the value given at input. Working from Figure A.2, which follows the template we used for the galaxies in this paper, we have at most 26(+2) free parameters, with the x, y position parameter of each component held fixed throughout the duration of fitting. The spiral inner (R1) and outer radii (R2) are held fixed prior to the final step as discussed in Section 4.3, hence the 2 additional free parameters.

3.3 `SpArcFiRe` to `GALFIT` (S2G)

`SpArcFiRe`, as described in Section 2.1, provides limited morphological information about spiral galaxies but offers a wealth of image-level measurements which can be used to estimate `GALFIT` input parameters. Table 3.1 shows the correlations we have chosen between the two programs. Length measurements must be scaled according to the ratio of the cropped image and the size of the original image fed into `SpArcFiRe`. We introduce two scale factors when transforming the length quantities, calculated as the ratio of the cropped image to the size of the either the original image or the size of `SpArcFiRe`'s standardized image, 256 x 256 pixels, and use whichever scale factor is appropriate to the selected quantity as dictated by Davis (2014). Angle measurements are also corrected to accommodate differing coordinate conventions and may otherwise be transformed to account for known behavior in `SpArcFiRe`. For a visual representation of the `SpArcFiRe`'s arc parameters in Table 3.1, please refer to Panel F of Figure 2.1. Also note that `SpArcFiRe` does not provide any information which can be used to initialize the magnitudes of the bulge and disk in our model. For the bulge, we use a parameter search method, detailed in Section 4.4, and initialize the disk to the value of the r-band Petrosian magnitude `petromag` provided by SDSS DR7.

In the following sections, we will present and discuss the transformations used to create the input to `GALFIT`. We follow the general ordering of the parameters in each component per Table 3.1 and A. The code that performs these transformations is entitled `sparc_to_galfit_feedme_gen.py`.

Feature (GALFIT Component)	SpArcFiRe Output	GALFIT Parameter
	inputCenterR, inputCenterC	Position x, y
	CropRad	Image region to fit
Bulge (Sérsic 1)	bulgeMajAxsLen	R_e (Effective Radius)
	bulgeAxisRatio	Axis ratio
	bulgeMajAxsAngle**,	Position angle
Disk (Sérsic 2/3)	diskMajAxsLen	R_e (Effective Radius)
	diskAxisRatio	Axis ratio
	diskMajAxsAngle**	Position angle
Arms (Power)	r_start*	Spiral Inner Radius
	r_end*	Spiral Outer Radius
	math_initial_theta*, relative_theta_end*	Cumulative Rotation Out
	pa_alenWtd_avgdomChiralityOnly	α (Powerlaw Exponent)
	chirality_maj, chirality_alenWtd	Spin Direction
	diskAxisRatio, diskMajAxsAngle*	Inclination
	diskMajAxsAngle*	Sky Position Angle

Table 3.1: Correspondences between SpArcFiRe output and GALFIT input parameters (Appendix A) and the galactic feature the parameters will be used to model. For more information regarding the meaning of the SpArcFiRe parameters, see Davis & Hayes (2014). *From arc-level output. **SpArcFiRe measures the bulge and disk angles as the closest angle to 0 so the resulting angle found always lies within the I and IV quadrants.

3.3.1 S2G – Bulge & Disk

The effective radii (or half-light radii), as informed by `SpArcFiRe`, are first scaled by the scaling transformation described in Section 3.3. For the non-rotated disk, and for NC3, we reduce the effective radius slightly given that `SpArcFiRe` measures the major axis length of the ellipse to completely encompass the arms which sometimes causes `GALFIT` to over-extend the disk and lose precision in the arm fit. In general however, we find that `SpArcFiRe`'s measure of axis length to be, at worst, a slight overestimate of the effective radius but generally close enough that no further adjustment is necessary. For the Sérsic index of the bulge, we initialize its value to that of the classic de Vaucouleurs profile, i.e. $n_{\text{bulge}} = 4$ (de Vaucouleurs, 1948; Peng et al., 2010). In testing this choice, we find that `GALFIT` tends to drive the index lower to accommodate for using the disk to create the arms (hence 'overfitting' the model to the galaxy) but do not find significant improvement when initializing the index at a lower value. For the disk, we initialize the Sérsic index to that of the exponential profile, i.e. $n_{\text{bulge}} = 1$.

For the position angles, `SpArcFiRe` uses standard mathematical convention and measures angles with respect to zero on the x-axis unless the measurement is expressly relative to another while `GALFIT` defines the position angle (of the semi-major axis on-sky) to be relative to Celestial North, i.e. zero on the y-axis. The difference in convention is complicated by the fact that `SpArcFiRe` restricts the bulge and disk position angle measurements to $\pm 90°$ on the assumption that both are symmetric about the origin. This behavior however simplifies the transformation–we only have to subtract `SpArcFiRe`'s measurement from $90°$. This comes with the additional advantage that large angular measurements do not seem to play well with `GALFIT`'s optimization process.

The axis ratio of the bulge is used directly without scaling due to the length invariance of the ratio. For the disk, we may elect to avoid fitting arms to an image of a galaxy as determined by a max arm length threshold (via `SpArcFiRe`) of 75 pixels from the original image. In this case, we use the disk axis ratio as-is. When we do incorporate the spiral arm rotation function, the axis ratio directly affects GALFIT's ability to model the arms as this second Sérsic profile is ultimately what undergoes the coordinate rotation—as a description of the underlying ellipse, the axis ratio therefore directly represents the width of the arms. The inherent trade-off is that the more elliptical the profile, the more well-defined the arms are, the less this component is able to accommodate for disky, inter-arm light. This trade-off is the main crux of the difference between the NC2 and NC3 methods. For some galaxies, there is not enough inter-arm or extended disk light to warrant an additional component, especially at the resolution scale of SDSS DR7, while for others (when no additional disk is supplied), GALFIT may sacrifice either the bulge or arm fit in order to properly model the rest of the disk. We will discuss the difference in results from using the two component methods in Section 6.

3.3.2 S2G – Spiral Arms

The SpArcFiRe parameters which inform the power function (2.6) undergo a variety of transformations as few directly correlate to a purely mathematical description of spirals, with the exception of SpArcFiRe's determination of chirality. Furthermore, SpArcFiRe outputs arm measurements on a length-sorted, per-arc basis, which uses an empirical threshold to merge nearby clusters (panel d in Figure 2.1), but which is also agnostic to a preferred winding direction. This all implies that great care must be taken in the selection and transformation of arc measurements to usable GALFIT parameters. We also operate on the assumption that most spiral galaxies will have two distinct, bright arms—duly appropriate given GALFIT's coordinate rotation implementation of an ellipse—and allow for Fourier modes to assist in handling additional arms when this is not the case. In the following paragraphs, we will discuss first the parameters taken from SpArcFiRe's galaxy level output before continuing with those from the arc-level output as indicated by a * in Table 3.1.

In the power function, the α parameter can be described by the relation:

$$\theta \propto r^{\alpha} \rightarrow \alpha \propto \log_r \theta \tag{3.1}$$

as derived from the pure power law spiral formulation which `GALFIT` adopts beyond θ_{out}. Based on both this mathematical description of α and on the description and graphical representation of (varying) the parameter in P, 2010, we use an empirically derived correlation between α and the length-weighted pitch angle of the galaxy. Furthermore, we expect α to be between 0 and 2 but find that `GALFIT` has a tendency to drive α negative when the cumulative rotation is inaccurate. This produces a galaxy with arms which rotate in one direction out to the outer radius and then switch directions beyond—a behavior which is often undesirable. We have not yet found a `SpArcFiRe` parameter that better correlates to α although we are exploring the relationship between `GALFIT`'s power law spiral formulation and `SpArcFiRe`'s logarithmic implementation. `GALFIT` is relatively sensitive to the initial choice of α, especially when α is small, so we attempt to ensure a successful fit by limiting our input (note: *not* the `GALFIT` parameter itself) to be greater than 0.75.

The inclination of the galaxy, or the tilt in the plane of the sky, is directly related to the axis ratio of the ellipse which fits the galactic disk and can be retrieved by taking the arccosine of this ratio. Although the transformation is simple, the arccos function does not allow us to retrieve the sign of the original angle so we use the same parity as that of the `diskMajAxsAngle`, which influences how `SpArcFiRe` determines which side of the galaxy is into/out of the plane of the sky—this is done after accounting for `SpArcFiRe`'s rotation about the origin and thus avoids the need for additional considerations.

The sky position angle, as defined by `GALFIT`, is measured with respect to zero on the x-axis, which corresponds directly with `SpArcFiRe`'s convention. It operates in the same fashion to that of the Sérsic position angle in that it rotates the spiral (disk) in the plane of the sky and does so in addition to the rotation from the disk position angle since the power law directly modifies the disk profile. We overcome this degeneracy by subtracting the transformed disk position angle from SpArcFiRe's measure of the disk angle and apply a modulo 180° operation to avoid too-large numbers, before taking the result. It is tempting to avoid this process by initializing the sky position angle to zero but we found that GALFIT does not seem to perform as well with no rotation.

Of the arc-level output, we ingest the file as-is to iterate through the arcs in decreasing length-sorted order. If the winding direction of an arc disagrees with the winding direction of the longest arc, we skip that arc and continue. This approach is agnostic to the previously found (in the code) spin direction in order to prioritize—on the assumption of the dominance of two-arm galaxies—well-resolved measurements which naturally arise from the longest arcs. Each parameter is transformed for scaling and to account for SpArcFiRe behaviors and weighted according to their position in the (winding direction) order before we reach a final value to be input to GALFIT. Similar to the galactic measure of the pitch angle (pa_alenWtd_avgdomChiralityOnly), this weighting is a length weighted, dominant chirality estimate. It should be noted that we scale the arc parameters to be slightly smaller than the direct scaling relation—in prototyping the pipeline, GALFIT seemed to favor shorter lengths, possibly to accommodate for the extension of light along the underlying ellipse. r_end is the simplest to transform and is done so without further considerations. We detail the considerations and transformation of r_start and math_initial_theta, relative_theta_end below before discussing the final component related to the spiral arm component, the Fourier modes.

The spiral inner radius formally functions as the mathematical construct of a bar (P, 2010) (see also Section 2.2.2) and in practice mimics the light profile of a bar quite well.The edge of a bar is, however, an excellent proxy for the beginning of the spiral arm in a model and is treated as such in this study. With this in mind, SpArcFiRe's r_start parameter correlates nicely with `GALFIT`'s representation of the spiral inner radius–for a discussion on how `SpArcFiRe` handles bars, please see Davis & Hayes (2014); Davis (2014). Furthermore, we use the `r_start` parameter to identify when `SpArcFiRe` extends an arc too far into the bulge, by comparing to the previously found bulge radius, and heavily punish the resulting measurements to accommodate.

The cumulative rotation out parameter represents the rotation angle roughly at the spiral's outer radius and controls the chirality of the galaxy, S-wise is positive and Z-wise negative. This parameter is unique in that it is measured with respect to the angular position of the beginning of the spiral as opposed to having a fixed angular origin. `SpArcFiRe` outputs information regarding both the initial angular measurement of the arm as well as the angular distance swept out by the best-fit logarithmic spiral function with no restriction on full (360°) or beyond full rotation. This means that `SpArcFiRe` can and often does measure the beginning or the end of an arm well beyond 360° by design. In the case of the cumulative rotation parameter, the greater the value, the more winding occurs until the rotation reaches that angle, again with no restriction to 360°. To avoid over-winding then, we take the value of both `math_initial_theta`, `relative_theta_end` modulo 270°, as opposed to 360° to account for over-determination due to `SpArcFiRe`'s weighting scheme, and weight as described above.

The Fourier modes are an inessential but useful tool for refining the fit of the spiral arms by modeling the higher order features of the galaxy. We can tune the amplitude of each mode to emphasize the arms with a direct correspondence between the number of the mode and the number of arms (plus one) we should expect the model to have–see Figure 9 from P, 2010 for a simple demonstration of this phenomena. This is complicated by a degeneracy between the $m = 2$ mode and the axis ratio of the underlying ellipse (P, 2010) so we opt to instead use the $m = 1$ and 3 modes, initializing the latter with a much smaller amplitude. We have found no means of correlating `SpArcFiRe` output to the value of the amplitudes and phase angles but choose initial values which generally reflect the values `GALFIT` has produced in hand-fit test models. We have not found `GALFIT` to be sensitive to this initial parameterization.

The final component of the model to fit is the background sky. When fitting galaxies for photometric accuracy, this component is of great importance, especially when using a Sérsic profile with a high ($n \gtrsim 4$) Sérsic index (Guo et al., 2009; Peng et al., 2010) due to the extended wings produced by a Sérsic profile with a high n. In our case, this may occur when `GALFIT` determines that a high Sérsic index value is necessary since we opt to initialize each Sérsic profile with the canonical value associated with the bulge and disk. The `GALFIT` webpage recommends providing an estimate of the sky background where possible but explicitly recommends against using `SourceExtractor` to avoid introducing systematic errors as it has been shown to overestimate this background (Häussler et al., 2007). The paper by Häussler et al. further emphasizes that `GALFIT`'s estimation of the sky background returns more reliable results than `SourceExtractor` when the component is left free for `GALFIT` to optimize on. This can be seen most clearly in Figure 15 of Meert et al. (2013) which shows that `GALFIT`'s ability to determine the sky background is indeed robust. We take this approach and use an initial estimate of 1000 ADUs [counts] to accommodate the SDSS soft bias,[2] and allow `GALFIT` to optimize on this component. The choice to use a 'global sky background' is supported by previous studies, specifically those by Von Der Linden et al. (2007) and Guo et al. (2009) which performed their analysis on galaxies from SDSS DR4 (Adelman-McCarthy et al., 2006). The former asserts that the global value is a good enough approximation of the sky to study the structure of galaxies while the latter demonstrates the sensitivity of the Sérsic index when choosing this value. We note here that the approach in Guo et al. differs from our own in that they opt to hold the component fixed so the systematic uncertainties they find may not be wholly applicable. We would also like to emphasize that sky values are provided by SDSS (if still underestimated, Blanton et al., 2011) but we do

[2]https://classic.sdss.org/dr7/algorithms/fluxcal.php#assessment

not think they are necessary to satisfy our scientific goals given `GALFIT`'s robustness by this approach.

Chapter 4

Fitting Procedure

As discussed in Section 3, our pipeline is completely modular and can be run after `SpArcFiRe` has performed its analysis. The pipeline can be run in either serial or parallel via distributed computing (on-node multi-processor or cluster) in which case the outputs and procedure are slightly different. These differences are discussed in detail in Section B and in Section 4.5. The fitting process is generally as-follows: 1) locate input FITS files and perform logistical checks, 2) check for star masks from `SpArcFiRe` and if not found, generate star masks prior to fitting, 3) generate feedmes, 4) run `GALFIT`, 5) analyze FITS and output results. In addition to the results, we output PNG files which tile the observation, model, and residual using `fitspng` (F. Hroch, 2019) and `ImageMagick` (The ImageMagick Development Team, 2020).

The pipeline is primarily composed of three scripts—one script controls the conversion from `SpArcFiRe` to `GALFIT` parameters, while the other two, a control script and a 'run' script, generally perform the set-up, calls to `GALFIT`, preparation for analysis, and clean-up. Figure 4.1 shows a flow chart of the control script (`control_script.py`) and Figures 4.2 and

4.3 show the flow chart of the 'run' script (`go_go_galfit.py`) for the different component initializations, 2-component and 3-component.

4.1 `control_script.py`

`control_script.py` is the script which sets-up the pipeline and calls the programs used to perform the fitting and subsequent processing. A flow chart which summarizes the script's operation can be seen in Figure 4.1.

`control_script.py` begins by first ingesting command-line arguments which specify set-up and run conditions. We would like to highlight a few such command line arguments here:

- `-p | -parallel` – indicates whether to run in serial (option 0) or parallel (options 1 and 2). Option 1 corresponds to on machine parallel via the parallel driver `parallel`. Option 2 corresponds to distributed computing via SLURM via parallel driver `distrib_slurm`. Both parallel drivers can be found in the `GalfitModule` under the `ParallelDrivers` directory. '1' is the current default.

- `-NS | -num-steps` – indicates the number of steps to use in the multi-step fitting process, Section 4.3. The number of steps is allowed to range from 1–3 and corresponds with fitting different components per step depending on the number of components selected Section 3.2. '2' is the current default.

- `-NC | -num-components` – indicates the number of components to use in the fitting process, Section 3.2. '2' corresponds to the 2-component method and '3' the 3-component method. '3' is the current default.

- `-r | -restart` – indicates whether to restart the pipeline assuming some failure or early stoppage.

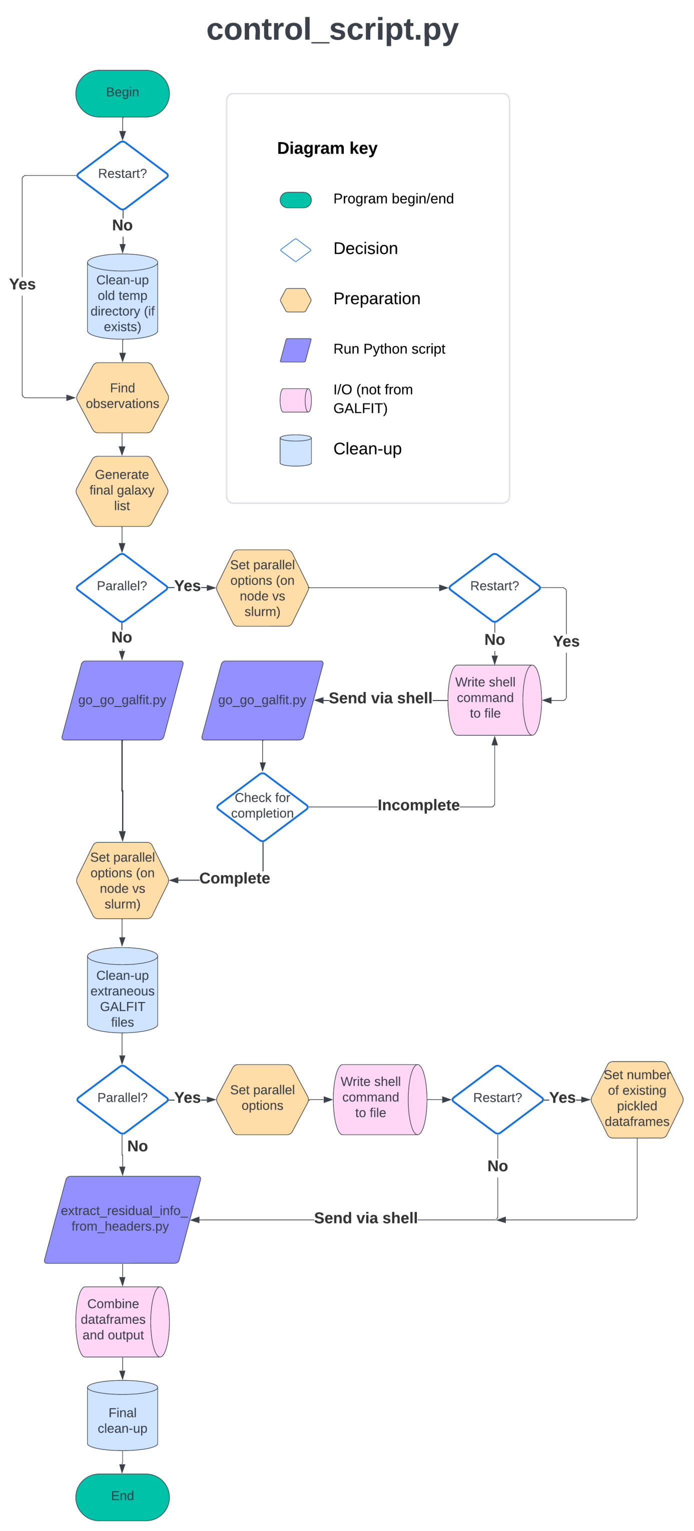

control_script.py
Begin
Restart?
No
Yes
Clean-up old temp directory (if exists)
Find observations
Generate final galaxy list
Parallel?
Yes
No
Set parallel options (on node vs slurm)
Restart?
No
Yes
Write shell command to file
Send via shell
go_go_galfit.py
go_go_galfit.py
Check for completion
Incomplete
Complete
Set parallel options (on node vs slurm)
Clean-up extraneous GALFIT files
Parallel?
Yes
No
Set parallel options
Write shell command to file
Restart?
Yes
No
Set number of existing pickled dataframes
extract_residual_info_from_headers.py
Send via shell
Combine dataframes and output
Final clean-up
End
Diagram key
Program begin/end
Decision
Preparation
Run Python script
I/O (not from GALFIT)
Clean-up

Figure 4.1: A flow chart detailing the operation of the `control_script.py` script.

- `-n | -basename` – dictates the name of the results folder, found in the `-out` directory, and the prefix for the results. The basename defaults to “GALFIT”.

The paths to the directories populated by `SpArcFiRe` may also be specified. If paths are not supplied, the script defaults to using `sparcfire-in`, `sparcfire-tmp`, and `sparcfire-out`. After these arguments are handled, the script cleans and creates the directories necessary to use the pipeline. We then build a master galaxy list from the observations in the input directory. At this point in the program, we can begin the fitting process. If the `parallel` option is set to 0, `go_go_galfit.py` is run in serial. For the distributed fitting process (Section 4.5), we begin by setting the `ParallelDriver` options and chunk size, i.e. the number of unique observations to be sent to each call to `go_go_galfit.py`. Currently and for the results in this study, our code utilizes a chunk size of 10. In our experience, 10 observations seems to provide a good balance between limiting the heavy I/O involved in the fitting process—especially when the number of compute nodes is high (e.g. when using SLURM—and accommodating bad observations or otherwise poorly initialized models which can greatly increase the overall runtime. We then prepare the text file to be piped to the `ParallelDriver` and run the selected driver via a python subprocess shell.

If the `parallel` `ParallelDriver` is selected (option 1), both `STDERR` and `STDOUT` from `go_go_galfit.py` are captured. Given this, and although this option is the default, we recommend for users to first run the code in serial to confirm that their files and directories are set-up correctly before attempting to perform the fitting procedure in parallel. We furthermore do not recommend using this option for very large runs as miscellaneous errors do pop-up from time to time and the capturing of `STDERR` makes these errors nearly impossible to debug.

If `distrib_slurm` `ParallelDriver` is selected (option 2) as the parallel driver, errors and `STDOUT` are output to files corresponding with node and process name in a subdirectory, `GALFITTING`, of the affectionately named `SLURM_TURDS` directory in the user’s home directory. We highly recommend that users of this pipeline check the `.err` files in this directory periodically to monitor the progress of the fitting procedure.

After all chunks have been sent to `go_go_galfit.py`, we check whether the number of output model files (including failures) matches the total galaxy count found during set-up. If fitting errors occurred or a bug otherwise prevented a chunk from being processed, the remaining galaxies are re-chunked and fed back to `go_go_galfit.py` to be processed again with a notification of the number of galaxies remaining printed to `STDOUT`. We set an upper limit on this processing loop to 10 attempts. If an unmanageable error occurs, this is most often evidenced by what appears to be a fixed number of galaxies remaining, as reported to the user, throughout the duration of the re-processing loop. When all model or failure files have been successfully produced, the fitting process is complete.

The final step `control_script.py` takes is to extract the `GALFIT` parameterization and residual analysis, performed in `go_go_galfit.py`, Section 5.1, from the output `FITS`. We again take advantage here of our distributed computing scheme and split the results into chunks, combining the individual galaxy data together (per chunk) into a `pandas` dataframe, saving to disk, and then combining all chunks into a single output file. This output file is dropped in the "basename" folder located in the output or `-out` directory with the name `[basename]_output_results.pkl`. If multiple runs are performed with the same name, the script appends an iterative number before the extension so as not to overwrite previous results. This is accommodated in the results analysis (Section 5.3) as the analysis framework always uses the most recent output, i.e. the output with the highest number. Once this is complete, we tarball the final fits to the "basename" directory.

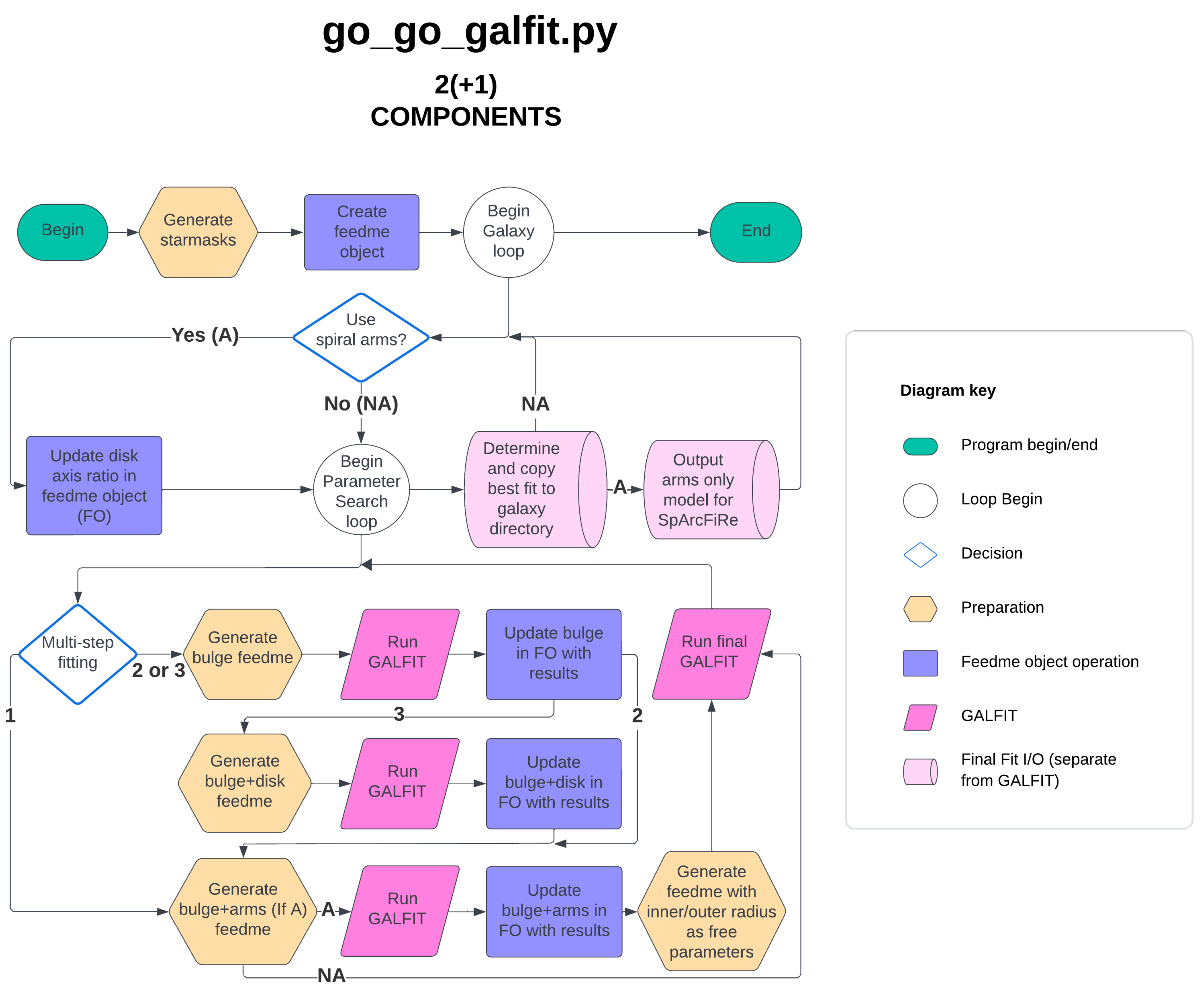

go_go_galfit.py
2(+1)
COMPONENTS
Begin
Generate starmasks
Create feedme object
Begin Galaxy loop
End
Use spiral arms?
Yes (A)
No (NA)
NA
Update disk axis ratio in feedme object (FO)
Begin Parameter Search loop
Determine and copy best fit to galaxy directory
A
Output arms only model for SpArcFiRe
Multi-step fitting
2 or 3
1
Generate bulge feedme
Run GALFIT
Update bulge in FO with results
Run final GALFIT
3
2
Generate bulge+disk feedme
Run GALFIT
Update bulge+disk in FO with results
Generate bulge+arms (If A) feedme
A
Run GALFIT
Update bulge+arms in FO with results
Generate feedme with inner/outer radius as free parameters
NA
Diagram key
Program begin/end
Loop Begin
Decision
Preparation
Feedme object operation
GALFIT
Final Fit I/O (separate from GALFIT)

Figure 4.2: A flow chart detailing the operation of the `go_go_galfit.py` script for the two component fit method.

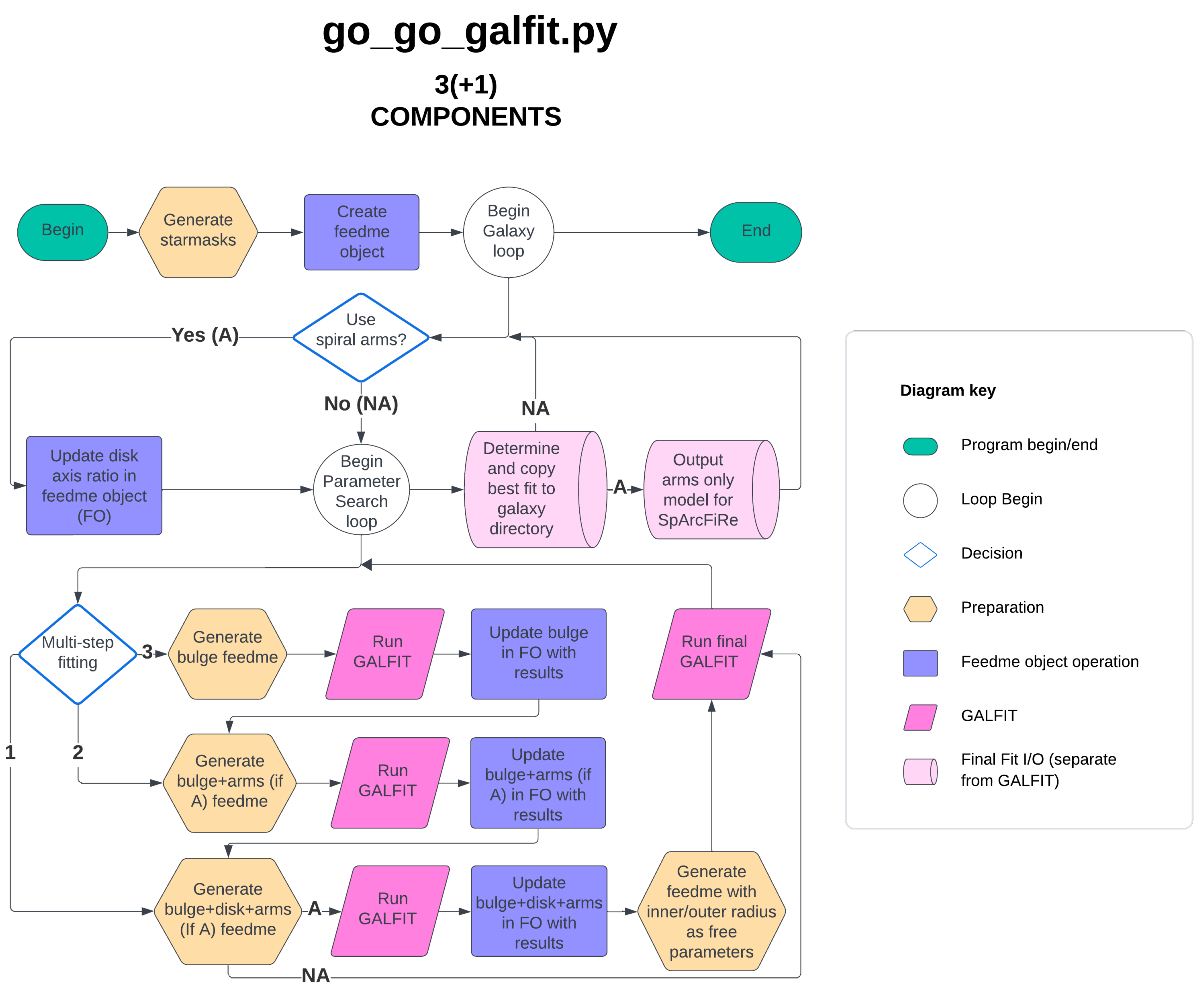

go_go_galfit.py
3(+1)
COMPONENTS
Begin
Generate starmasks
Create feedme object
Begin Galaxy loop
End
Use spiral arms?
Yes (A)
No (NA)
NA
Update disk axis ratio in feedme object (FO)
Begin Parameter Search loop
Determine and copy best fit to galaxy directory
A
Output arms only model for SpArcFiRe
Multi-step fitting
3
Generate bulge feedme
Run GALFIT
Update bulge in FO with results
Run final GALFIT
1
2
Generate bulge+arms (if A) feedme
Run GALFIT
Update bulge+arms (if A) in FO with results
Generate bulge+disk+arms (If A) feedme
A
Run GALFIT
Update bulge+disk+arms in FO with results
Generate feedme with inner/outer radius as free parameters
NA
Diagram key
Program begin/end
Loop Begin
Decision
Preparation
Feedme object operation
GALFIT
Final Fit I/O (separate from GALFIT)

Figure 4.3: A flow chart detailing the operation of the `go_go_galfit.py` script for the three component fit method.

4.2 go_go_galfit.py

go_go_galfit.py is written to be called from command line to ensure compatibility with the distributed computing scheme we employ, Section 4.5. It takes in, as command line arguments, setup information as well as the chunk of identifying galaxy names to be fit. go_go_galfit.py begins the fitting process by generating starmasks via remove_stars_with_sextractor.py, a slightly modified version of the starmasking code used by SpArcFiRe, and GALFIT feedmes, generated by sparc_to_galfit_feedme_gen.py, for the list of galaxies fed in. Notably, sparc_to_galfit_feedme_gen.py returns a dictionary of FeedmeContainer objects which contain the initial parameterization for all galaxies in the chunk. This object, referred to as the "feedme object" or "FO" in Figures 4.2 and 4.3, is ultimately what handles the GALFIT parameterization throughout the fitting process. go_go_galfit.py then prepares to run GALFIT itself.

The entire fitting process is detailed in the flowcharts in Figures 4.2 and 4.3, which demonstrate the process for two and three component fits (Section 3.2) respectively. These flowcharts include several fitting techniques developed for and employed in this study—multi-stage fitting, Section 4.3 and parameter search fitting, Section 4.4—but we will provide a brief overview here.

Two parameters are set before GALFIT is first run: the disk axis ratio of the component to be rotated to form the arms and the magnitude of the bulge in accordance with the parameter search. The former is set using a semi-empirical relationship between it and the probability of spirality, p_spirality, as reported by GZ1 in order to give GALFIT the best chance to generate arms (Section 3.3.2). The step-wise relationship boils down to: the higher the probability, the lower the disk axis ratio, the more well-defined the arms should be in the initialization given to GALFIT. Both these parameters are then fed into the parameter search loop so that the multi-stage fitting process can begin.

The parameter search loop is performed asynchronously (per galaxy) using python's `asyncio` library. Each asynchronous instance performs the entire multi-step fitting procedure for each iteration of the parameter search. To be clear, each galaxy fed in is processed in serial on-node—only the parameter search variations of each galaxy are processed asynchronously. For this reason, we have implemented a process watcher that monitors all instances of asynchronous subprocesses (`GALFIT`) in order to timeout the fitting procedure for models that are poorly initialized. After the completion of the multi-stage fitting procedure, we perform a residual analysis on the output and store the result for comparison against the other parameter search models. This is the final step of the multi-stage fitting procedure per parameter iteration.

After all models for a single galaxy have been generated, we find the model which minimizes our residual metric. We select this model as the "best" model, update its `FITS` header with the residual analysis, and copy it to its corresponding galaxy folder in the output directory. If `GALFIT` failed to produce a model for all iterations of the parameter search, a file entitled `failed_[galaxy name]_galfit_output.fits` is generated in the `-tmp` directory—but not copied to the galaxy folder—which holds all output `GALFIT` files. Once the successful fit has been determined, we create an arms-only model (if applicable) for `SpArcFiRe` by modifying the feedme to 1) turn off optimization and 2) only output the rotated disk component so as to improve the chances of `SpArcFiRe` successfully processing the model.

4.3 Multi-Stage Fitting

A unique feature of our pipeline is the ability to fit the model to the image in multiple stages corresponding to the NC2 and NC3 methods we use to generate the spiral, from one to four stages. For the first set of results in this paper, we perform a three-stage fit using the NC2 structure, in which we first fit our initial guess of the bulge and sky, then fit the resulting output with our initial guess for the rotated disk to form the arms. In the final stage of the fit, we allow the inner and outer radii of the power function to be free parameters in an attempt to refine the modeling of the arms.

For the second set of results, NC3, we perform another three-stage fit, in which we first fit our initial guess of the bulge, arm, and sky components, then fit the resulting output with the guess for disk, updating the disk magnitude with that of the rotated disk component. As with NC2, we perform a final fit, setting the inner and outer radii of the power function to be free parameters. We choose to fit the arms during the first step to aid `GALFIT` in retaining the arms when the disk is introduced—`GALFIT` has a tendency to use the light from the disk to cover the arms and this strategy appears to be the best way to mitigate that.

We motivate the choice to start both procedures with the bulge on the assumption that our initial guess is imperfect but that the bulge will provide the most reliable starting point for the rest of the convolution given the density of light in that region and `GALFIT`'s inherent sensitivity to flux differences between observation and model. In the initial development of the multi-stage procedure, we attempted to fit the disk first on the dual assumption that `GALFIT` would be more successful in fitting the arms of the galaxy having been given the chance to fit them pre-rotation by a pure disk profile and that the components in the final fit would be more easily separable. However, we did not find either of these assumptions to be the case and in comparing the two methods, found that the bulge-first procedure resulted in a higher success rate overall.

There is no guarantee that each stage of the multi-stage fit will work. If this is the case, we resume with the following stage and allow `GALFIT` to proceed with the non-failure outputs and the initial input(s) all at once. If only the first stage succeeds, we proceed with our analysis taking the galaxy as a single component fit but otherwise, nothing changes in our analysis. A new feedme is generated at each stage, named for the component(s) being fitted (with the exception of the final feedme entitled `GalaxyID.in`), and retained, if specified, for each galaxy. Subsequent FITS outputs are not retained but can be generated again using the appropriate feedme.

As mentioned in Section B, the multi-stage fitting procedure is handled by the `OutputContainer` class from the `GalfitModule`, written to ingest `GALFIT` output text from console (STDOUT). The ability to do so without file I/O is a key feature of our approach, especially in the context of parallelizability. An important byproduct of using the `OutputContainer` class to handle output and subsequent feedme generation is that the components are always output in the same order to the feedme file. In prior versions of the multi-stage fitting code, we noticed result and runtime discrepancies when `GALFIT` attempted to convolve images with components in a different order, i.e. sky before disk or disk and arms before bulge, and the additional processing necessary to avoid this, while maintaining the flexibility of the current approach, was impractical. This directly motivated the development of the `OutputContainer` class, and the `GalfitModule` as a whole.

4.4 Parameter Search Fitting

In addition to the multi-stage fitting, our pipeline incorporates a parameter search method into the fitting process. This is achieved on initialization of the multi-stage fitting procedure in `go_go_galfit.py`—we set the desired value of the parameter, modify the name of the output model accordingly, and substitute these into the feedme generated from the S2G procedure, before running the full multi-stage fit for each iteration of the parameter. We then compare the resulting output using the residual analysis methods detailed in Section 5.1 below, and choose the result which minimizes the difference between the observation and model. For this study, we perform our parameter search on the bulge magnitude, initializing this magnitude to the `petromag` provided by SDSS DR7, and search above and below this value in steps of 1 magnitude.

The parameter search method can be computationally expensive as it requires `GALFIT` to be run repeatedly on the different initializations in order to search the parameter space. Care must be taken when selecting the parameters and values to search. For this study, we utilize this method on the bulge magnitude having found `GALFIT` to be the most sensitive to this parameter in the context of our component selection. We attempted to perform the parameter search on the disk magnitude along with the bulge but did not find significant improvement in the results especially when considering the twofold increase in runtime. The parameter search approach is similar, in its treatment of the magnitude, to that of the multi-band fitting technique (Simard et al., 2002; Simard et al., 2011; Lackner & Gunn, 2012; Meert et al., 2014; Kim et al., 2016, among others) which fits an observation in one band and then uses this result as input—fixing all parameters but the magnitude—for an observation in a second band. We discuss this technique further in Section 9 but have not yet integrated it into our pipeline.

4.5 Distributed Processing

SDSS DR7 contains approximately 900,000 galaxies, 600,000 of which are likely to be spiral. It is therefore important to optimize our pipeline to fit models to these galaxies as efficiently as possible. `GALFIT`, although well-optimized and fast for low resolution galaxies with few components, which avoids the need to parallelize `GALFIT` itself, has no built-in bulk fitting capability. This suggests the need for a generalized distributed computing scheme to be the solution. We outline the method we use to perform the parallelization below to motivate our approach.

As described, the main `GALFIT` driver in our pipeline is the `go_go_galfit.py` script which can be run from the command line. This script, along with the `GalfitModule` containers, are what allow us to pipe galaxies to different processors or nodes via SLURM or other workload managers. In the current version of the pipeline, we utilize command line options to set the style of distributed computing, either by serial, on-node, or cluster computing via SLURM, and also use python's `asyncio` library for asynchronous computation with the serial and cluster parallelism options to accommodate the parameter search fitting procedure. The scripts/executables that handle the distributed processing can be found in the `ParallelDrivers` directory of the `GalfitModule`.

The main challenge of parallelization is the need to update the `GALFIT` components between stages. `GALFIT` outputs the model's parameterization in three places: the header of the output model FITS image block, STDOUT, and an output feedme file named, by default, `galfit.NN` where NN is an integer which keeps increasing so as not to overwrite the previous output file (from any galaxy) (Peng et al., 2010) and are dropped in the current working directory. The final method is the simplest way to manage subsequent fits as the output parameterization can be fed directly back into `GALFIT` however, beyond the runtime problem described above, a concurrency issue arises when using distributed computing as multiple nodes may attempt to write the same `galfit.NN` to file. Without these `galfit.NN` files being available to use, we instead ingest the resulting parameterization from STDOUT, update the components in the `GalfitModule` container, and output a new feedme file for the next stage of fitting.[1] If we have reached the final stage of fitting, we do not output a final feedme but pull the final model from the output FITS file and calculate our residual metrics at this stage. It would be possible to modify `GALFIT`'s source code as distributed via the Facebook group to change the naming convention of the `galfit.NN` files and thus avoid the concurrency issue, but our current approach, especially in the context of multi-stage fitting, saves on I/O and is ultimately more flexible.

[1]It is due to this limitation that we are unable to use other concurrency frameworks (such as `cofutures` or `multiprocessing`) to aid in the parameter search fitting procedure.

Chapter 5

Results Analysis

5.1 Residual Analysis

To evaluate the efficacy of our models, we use the combination of two methods to assess the residuals that are output by `GALFIT`. The first is a statistical Kolmogorov–Smirnov (KS) test as implemented from the `SciPy.stats` package (Virtanen et al., 2020) and the second uses a variation on Equation (2.1) to determine a numerical measure. Neither of these methods is particularly sensitive to the arm fitting of the model but in using both, we hope to set a numerical threshold on good/bad fits before we can proceed to use analytic methods to otherwise determine model success. We are hesitant to apply physically-motivated methods at this stage in the development of the pipeline as the multi-component structure of our models do not always lend themselves to direct physical analogues, an issue we discuss in Section 6.4.2.

To prepare for these two methods, we use our `OutputFits` to ingest the residual from the output FITS file as a `NumPy` array as well as the star mask generated by `SourceExtractor`. We then multiply the residual array by this star mask and store the result as an attribute in the object. The masked residual is the basis of both methods. We have attempted, in the past, to use a bulge mask to assist in the calculation of the residual in order to better isolate the arm fit according to the scientific goals laid out in Section 1. We generally find that this method produces significantly worse results when used during the fitting procedure (Section 4.5) so we opt to avoid its use for this study.

For the KS test, we perform our comparison of the masked residual against a Gaussian centered at the mean of the residual and with the same standard deviation to represent a pure-noise image. We take our null hypothesis to be that the two distributions are the same and reject the null hypothesis if the p value is less than 0.05. The KS test is incredibly sensitive to fitting errors near the bulge and in the brighter regions of the disk. For this reason, it is not suitable to be used alone for analysis according to the goals of our research.

The second method we use evaluates the norm (2-norm) of the residual to give a pure numerical measure of the goodness of the fit. This measure operates in a similar method to the built-in χ^2 measure output by `GALFIT`, (2.1), but it uses the minimum between the norm of the masked model and norm of the masked observation image as the normalizing value. We then take the norm of the normalized masked residual (NMR) and this gives us a value of the goodness of fit. This method is less sensitive than the KS test but tends to miss fine details and slight-if-noticeable errors in the arm fit. When combined, by multiplying the NMR with 1 - $\mathrm{p}_{\mathrm{KStest}}$, our fit quality measure, referred to as `NMR_x_1-p`, does a good job of filtering out exceedingly bad models and provides a robust method for sorting the results. It also provides a means of quantifying non-spiral fits, of which we categorize as ‘not successful’ in this study, despite how well the model may fit the observation.

These methods are integrated as a class method in the `OutputFits` class and store the results as each as attributes. The method also updates the header of model image block in the FITS file with the value of the NMR as well as the p value of the KS test which we use when we collate our final results.

We perform the residual analysis in the pipeline in `go_go_galfit.py`, as part of the parameter search process. After all steps of a fit have been completed, we calculate the residual of the different galaxy models so that we can determine a minimum value to select the best fit. To avoid extraneous file I/O during the already I/O intensive fitting process—i.e. by writing the results of this calculation to the header using the `generate_masked_residual` method of the `OutputFits` class—we store the `NMR_x_1-p` and find the minima of these values after the parameter search process is completed. We then update the header of the minima with the `NMR_x_1-p`.

In the main running script, `control_script.py`, the accumulation of residuals takes place after all `GALFIT` models have been generated. We again take advantage of the work done in `go_go_galfit.py` and merely extract the residual information from the headers of the FITS files. We also also collate the final model parameterizations at this time.

5.1.1 `GalfitModule`: Calculating and Extracting Residuals

We have created three utility codes which enable the user to calculate and extract residuals from `GALFIT` models and their FITS headers. These scripts are entitled:
`calculate_residual_info.py`, `calculate_residual_info_helper.py`, and
`extract_residual_info_from_headers.py`. Of the first two scripts,
`calculate_residual_info.py` performs the bulk of the preparation for calculating the residuals, and only calls the helper when distributed computing is employed via shell. The primary utility of this script is to calculate and collate the `NMR_x_1-p` values for every model (updating their headers along the way) as well as their parameterizations to be output as a `pandas` dataframe for later analysis. The secondary utility of this script is to test alternative residual methods as it can easily be updated to accommodate any changes to the masked residual calculation done by the `generate_masked_residual` method. Finally, the command line

arguments which can be fed to this script are structured in the same way as that of the control_script.py albeit with fewer options.

The extract_residual_info_from_headers.py script, is a utility developed to be used in the control_script.py. Assuming residuals have already been calculated and the headers updated, this script is used to collate both the NMR_x_1-p values and their parameterizations to be output as a pandas dataframe.

5.2 Arm Fit Analysis

The above methods can provide a general means for determining the success of a fit but neither is able to consistently quantify the success of the arm fit. We can, however, re-process the models through SpArcFiRe and use the arc metrics that SpArcFiRe provides to assess the models for arm consistency with the original observations.

To do this, we first generate a GALFIT model of the best-fit galaxy which contains only the arms of the model. For the arms-only model, we reduce the value of the rotated disk component's diskAxisRatio to 0.15, thus producing thinner arms (Section 2.2.2), to provide a 'pure' representation of the arm from which SpArcFiRe can determine an accurate pitch angle measurement. This generally has the effect of reducing non-detections at the risk of over-determining the arm length and occasionally losing contrast from the flux difference between bulge, arms, and sky. We then re-process the model through SpArcFiRe, taking advantage of SpArcFiRe's ability to fit spirals arm-by-arm and extract from each galaxy—before (observation) and after (model) performing the GALFIT decomposition—the top two arms in length which match the spin direction SpArcFiRe has determined for the galaxy. For each arm, we use the length, in pixels, and the pitch angle of each arm for our analysis.

To compare the lengths, we take the ratio of the shortest and longest arms of the model (again using the top two chirality-agreement arms) and use a fractional cutoff to determine success, i.e. the model's second longest arm must be some fraction of the length of the longest. In the case that `SpArcFiRe` does not detect more than one chirality matched arm, we take this ratio to be one and proceed with the analysis. Note: the original intention of this metric was to compare the shortest arm (of the top two in length) against the longest arm of the observation. We found, however, that a model-only comparison reduces the overall mis-classification rate and allows for a slightly more stringent cutoff value.

To compare the pitch angles, we take the minimum of the difference of the pitch angles for before and after as well as the difference between the reported `pa_alenWtd_avgdomChiralityOnly` values and use this difference to determine success. Table 6.1 displays the values chosen for the cutoffs for each metric. We discuss the considerations made in further detail in Section 7.1 but note here that the choice of cutoff value is arbitrary until compared with by-eye validated results. Finally, galaxies which do not have: an arm component in the final `GALFIT` fit, arc level information for either the observation or model from `SpArcFiRe`, or near-zero/near-90 deg pitch angles—although both these types of values are possible, it's more likely they represent failed galaxies—are excluded from consideration and are counted as failures. The results in Section 6 are produced using the cutoffs in Table 6.1 and considerations described above. Rerunning the models through `SpArcFiRe` is time consuming but doing so the most reliable approach we have found so far.

The code that reruns `SpArcFiRe`, `run_sparc_on_models.sh`, is currently found in the `Utilities` submodule but this may change with future versions of the `GalfitModule`. This script operates by preparing the working directory and running `SpArcFiRe` before placing the `csv` results in the user-specified runname (alternatively referred to as `basename`) folder in the output directory. The per-galaxy results are retained in a directory entitled `post_galfit-out`. As with the pipeline, this script takes advantage of the same distributed processing executables/scripts found in the `ParallelDrivers` directory of the `GalfitModule`.

5.3 GalfitModule: ResultsAnalysis

The GalfitModule contains several scripts in the ResultsAnalysis submodule which can be used to assess the results from running the pipeline. This analysis includes filtering the results by the metrics laid out in Sections 5.1 and 5.2, plotting functionality, the ability to combine results from multiple data sets, and a utility for extracting a representation of the results by residual metric (Section 5.1) quantiles. In order of the processing of results, we:

1. extract residual and parameterization information from the models, as collated into the `[basename]_output_results.pkl` file (Section 5.1),
2. extract SpArcFiRe results pre and post-GALFIT,
3. summarize and combine (1) and (2) according to the processing described in Sections 5.1 and 5.2,
4. evaluate success by user-defined metric cutoffs,
5. compare this performance to that of the by-eye success determination (optional),
6. combine the results from multiple data runs (if applicable), as used in this study to combine the NC2 and NC3 results,
7. plot several features of the results to evaluate their behavior in bulk, and
8. sort and output the results by `NMR_x_1-p` quantiles—using the PNGs if generated—for quick image reference (optional).

The combination of these can be found in the `results_analysis.py` script, which combines all of the above to complete the processing of the results. The results are output as a dictionary with format:

```
{
        basename                  : results namedtuple,
      "[basename prefix]_combined" : combined results namedtuple,
    }
```

where the results namedtuple has index number/fields:

0 / `full_df` – a dataframe containing all information for all results.

1 / `success_df` – a dataframe containing all information for the models which were deemed 'successful'.

2 / `not_success_df` – a dataframe containing all information for the models which were deemed 'not successful'.

3 / `by_eye_success_df` – a dataframe containing all information for the models which were deemed 'by-eye successful'.

4 / `by_eye_not_success_df` – a dataframe containing all information for the models which were deemed 'by-eye not successful'.

and where the combined results namedtuple has index number/ fields:

0 / `bool_df` – a dataframe containing (generally) boolean information regarding the combination of the results, i.e. success by metrics individually, success by metrics when combined per Section 5.3.1 below, by-eye success, etc. This dataframe should be used for counting results (as is used in `combine_multi_run_results.py`) given the choices used to determine success per the combination of result sets.

1 / `full_df` – a dataframe containing all information for all results across all data runs combined.

2 / `success_df` – a subset of `combined_full_df` which contains all information for the models which were deemed 'successful'.

3 / `by_eye_success_df` – a subset of `combined_full_df` which contains all information for the models which were deemed 'by-eye successful'.

purposefully re-used to provide flexibility when analyzing the results.

We recommend using the `jupyter notebook` `results_analysis.ipynb` to examine the results as it is the most convenient method to do so. In the notebook environment, and with the proper key word arguments set, the user can both see the summary of the results and view the plots generated accordingly.

5.3.1 Combining Results

In the schema we employ, the `GalfitModule` can produce n-component models, from 1–3 (Section 3). Given the morpohological diversity of spiral galaxies, it is impossible to choose a single n-component scheme which can provide the best fit for all observations in a survey. It is for this reason that we produce both 2 (NC2) and 3 (NC3) component fits and *combine* the results so as provide better coverage of the morphological diversity and to select the best fit among the two schemes.

The combination of multiple result sets occurs during the results analysis in the `combine_multi_run_results.py` function. This function takes as input the `full_df` dataframes produced as part of the results analysis described above and produces a series of comparisons between the n-component runs. When ran, these comparisons are summarized in STDOUT. We provide two primary methods of combination, and one additional method when compared with the by-eye results. These are referred to as:

$$\texttt{success\_m} \lor \texttt{success\_n} \lor \ldots \qquad (\land\ \texttt{by-eye}) \tag{5.1}$$

$$\texttt{minima} \rightarrow \texttt{success\_minima}\ (\land\ \texttt{by-eye}) \tag{5.2}$$

$$[(\texttt{success\_m} \lor \texttt{success\_n} \lor \ldots)\ \lor\ (\ \texttt{minima} \rightarrow \texttt{success\_minima})]\ \land\ \texttt{by-eye} \tag{5.3}$$

where `success_m` and `success_n` refer to the boolean variable indicating success by our metrics, `minima` $\rightarrow$ `success_minima` refers to selecting the best fit by the minimum of the fit quality score (if minimum score then that model is what we select in the combination to determine success), and `by-eye` is the boolean variable indicating by-eye success. The first two, (5.1) and (5.2), are also used in conjunction with by-eye results where available. Each of these combination methods are made available in `bool_df`, the combined boolean result dataframe, as: `by_sparcfire_score_success` (5.1), `by_sparcfire_and_best_score_success` (5.2), and `by_minima_or_sparcfire_success_by_eye` (5.3). For Eq. (5.1), if both models are 'successful', we choose the model with the lowest fit quality score to be the 'best fit' model, provided the result (by its runname) in a column entitled `best_fit`. Similarly, we include a column entitled `best_fit_by_eye` which also selects the smallest fit quality score when both are by-eye successful. We use these methods via Eq. (5.1) as the basis for our results in Chapter 6.

Note, it is possible to misunderstand the summary of results output to STDOUT by the `results_analysis.py` script when compared to performing the analysis on the results in `by_eye_success_df` via columns `success`, `by_eye_success`, and `runname`. The STDOUT output is derived from multiple operations on `bool_df` whereas `by_eye_success_df`, as a subset of `full_df`, is chosen by the runname which produced the best fit, i.e. is successful and has the lowest residual. It is sometimes the case that the 'successful+residual' model between runs is not the model selected in the by-eye evaluation so a boolean comparison between these two columns may cause discrepancies with the results in STDOUT.

5.3.2 Results Plotting

To plot the results, we use the `Plotting` submodule of the `GalfitModule` which contains templates and a common interface function, `create_plot`, for several types of plots—ECDF, scatter, scatter matrix, histogram. All plots are produced in Plotly (Inc., 2015). When results are generated, the user has the option to specify whether to produce plots via key word arguments to `results_analysis` which uses the `create_all_plots` function. Adjustments to the plots can be made by specifying a dictionary through the key word argument `plot_options` to `results_analysis`—We recommend viewing the plots in 'interactive' mode to determine appropriate bounds and adjustments before saving them. An example of this can be found in the `results_analysis.ipynb` notebook but dictionary keys generally follow the syntax: `[plot_feature_name]_[component]_[plot_type]`. Individual plots can also be generated through `create_plot` directly though we recommend consulting `create_all_plots` to identify preparation methods.

Chapter 6

Results on a Sample of Galaxies from SDSS DR7

6.1 Data Selection

We select galaxies from SDSS DR7 to coincide with and utilize the analyses performed by `SpArcFiRe` in Davis & Hayes (2014) and Silva et al. (2018) as well as Hart et al. (2016) who uses Galaxy Zoo 2 data (Willett et al., 2013, or GZ2) to quantify spiral galaxy morphology from DR7. We continue their analysis with r-band observations because the transmission intensity is maximized for SDSS (Stoughton et al., 2002). From this data, we have created three sample sets: 1) 14 well-resolved spiral galaxies for which `SpArcFiRe` can accurately map the arcs (as identified by-eye in the final fitting step, f, in Figure 2.1), 2) 1000 galaxies pulled at random from a set of GZ1 galaxies which were classified as having a spirality probability of $> 90\%$ and with apparent magnitudes in the r-band between 16 and 17, and 3) 28,912 galaxies from the original GZ1 sample with clearly observable structure as described

Sample Size	14	1000	28912
`NMR_x_1-p` Cutoff Value	0.007	0.007	0.009
Pitch Angle Cutoff Value	10	7	9
Arm Length Ratio Cutoff Value	0.5	0.62	0.45

Table 6.1: Table of the cutoff values used for each sample size for the fit quality, pitch angle difference, and arm length ratio metrics, selected to minimize the false positive and false negative classification rates.

in Davis & Hayes (2014) and Silva et al. (2018).[1] These approximately 29K galaxies are not guaranteed to be spirals but, by the analysis done by the human volunteers of Lintott et al. (2011b), they are more likely to be so. 897 of the set of 1000 galaxies can be found in this set of 29k.

We have chosen these three sets to optimize our pipeline to provide as much generality as possible while prioritizing the by-eye accuracy of spiral galaxy models—more discussion on this metric can be found in Section 6.4. Additionally, these galaxies have already been downloaded and used as part of `SpArcFiRe`'s most recent data release, as part of Silva et al. (2018), which saves additional computational time from needing to run `SpArcFiRe` again (see Section 4) and allows us to be consistent across our studies. The input images, or source observations, for all three sets, are postage stamps extracted from the fpC images provided by the SDSS Data Archive Server (DAS).

6.2 Summary of the Results

We present the results of our `SpArcFiRe` to `GALFIT` pipeline for the NC2 and NC3 methods as well as their combination (referred to as NC2 + NC3) for each of the data sets, 14, 1000, and 29k in Table 6.2. Table 6.1 displays the values chosen for the 'success' cutoffs for each metric. A 'success' is defined as any model which satisfies all of these cutoff values and which also matches the chirality between observation and model as determined by `SpArcFiRe`. We

[1]We have since pruned some of the galaxies from the initial sample of 29,250 but the premise remains the same.

Sample Size	14		1000		28912	
	NC2	NC3	NC2	NC3	NC2	NC3
Total Number of Models Generated	14	14	1000	1000	28912	28912
Processed by `SpArcFiRe`*	14	14	951	965	27469	27643
Fit Quality Metric	12	13	909	896	26048	27496
Pitch Angle Metric	12	10	606	617	18126	18206
Arm Length Ratio Metric	9	14	720	709	20925	20943
Chiral Agreement	7	13	626	584	14341	15095
Success by Metrics	5	9	364	319	9866	10312
Success by-eye*	5	11	406	418	358**	461**
False Positive Rate*	1/10	0/3	17.8%	15.4%	17.2%**	17.6%**
False Negative Rate*	1/6	2/13	29.6%	32.9%	28.7%**	32.1%**

	NC2 + NC3	NC2 + NC3	NC2 + NC3
Success By Metric	10	523	14905
Success by-eye	11	566	568**
False Positive Rate*	0/3	24.8%	23.5%**
False Negative Rate*	1/12	24.7%	23.8%**

Table 6.2: Table of results for our three sample sets, 14, 1000, 28912 (29k) for the NC2 and NC3 methods, and the metrics used for evaluating `GALFIT` success. We also present the results for NC2 + NC3, as a sub-table, counting a success in either set as contributing to the total. *All by-eye evaluations are with respect to the number in the 'Processed by `SpArcFiRe`' category. **As estimated by evaluating a sub-sample of 1000 random models from the 29k set by-eye.

discuss the considerations made for these cutoffs in further detail in Section 7.1 but note here that the choice of cutoff value is arbitrary until compared with by-eye validated results. Galaxies which do not have: an arm component in the final `GALFIT` fit, arc information for either the observation or model from `SpArcFiRe`, or near-zero/near-90° pitch angles are excluded from consideration and are counted as failures.

When combining the methods, i.e. NC2 + NC3, the success reported by our metrics is taken as *either* method producing a success. In the case that a success is reported for both methods, the model with the better fit quality score is taken as the best model fit. Several of the analysis plots presented display the results across NC2 + NC3 for every set—given that a model for each has been generated, the metric for both will be present. Appendix D presents the observation, model, residual images sorted by fit quality score at 20% quantile intervals for the 1000 and 29k data sets.

For the sample of 14 and 1000, we evaluate our results by-eye and compare these to the residual and arm fit analysis (Sections 5.1 and 5.2). For the 29k set, we evaluate a randomized subsample of 1000 galaxies by-eye and use this in support of our assessment of the pipeline's performance. We evaluate our false positive rate as False Positive / (False Positive + True Negative) and false negative rate as False Negative / (False Negative + True Positive) as illustrated in the 14 sample column. For each data set, we begin the process of minimizing false positive/negative rates from an empirical residual cutoff value of 0.007 based on a survey of all models from the sample of 1000 and 29k galaxies. From there, we adjust the cutoff values to minimize these rates in the combined sample. Between the sample of 1000 galaxies and the 1000 sub-sample of 29k galaxies, we see consistent false positive and false negative rates suggesting that 23 – 25% is a robust estimate for each of these rates.

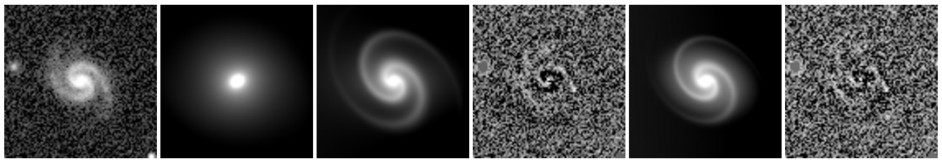
6.3 Results on the Selected Sample of 14 Galaxies

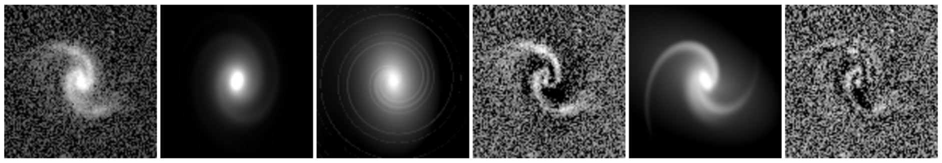
Figure 6.1 shows our results with each model tiled by: observation, model prior to GALFIT's convolution, NC2 model, NC2 residual, NC3 model, NC3 residual. The results are displayed in fit quality metric sorted order. Note that although the galaxies in the successful models may represent the visible structure of a galaxy well, the flux normalization process for converting between FITS and PNG often obscures issues in magnitude estimation so it is not necessarily the case that GALFIT has succeeded despite this advantage. Also note that we do not mask stars in the residual that may have otherwise been masked during the fitting process—this affects the scaling of the residual but it is often the case that these stars do not affect the fit as can be identified by the model image itself. We intend to apply this masking of the output in future versions of the GalfitModule.

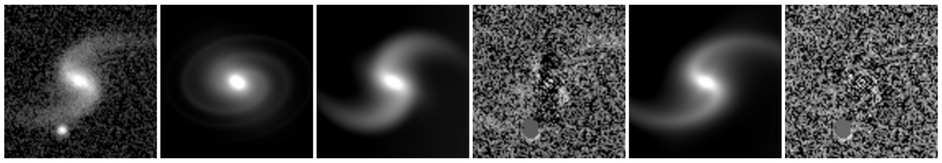
6.3.1 By-Eye Evaluation on the Set of 14 Galaxies

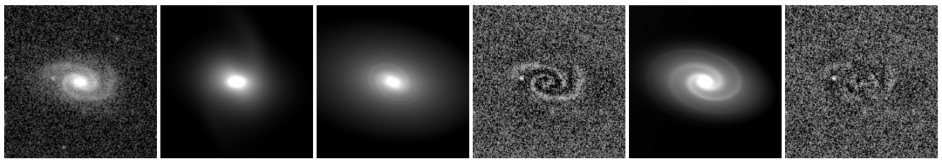
The true positive and false negative tiles in Figure 6.1 are the by-eye successful results for NC2 + NC3 for the set of 14. In general, we find that these models fit the pitch angle and widths of the arms in the observation well regardless of the bulge fit. There are also several cases where the model arms extend well beyond that of the original galaxy. This is expected behavior as it is exceedingly difficult to place a limit on the length of the arms—GALFIT has a truncation function built-in P, 2010 but the added complexity and difficulty of translation makes its use infeasible in an automated setting. Furthermore, SpArcFiRe provides an estimate of the beginning and ending of an arm which can be used to truncate the results of interest, as is done in Section 7.2. It is sometimes the case, as in the NC3 model of galaxy (2), that an arm from the observation (north of the bulge) can be clearly seen in the residual despite what appears to be a successful trace of that arm in the model. The success of this trace is further evidenced in the residual by the faint black structure which follows the (white) arm. This indicates two things: the first, that there is a brightness mismatch from

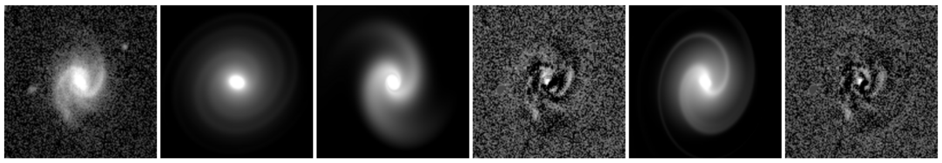
TRUE POSITIVE (TP)

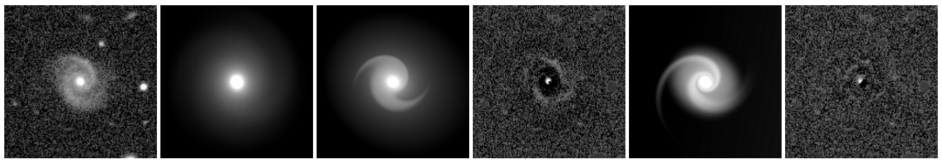
1) 

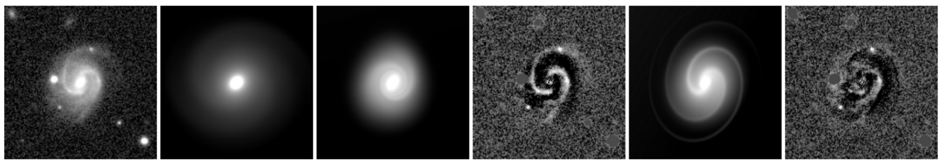
2) 

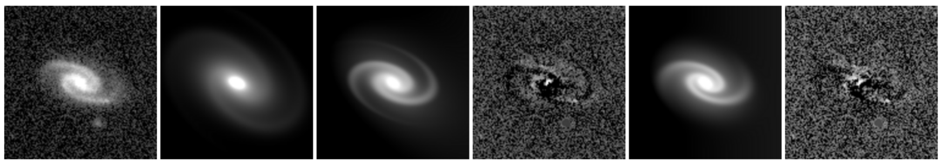
3) 

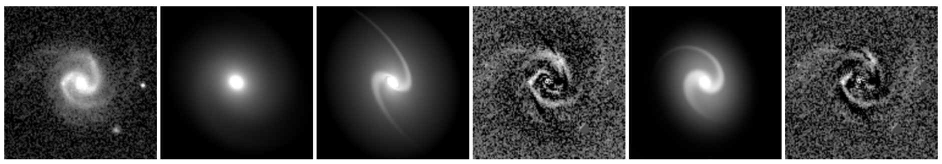
4) 

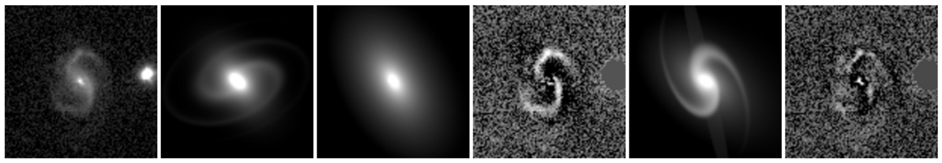
5) 

6) 

7) 

8) 

9) 

10) 

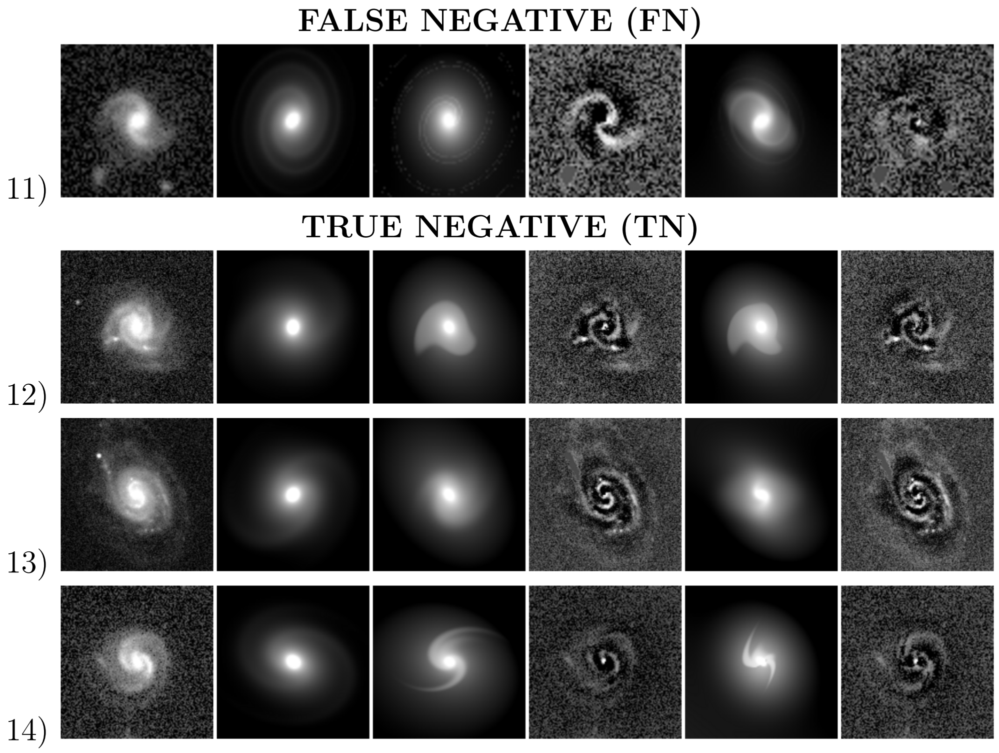


Figure 6.1: The results from the sample of 14 galaxies. Each result is tiled: observation, model as parameterized prior to `GALFIT`'s optimization, NC2 model and residual, and NC3 model and residual. The images in each portion—TP, FN, TN—are sorted by our measure of fit quality for the models which are deemed the best fit (i.e. smaller `NMR_x_1-p`) from NC2 + NC3.

the disk component—rotated to form the arm—in the region of that arm, and the second, that the model arm is thinner than that of the observation but that it still appears to fit both the winding and the pitch angle correctly. By the latter criteria and by the success of the other arm, we deem this model to be a good fit. To be clear, we purposefully overweight the pitch angle in our consideration to accommodate our scientific goal of examining spiral galaxy morphology and classification. We furthermore have a means of measuring the pitch angle as discussed in Section 7.2.

From working with this sample, it is clear that our approach is limited in its ability to model well-resolved galaxies, especially those with distinct sub-structure in the disk. This behavior was expected given the minimal component set used to describe the galaxies but will certainly need to be addressed in upcoming studies when higher resolution images are used. We anticipate incorporating something like the F-test used in Simard et al. (2011) and Lackner & Gunn (2012) which should integrate nicely with the multi-step fitting process and `GalfitModule`. Another issue which arose while working with this smaller subset, prior to and as partial justification for incorporating the parameter search method, is `GALFIT`'s inherent sensitivity to magnitude estimates. Despite much testing, we were unable to determine a 'better' mapping between the `petromag` from SDSS and our model components: adjustments to accommodate identified trends (Figures 6.5 and 6.10, Sections 6.4.2 and 6.5.2) often failed to make a difference. This inconsistency does offer some measure of robustness from `GALFIT`, and which appears again in the larger sample sets: an inaccurate parameter estimation for one galaxy (by one method) is an accurate parameter estimation for another galaxy. What this means is that minor changes in the individual parameterization, no matter `GALFIT`'s sensitivity to those parameters, is unlikely to have a major effect on the results for larger sets of data. This implies that we will have to turn to other methods (such as machine learning, Section 9) of parameter estimation improvement or of success validation, in order to see substantial increases in the performance of our pipeline.

6.4 Results on the Sample of 1000 Galaxies

Table 6.2 provides a quantitative description of the overall results as filtered by the analysis metrics used to determine success. Along with this table, We have produced several plots using NC2 + NC3 which highlight features of the models or which showcase the results of applying our metric cutoff values, Figures 6.3-6.6. We have included an additional figure that illustrates the distribution of arm-to-total flux ratios (S/T), defined as the ratio between the flux associated with the spiral arms and the total flux of the model, calculated by Equation (2.5). In these figures, we purposefully choose to label the components as 'Sersic 1,2,3' as opposed to 'bulge, disk arms' given the possibility of `GALFIT` using one component to aid in or otherwise take over the fitting of another component, i.e., a bulge expanding to become the disk according to Sérsic index and half light radius conventions. This also allows for clarity across NC2 and NC3 when compared directly in the same figure. We have also evaluated the 'Processed by `SpArcFiRe`' sample of both the NC2 and NC3 sets independently by-eye (Figure 6.2), choosing models which generally fit the shape and pitch angle of the arms to be 'successful' as motivated by the scientific goals stated in Section 1 and discussed further in Section 7.2. This selection criteria purposefully ignores arms which extend beyond that of the original observation (see Section 6.3.1), de-weights the contribution of the bulge fit, and furthermore excludes non-spirals from consideration, casting these as failures. We evaluated the 14 model sample set and the sub-sample of the 29k galaxy set under this same criteria. It is possible, in this scheme, that some valid models are excluded from consideration by `SpArcFiRe`'s failure to process them, however, careful observation suggests this number to be minimal and we don't expect our results to deviate much with their inclusion.

6.4.1 Residual Evaluation on the 1000 Galaxy Set

Figure 6.2 shows models at 20% quantiles along the residual analysis hierarchy for the 'by-eye successful' models of NC2 + NC3 for the 1000 galaxy set. When we examine the residuals

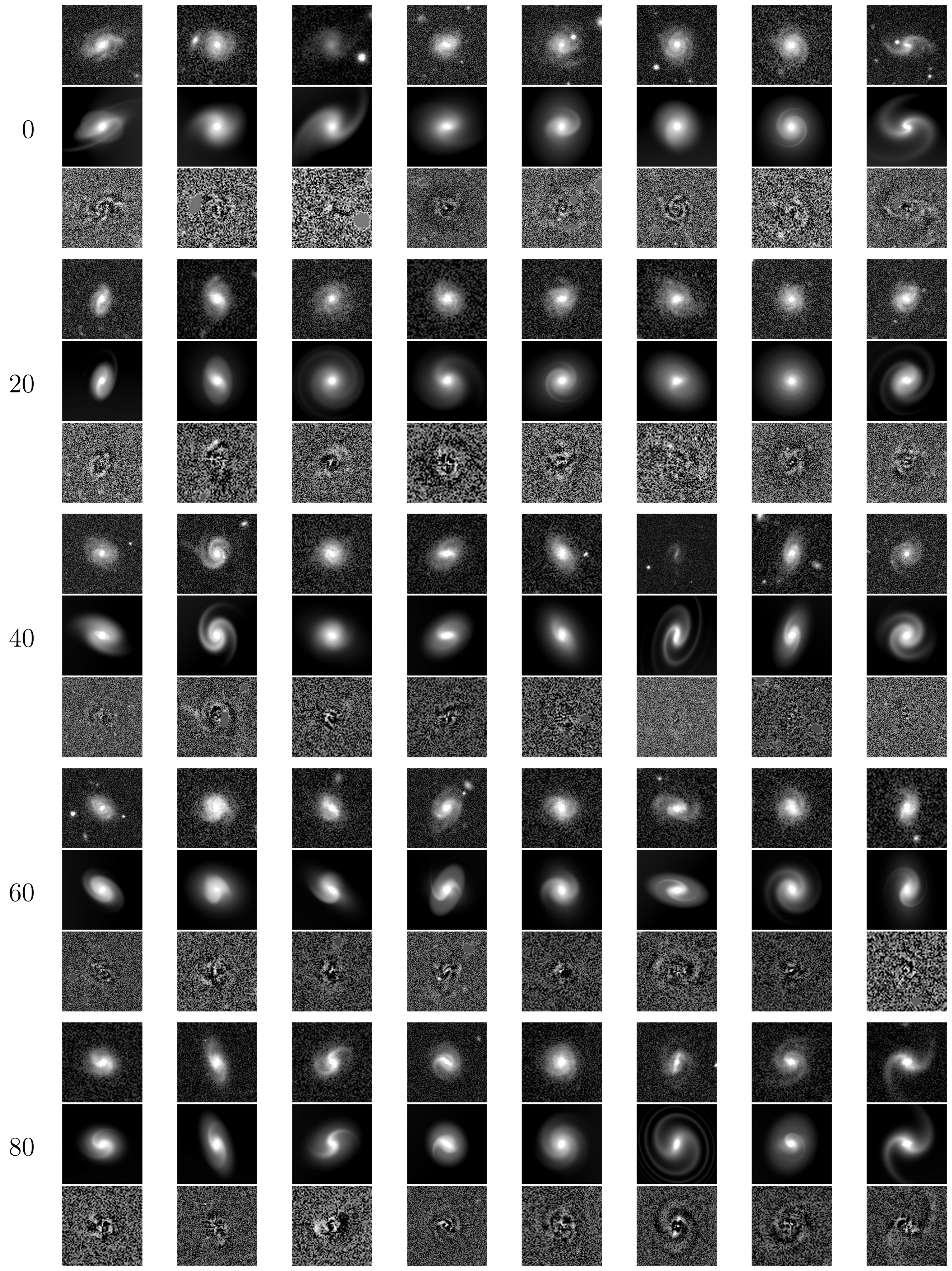


Figure 6.2: A representation of the 1000 galaxies from the by-eye successful NC2 + NC3 results in 20% quantiles of the fit quality score, tiled observation, model, residual as output by `GALFIT` and converted to png by `fitspng` (F. Hroch, 2019).

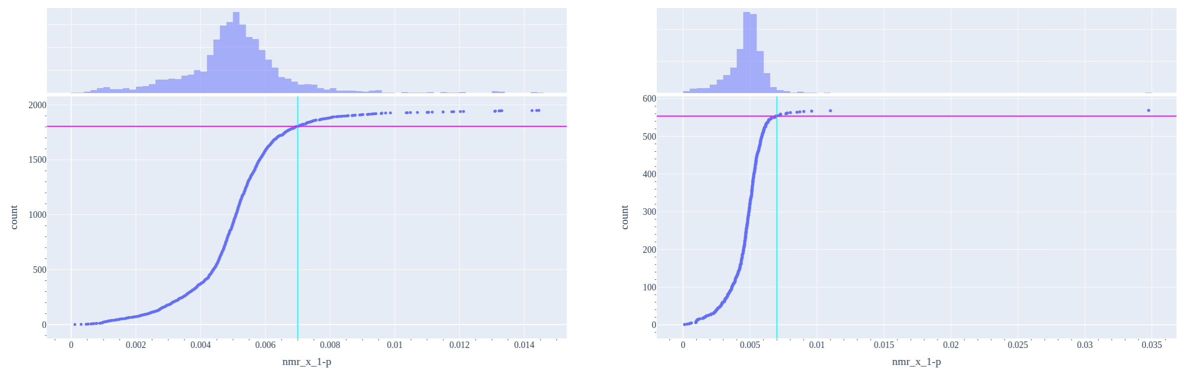


Figure 6.3: ECDFs of the fit quality measure for all models and all by-eye successful models in the combined methods result of 1000 galaxies.

of the by-eye successful models in Figure 6.2 we generally find little variation in the quality of the fits across the entire quantile range. Residuals are seldom completely indistinguishable from noise but the shape of the model galaxies and magnitude estimation are close enough to make it difficult to identify arm structures of the original galaxy. Structure, in the residual, is most often identifiable near the bulge or occasionally when the arms of the model don't match the length, thickness, or brightness of that of the observation. Given our scientific goals, none of these issues are of great concern.

The distribution of fit quality scores for all models of the 1000 galaxy set for NC2 + NC3 can be seen in Figure 6.3. We see from comparing these plots with those of the ECDF plots for the residual analysis of the by-eye-samples that the `NMR_x_1-p` threshold of 0.007 successfully captures a significant portion of the model fits, by-eye successful or otherwise. The residuals closest to this threshold, in the 80% quantile of Figure 6.2 (1000 by-eye successful) generally shows issues with the brightness distribution of the bulge and inner disk excepting a couple arm-less decompositions in the latter figure. We are therefore confident that although this measure is not a strong filter of the results, it nonetheless succeeds as a quality measure for our purposes.

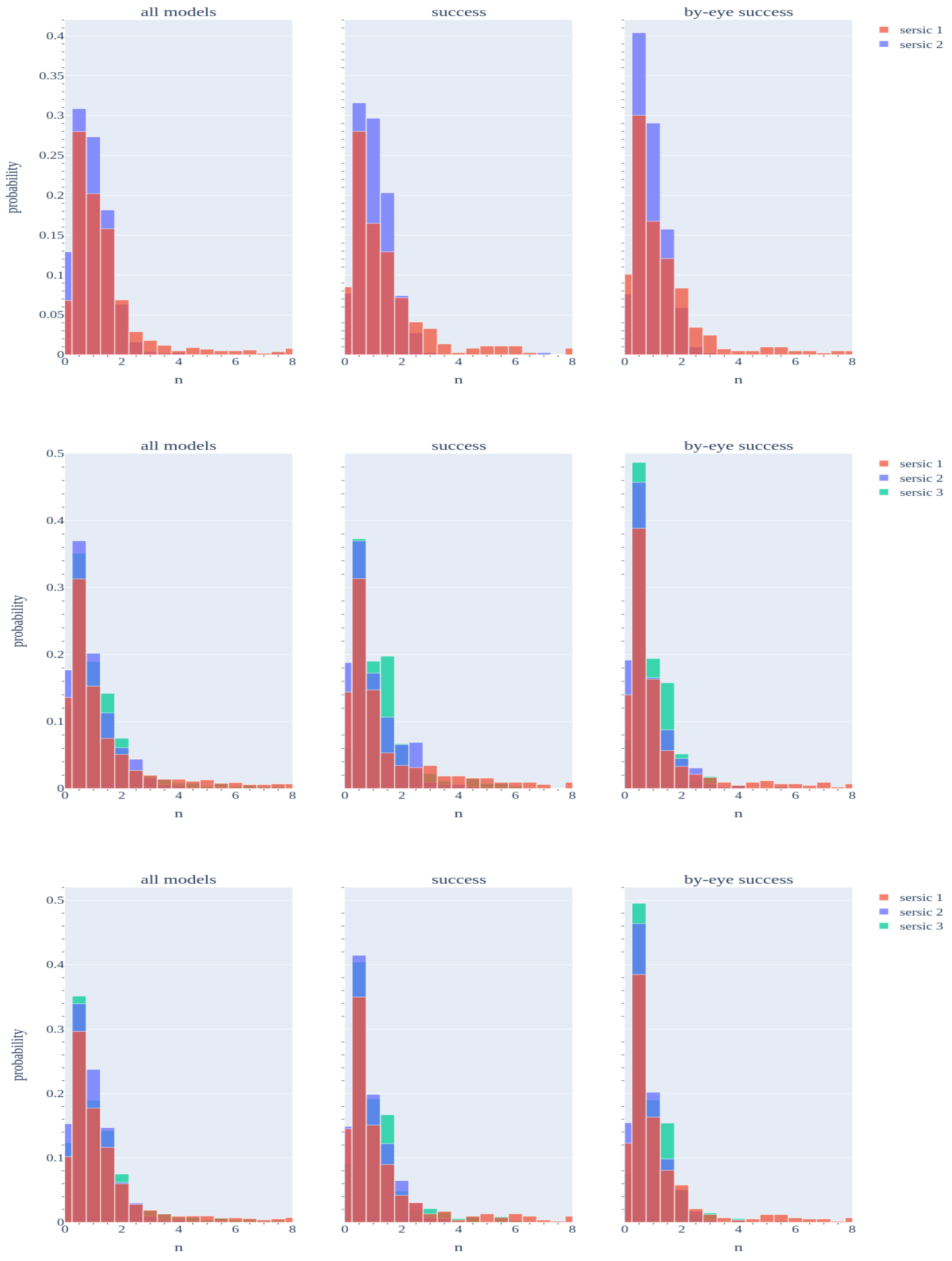


Figure 6.4: Histograms of Sérsic index values of the bulge, disk, and rotated disk for all, by metric successful, and by-eye successful models for NC2, NC3, and NC2 + NC3 of the 1000 galaxy set.

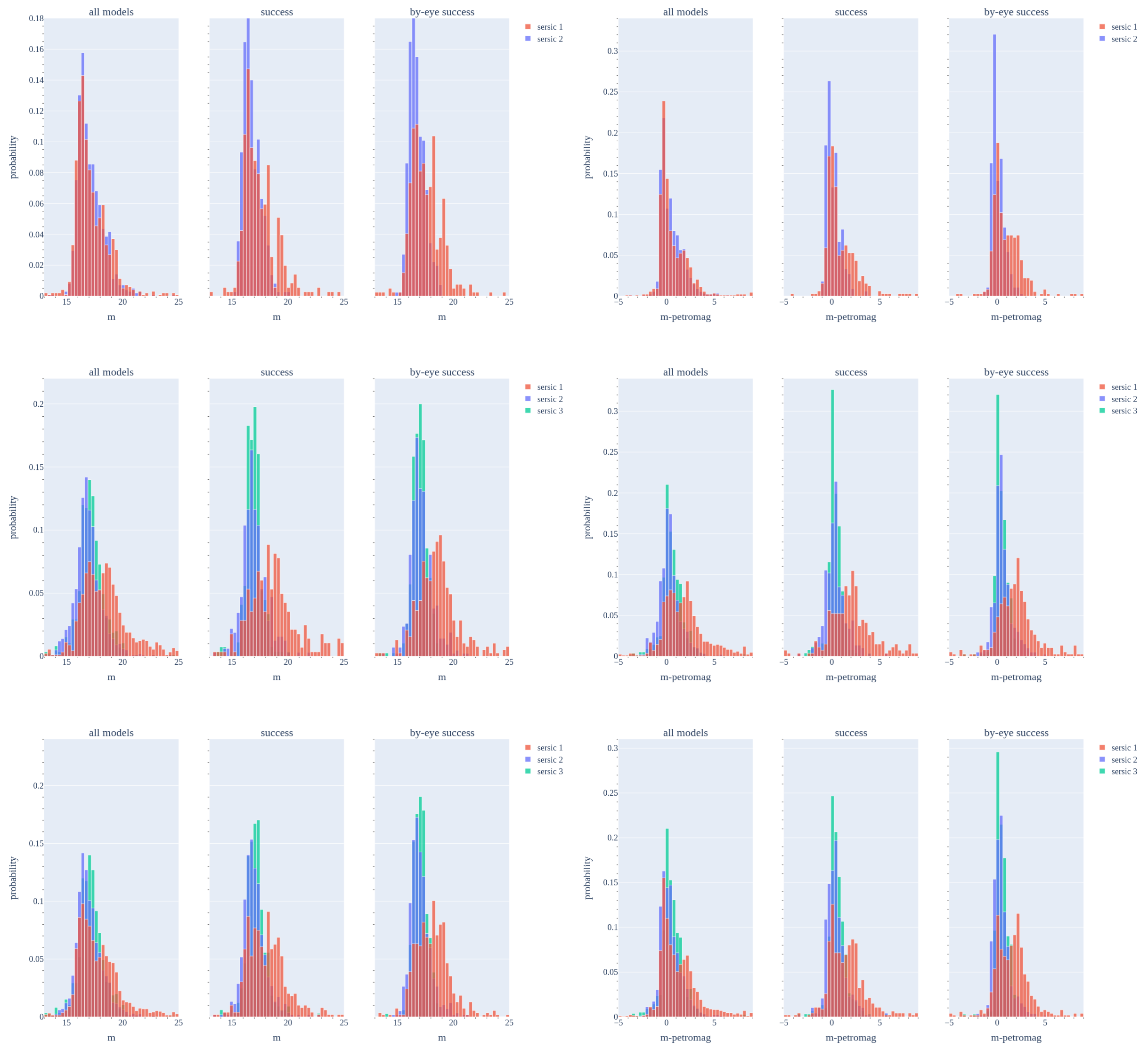


Figure 6.5: Left column: histograms of the magnitude values of the bulge, disk, and rotated disk for all, by metric successful, and by-eye successful models for NC2, NC3, and NC2 + NC3 of the 1000 galaxy set. Right column: histograms of the difference between magnitude and `petromag` for the categories detailed above.

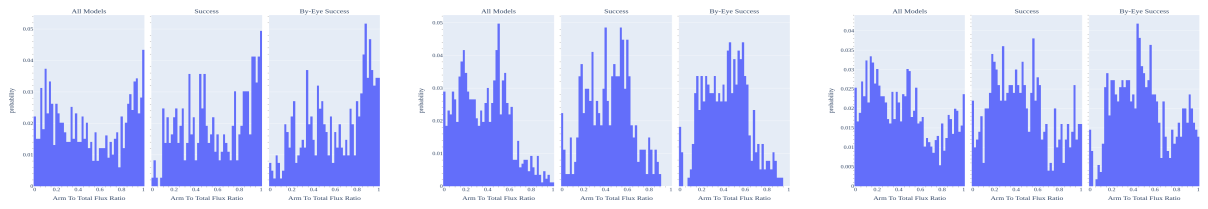


Figure 6.6: Histograms of the S/T for all, by metric successful, and by-eye successful models in NC2 + NC3 of the 1000 galaxy set.

6.4.2 Parameterization of the 1000 Galaxy Set

Figure 6.4 shows comparisons between the distribution of Sérsic indices for all Sérsic profiles in the NC2, NC3, and NC2 + NC3 results from the 1000 galaxy sample set across the different result categories: all models, by metric success, and by-eye success. These figures employ a 'probability' normalization scheme to equalize the representation across the result categories and to avoid over-representing the Sérsic 1 and Sérsic 2 profiles given that both NC2 and NC3 contain at least the two Sérsic components—this normalization happens with respect to the total of each component individually.

The distributions are generally consistent, with all results for all profiles peaking at or below the canonical disk value of $n = 1$. There is some scatter wherein the third Sérsic component demonstrates a second peak in the 'success' and 'by-eye success' plots near $n = 1.8$. This figure, however, shows very few results for any of the profiles at or near the canonical bulge value, $n = 4$. These findings are not consistent with the bulge-disk decompositions performed by Simard et al. (2011) and Meert et al. (2014). In Meert et al. (2014)'s discussion of the problem of overfitting, they indicate that this behavior likely implies either a low bulge Sérsic index or the inverting of the (intended) bulge and disk components wherein the bulge component is used to fit the disk and vice-versa. We find their assertion to be consistent with our NC3 results as evidenced by the magnitude distributions for the different profiles in the 'success' and 'by-eye success' plots in Figure 6.5. Gao & Ho (2017), in their study, uses `GALFIT` to model several NGCs with a series of configurations, the foundation of which being identical to our NC2 method. As with the B+D studies above, our Sérsic index results also generally disagree although their models too demonstrate generally lower bulge Sérsic indices than the canonical value. The phenomena of generally lower bulge indices is also found by Chugunov et al. (2023)—a significant majority of their models have bulge indices less than 4 with several near 1. These results do not fully explain the tendency for a substantial fraction of the indices to be lower than 1 but the discussions in Gao & Ho (2017) and Chugunov et al.

(2023) indicate that this issue may be a result of the increase in central surface brightness when arms (which reach the center of the galaxy) are applied. This is likely an excellent starting point for examining the issue as a whole.

As discussed in Section 3.3, large Sérsic indices are sensitive to the sky estimation. By this premise, it is possible that our treatment of the background sky is too imprecise and, by this imprecision, that the multi-step, multi-component fitting process may therefore exacerbate the issue by forcing the bulge component to accommodate other features in the image. We attempted to correct for this systematic error by fixing the sky between steps after performing an initial fit on the bulge but we found that `GALFIT` generally performed worse after doing so. However, we believe that this issue, and our previous attempt at its correction, is worth re-examining in light of our results. It may also be possible that our inability to use a PSF image in the convolution of the fit is affecting the Sérsic index here but in previous attempts at using the PSF, we found little to no difference in the bulk statistics of the results. It is also worth noting that the existence of spiral arms more generally affects the photometry described by the Sérsic profile (Sonnenfeld, Alessandro, 2022; Chugunov et al., 2023) but neither of these studies show an effect as drastic as can be seen in our results.

Similar to Figure 6.4, the following figures compare the distribution of a selected quantity of the Sérsic profile in our sample sets: 1) Figures 6.5 compares the distribution of magnitudes as well as the distribution of magnitude - `petromag` (as provided by SDSS DR7); 2) Figure 6.6 shows the distribution of the S/T. We will discuss this latter figure with respect to the 29k galaxy sample set in Section 6.5.2.

Of the magnitude distributions in 1: in contrast to the Sérsic index figures, we see the behavior of the magnitudes diverge across NC2 and NC3 in the Sérsic 1 component for both sample sets. Although both methods show some spread in this component (although with more samples, the spreading appears to smooth out), the centers are shifted: NC2 behaves closer to our expectation—brighter bulge, dimmer disk— than that of NC3. In the NC3 results, the shift is much greater than the offset between the two components in NC2. We suspect that this shift occurs due to the concentration of light at the center of the galaxy from the additional disk component. This behavior, again, indicative of overfitting, presents a dilemma: according to the results in the 29k column of Table 6.2, the NC3 method produces a significant fraction more by-eye successful galaxies than that of the NC2 method. It may be prudent then to compare the Sérsic 2 component in NC3 with that of the Sérsic 1 component in NC2 on the assumption that `GALFIT` is effectively using this second component in place of the first to model the bulge and using the first component to fill in where necessary. This is intuitive but aids in explaining the preference in the residual analysis for NC3 fits.

In the 1000 sample set, the Sérsic 1 of NC2 and Sérsic 2 of NC3 generally align. We see a slight difference in the spread of the rotated disk components but in looking at the comparison with `petromag` in Figure 6.5, it is not clear that the differences in spread are meaningful.

Of the magnitude - `petromag` distributions in 1: the behavior here is similar to that of the magnitudes in that we see a shift in the Sérsic 1 component difference for NC3. In comparison to the results of the catalogue created by Simard et al. (2011, Figure 8) and that of Meert et al. (2014, Figure 12), we expect a slight offset (± 0.3) from the `petromag` for both components. The Sérsic 1 component in NC2 and Sérsic 2 component in NC3 better satisfy this constraint than the other components although in both methods, the distribution is tightened for galaxies labeled as 'successful'. Note that the NC3 results are distinctly not centered at 0 for all components but that these are slightly shifted to the right. It may be possible, in the future, to use this comparison as a metric for determining overfitting and to subsequently re-parameterize a model to avoid this issue.

Additional parameterization-related plots can be found in Appendix E where figure E.1 shows pairwise plots for several Sérsic parameters in the NC2 + NC3 1000 galaxy sample as colored by their success classification with 'by-eye success' taking color priority. This prioritization takes the form of boolean masking: models labeled ‘by-eye success’ retain this label regardless of their ’by metric success’ status. In the 1000 galaxy set, this prioritization is especially important given that all 1000 were labeled by-eye. This implies that all ’by metric success’ models in these plots are false positives, and all ‘not successful’ models are true negatives. In this scheme, we cannot identify false negatives from the scatter plots but this is less important in this context. The parameters, magnitude or `m`, effective radius or `r_e`, and Sérsic index or `n`, are labeled `parameter_component number`. Note that we artificially limited the axes ranges—consistent across parameter types—to identify trends in regions of the plots that are more densely populated.

6.5 Results on the Sample of 28912 Galaxies

As with the 1000 galaxy set, we presented tabulated results for the 29k sample in Table 6.2 and plots of various features of the results in Figures 6.8-6.11. We can estimate our final success count from the 56.8% of 1000 labeled by-eye. Applying this percentage to the 27643 models processed by `SpArcFiRe`, we estimate our final success count to be approximately 15,700 models, equivalent to 54.3% of the total 29k.

Figure 6.7 shows a sample of the by-metrics ‘successful’ models at 20% quantile intervals of the fit quality score. Note that although all galaxies pictured have satisfied our quantitative metrics, several would not pass our by-eye criteria.

6.5.1 Residual Evaluation on the 29k Galaxy Set

Figure6.7 shows models at 20% quantiles along the residual analysis hierarchy for the ‘successful’ models of NC2 + NC3 for the 29k set. When we examine the residuals of the by-metric successful models in 6.7 we find these results to be consistent with that of the 1000 galaxy set.

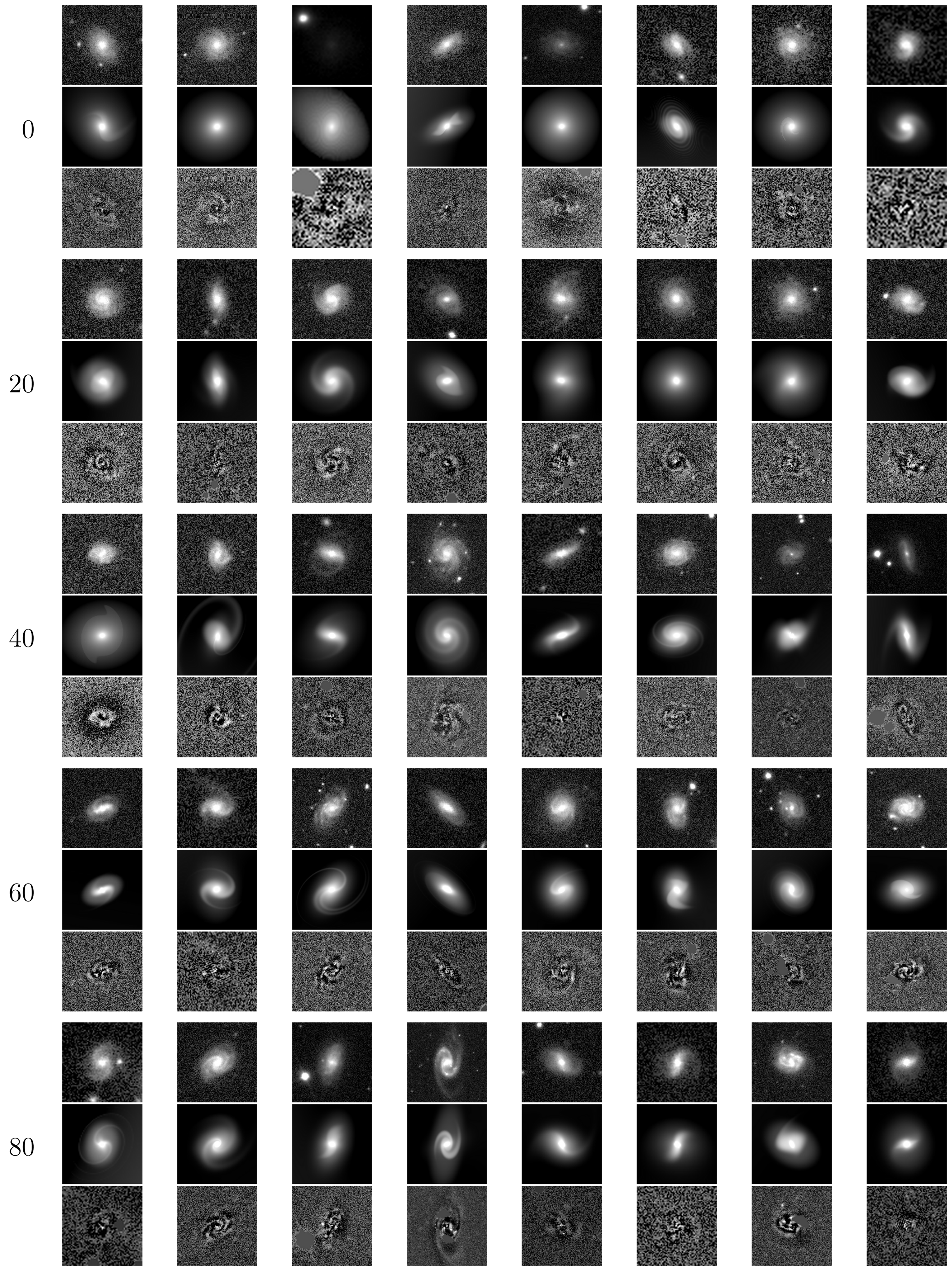


Figure 6.7: A representation of the 29k galaxies from the by-metric 'successful' NC2 + NC3 results in 20% quantiles of the fit quality score, tiled observation, model, residual as output by `GALFIT` and converted to png by `fitspng` (F. Hroch, 2019).

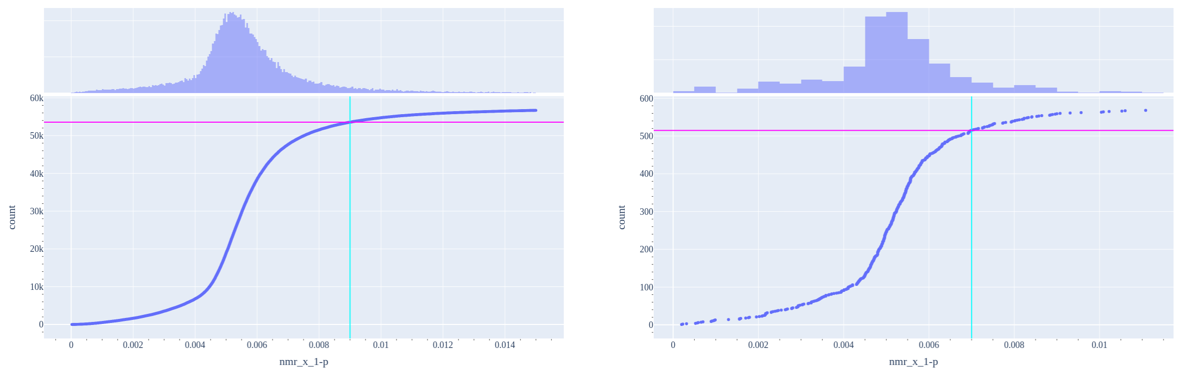


Figure 6.8: ECDFs of the fit quality measure for all models in the combined methods result of 29k galaxies.

The distribution of fit quality scores for all models of the 29k set for NC2 + NC3 can be seen in Figure 6.8. In performing the same comparison (between ECDF plot and our quantiled sample) as with the 1000 galaxy set, we again see consistency in the performance of our method and of our fit quality metric.

6.5.2 Parameterization of the 29k Galaxy Set

Figure 6.9 shows comparisons between the distribution of Sérsic indices for all Sérsic profiles in the NC2, NC3, and NC2 + NC3 results from the 29k galaxy sample set across the different result categories: all models, by metric success, and by-eye success. As with the 1000 set, these figures employ a 'probability' normalization scheme to equalize the representation across the result categories and to avoid over-representing the Sérsic 1 and Sérsic 2 profiles given that both NC2 and NC3 contain at least the two Sérsic components—this normalization happens with respect to the total of each component individually.

The results for this data set generally agree with the 1000 sample set—all results for all profiles peak at or below the canonical disk value of $n = 1$ and bulge value of $n = 4$. With respect to the 1000 sample set, we do not see the same second peak in the 'success' and 'by-eye success' plots near $n = 1.8$ for the third Sérsic component. These findings are again

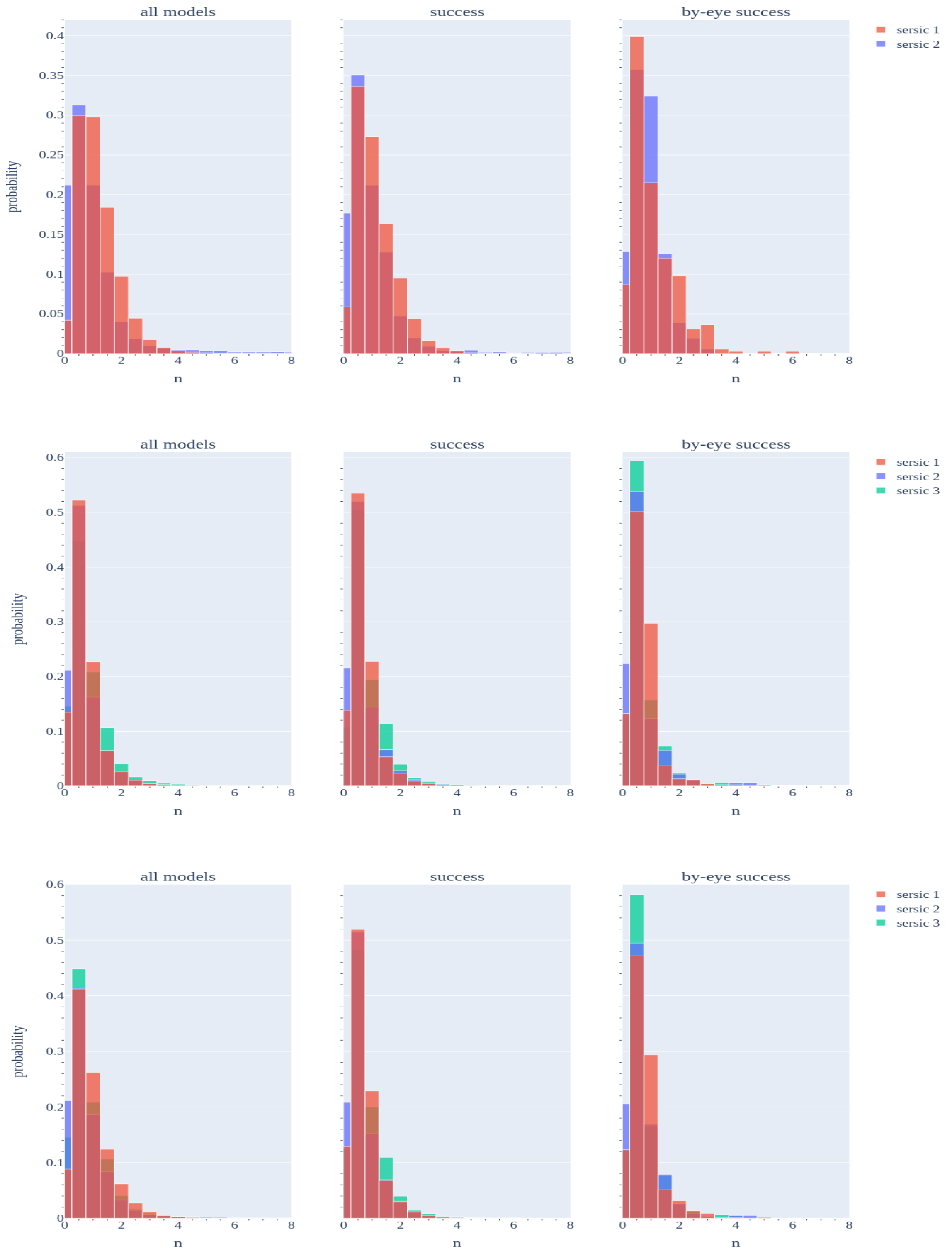


Figure 6.9: Histograms of Sérsic index values of the bulge, disk, and rotated disk for all, by metric successful, and by-eye successful models for NC2, NC3, and NC2 + NC3 of the 29k galaxy set.

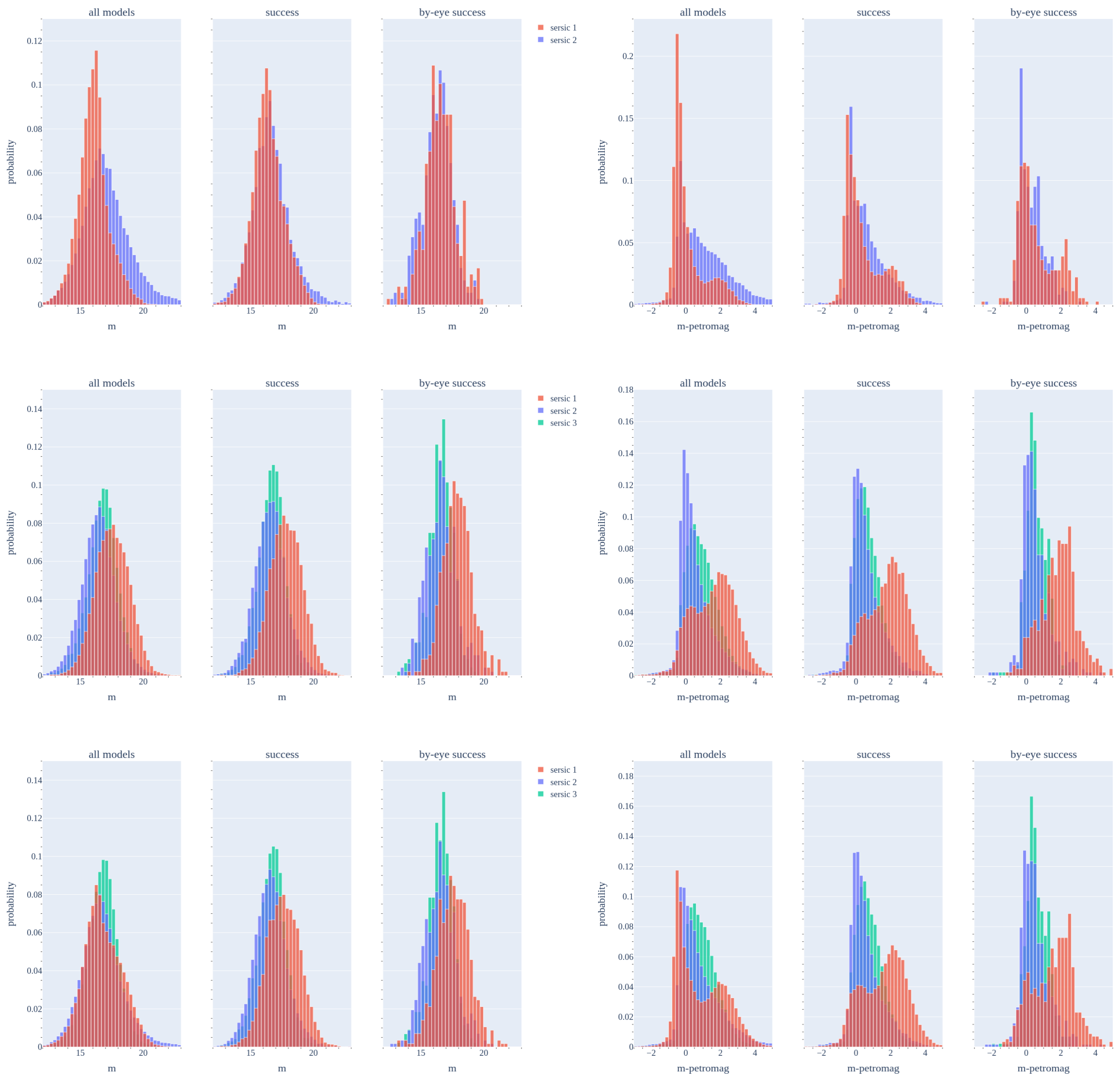


Figure 6.10: Left column: histograms of the magnitude values of the bulge, disk, and rotated disk for all, by metric successful, and by-eye successful models for NC2, NC3, and NC2 + NC3 of the 29k galaxy set. Right column: histograms of the difference between magnitude and `petromag` for the categories detailed above.

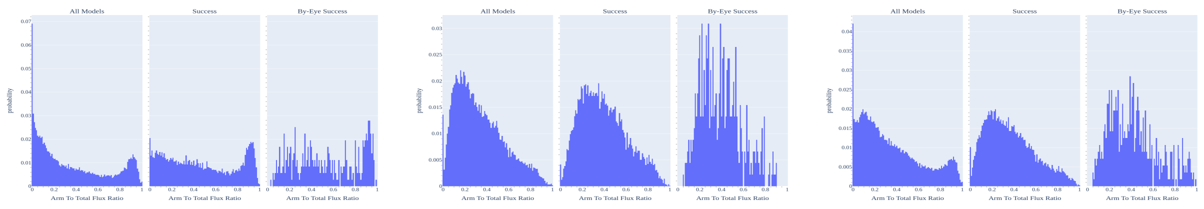


Figure 6.11: Histograms of the S/T for all, by metric successful, and by-eye successful models in NC2 + NC3 of the 29k galaxy set.

not consistent with the bulge-disk decompositions performed by Simard et al. (2011) and Meert et al. (2014), indicative of everfitting.

Similar to Figure 6.9, the following figures compare the distribution of a selected quantity of the Sérsic profile in our sample sets: 1) Figure 6.10 compares the distribution of magnitudes as well as the distribution of magnitude - `petromag` (as provided by SDSS DR7); 2) Figure 6.11 shows the distribution of the S/T which we will compare to that of the 1000 set.

Of the magnitude distributions in 1: we again see the behavior of the magnitudes diverge across NC2 and NC3 in the Sérsic 1 component. In comparison to the 1000 set, the Sérsic 1 of NC2 and Sérsic 2 of NC3 again generally align, however NC3 in the 29k sample set shows a wider spread. We see a slight difference in the spread of the rotated disk components for each as well but in looking at the comparison with `petromag` in Figure 6.10, it is not clear that the differences in spread are meaningful. The results for the magnitude - `petromag` distributions in 1 also agree with that of the 1000 set.

Of 2: the trends in the 1000 (Figure 6.6) and 29k galaxy sample sets differ substantially. This may be due, in part, to the selection criteria used for the 1000 galaxy set, in that the higher the likelihood of spirality, the brighter the arms will appear although it should be noted that the y-axis scale is also relatively closely spaced. If we examine the 29k sample set then, we see that the NC2 and NC3 results also show differing behavior. NC2 peaks at either end of the range while NC3 peaks around 0.2. The latter is encouraging as it generally agrees with the S/T results of Savchenko et al. (2020) and Chugunov et al. (2023) given that `GALFIT`'s rotation function favors two-arm spirals. Note, the distribution in the 'by-eye success' figures—which are limited to 1000 galaxies—shows a large scatter that seems to favor $S/T \leq 0.5$ which supports the assertion that the scatter found in the 1000 sample set may be due to the selection criteria. We see again here in comparing the 'success' plots across the methods that NC3 is favored in terms of minimizing the residual: there is *no* evidence of the peak in the NC2 plot near 1 in the NC2 + NC3 plot.

Additional parameterization-related plots can be found in Appendix E where Figure E.2 shows pairwise plots for several Sérsic parameters in the NC2 + NC3 29k galaxy samples as colored by their success classification with ‘by-eye success’ taking color priority. As with the 1000 set, this prioritization takes the form of boolean masking: models labeled ‘by-eye success’ retain this label regardless of their ’by metric success’ status.

Chapter 7

Additional Discussion

7.1 Validation by `SpArcFiRe`

As discussed in Sections 5.1 and 5.2, we use the combination of our fit quality measure (`NMR_x_1-p`) with the pitch angle difference and arm length ratio metrics to determine whether a model is successful in representing an observation of a galaxy. In the creation of this validation scheme, we initially developed the above metrics to corresponded with our intuition about the performance of `SpArcFiRe` and with our expectations for the 'success' of a model, predominantly by setting hard cutoff values for both metrics. We have found, however, that the cutoff values we had set were inconsistent in their ability to correctly classify the success of models, when compared with the by-eye classifications, across the different data sets and results presented in this study. We therefore chose to allow these values to vary across the different data sets. Individually (per component method), these values are generally too strict but when the results are combined, the strictness balances out.

Figures 7.1 and 7.2 show scatter plots of the arms-only pitch angle values found by `SpArcFiRe` for all model fits in NC2 + NC3 for the 1000 and 29k galaxy samples according

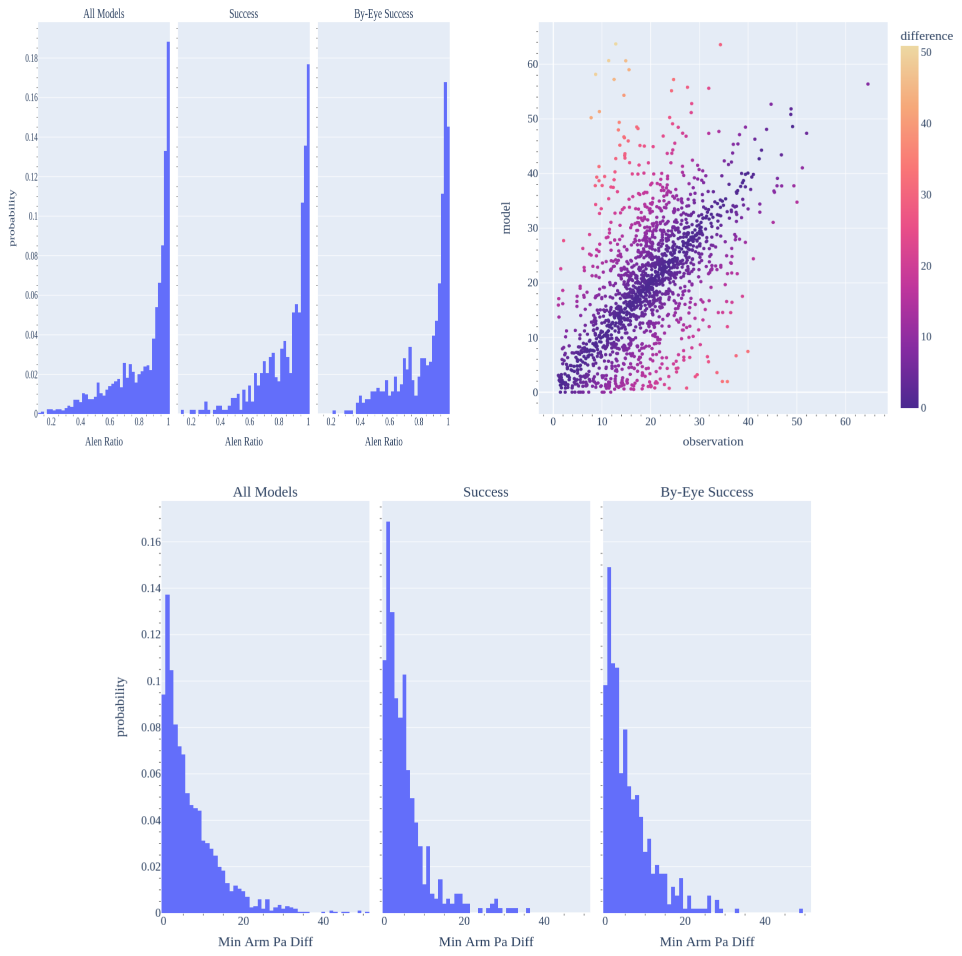


Figure 7.1: From the NC2 + NC3 results on the set of 1000 galaxies—Top Left: histogram of the arc length ratio between the longest and shortest arm in the model. Single arms are counted as arc length ratio = 1. Top Right: scatter plot of the pitch angle difference between the top two longest arms of the observation and model for all 2000 models from NC2 + NC3. Bottom: histograms of the pitch angle differences pictured in the scatter plot. All quantities in these plots are taken as measured by `SpArcFiRe` according to the analysis performed in Section 5.2.

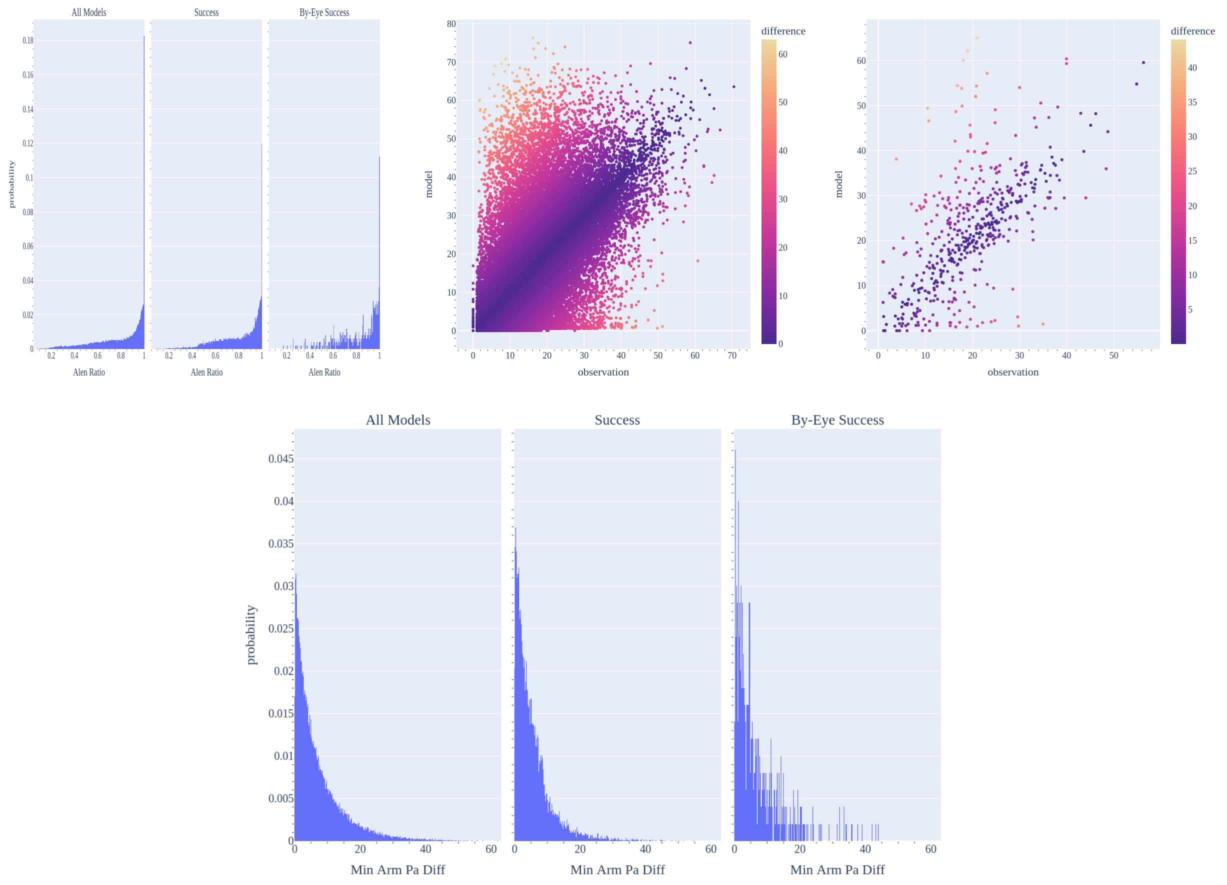


Figure 7.2: From the NC2 + NC3 results on the set of 29k galaxies—Top Left: histogram of the arc length ratio between the longest and shortest arm in the model. Single arms are counted as arc length ratio = 1. Top Middle and Right: scatter plots of the pitch angle difference between the top two longest arms of the observation and model. The middle shows the difference for all models, while the right is for those identified as successful by-eye. Bottom: Bottom: histograms of the pitch angle differences pictured in the scatter plot. All quantities in these plots are taken as measured by `SpArcFiRe` according to the analysis performed in Section 5.2.

to the analysis presented in Section 7.2, colored with respect to their difference. According to the results in Table 6.2, 1223/2000 models satisfy the cutoff for the 1000 galaxy set while 36,332/57,824 satisfy the cutoff for the 29k galaxy set. We see a relatively tight distribution for both plots around the y = x line—as validated by the pitch angle difference histograms—but note a bias in both plots lower than this line at lower model pitch angles and higher than this line at higher model pitch angles. Figure 7.2 shows an additional scatter plot which combines all by-eye successful fits for both methods of the 29k galaxy sets. In this plot, this bias is still present but far less noticeable. We have not as-of-yet identified the source of this bias.

In general, the mis-classification rates are more sensitive to variation in the pitch angle difference cutoff value ($\pm 1\,\mathrm{deg}$) than that of the 'equivalent' fit quality (± 0.001) or arm length ratio (± 0.1) cutoff values as evidenced by the arc length ratio and scatter plots in Figures 7.1 and 7.2 with respect to the ECDF plot of the residual fit quality (Figures 6.3 and 6.8).

There are two assumptions that were made in the development of `SpArcFiRe` that renders our arm-fit analysis imperfect: 1) `SpArcFiRe` is intended to run on real observations, and 2) `SpArcFiRe` utilizes a logarithmic spiral function to trace out the arms. Of the first assumption, the pre-processing that `SpArcFiRe` performs (Steps a and b in Figure 2.1, described in further detail in Davis & Hayes, 2012; Davis & Hayes, 2014) is used to isolate and enhance observations with features that can be systematically transformed for its use. When used on `GALFIT` models without further adjustment, we have found `SpArcFiRe` to perform inconsistently, most often in its contrast enhancement. Furthermore, the image standardization process may be inconsistent between observation and model, which may alter the results when comparing the two. We attempt to avoid this inconsistency by re-using the image standardization produced by `SpArcFiRe`'s analysis on the observation. We found, however, that the false positive/negative rates are slightly higher, likely due to the tightness of `SpArcFiRe`'s

Galaxy (8), NC3, from Figure 6.1

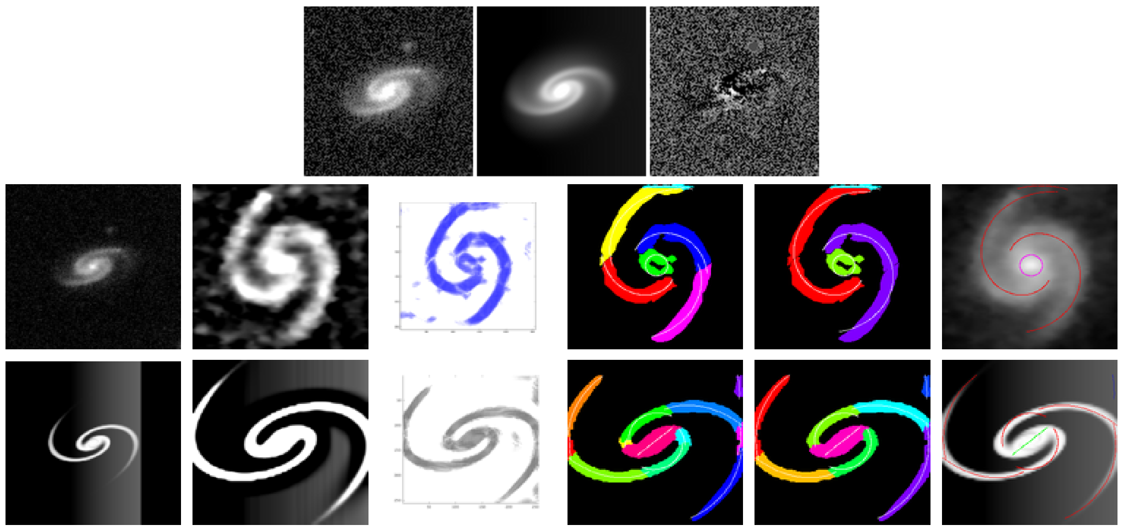


Figure 7.3: A false negative model as determined by-eye with respect to our success metric. Top row: observation, GALFIT model, and residual flipped from Figure 6.1 to match SpArcFiRe's output. Middle row: SpArcFiRe's analysis of the original observation. Bottom row: SpArcFiRe's analysis of the GALFIT model. Note that the model used as input has the arm width artificially reduced to aid in SpArcFiRe's analysis. According to SpArcFiRe, this galaxy failed by the pitch angle value metric. Note that the orientation flip between the original observation in the top row and SpArcFiRe's analysis is expected behavior (Section 3.1) and is accounted for in our processing—it is not indicative of a failure on either GALFIT or SpArcFiRe's part.

Galaxy (9), NC2, from Figure 6.1

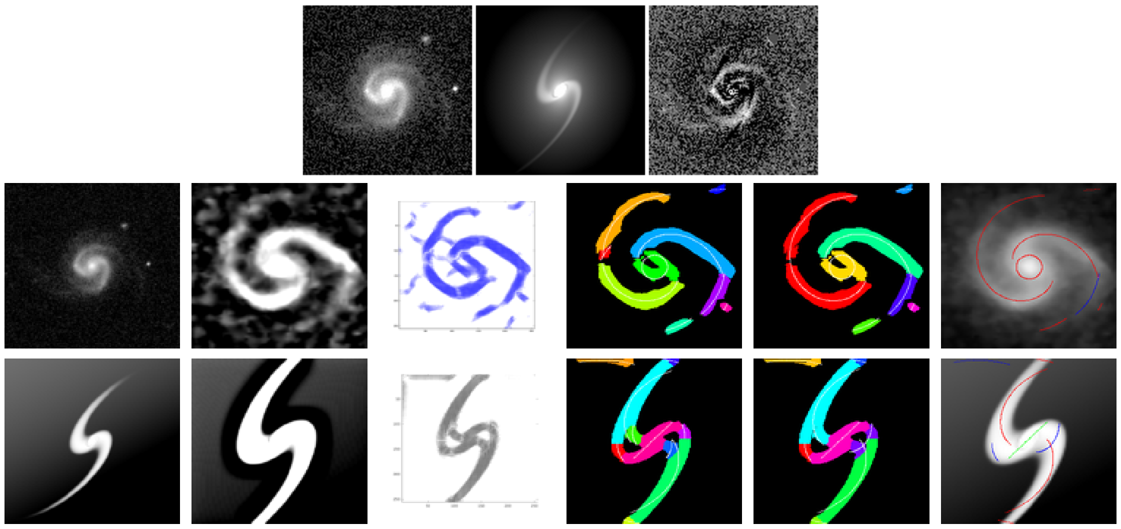


Figure 7.4: A false positive model as determined by-eye with respect to our success metric. Top rows: observation, `GALFIT` model, and residual flipped from Figure 6.1 to match `SpArcFiRe`'s output. Middle rows: `SpArcFiRe`'s analysis of the original observation. Bottom rows: `SpArcFiRe`'s analysis of the `GALFIT` model. Note that the model used as input has the arm width artificially reduced to aid in `SpArcFiRe`'s analysis.

cropping operation and the lack of contrast at the center of `GALFIT`'s coordinate rotation. Of the second assumption, `GALFIT` is capable of fitting a wide range of spiral structures with a description that is not limited to the logarithmic spiral formulation. This can cause a significant deviation in the pitch angle estimates when arms in either the observation or the model do not follow the logarithmic spiral prescription. These two issues appear to have caused `SpArcFiRe` to misclassify the models in Figures 7.3 (false negative) and 7.4 (false positive). In the case of the false negative, the difference in pitch angles between the observation and model was larger than that of our cutoff, as caused by the de-inclination of the observation during the image standardization process, i.e. a consequence of the first assumption. For the false positive, we find that this analysis falls prey to the second assumption as evidenced by the tightness of `SpArcFiRe`'s final arm fit.

7.2 Pitch Angle Analysis

The pitch angle of a spiral galaxy is a measure which describes the winding tightness of a galaxy's spiral arms and may be correlated with a number of galactic properties—the review by Sellwood & Masters (2022) provides an excellent overview of the work that has been done in this area over recent years. As discussed in Section 1, there are several automated methods by which it can be measured, the most common of which being the discrete Fourier transform technique. `SpArcFiRe` also provides galaxy-level as well as individual arm measures of the pitch angle (Davis & Hayes, 2014) as does the Galaxy Zoo Builder project (Lingard et al., 2020, 2021). These methods assume a constant pitch angle when conducting their measurements—by virtue of the governing approximation of logarithmic spiral arms, Equation (7.2)—while other methods, (Ringermacher & Mead, 2009; Davis et al., 2012; Savchenko & Reshetnikov, 2013) allow for varying pitch angles along the length of the arm (Davis et al. (2012) sums logarithmic spirals with differing pitch angles to achieve this).

The pitch angle is defined in Binney & Tremaine (2008) as the angle between the line tangent to a spiral arm, as formed with respect to the center of the galaxy, and the azimuthal direction—the lower the angle, the tighter the winding, limited between 0 deg and 90 deg. It can be described by the equation

$$\cot\phi = \left| r\frac{\partial\theta}{\partial r} \right| \tag{7.1}$$

where ϕ is the pitch angle, r is the radius, and θ is the rotation angle in the plane of the galaxy. In the context of logarithmic spiral arms, described by

$$r = Ae^{\theta\tan\phi} \tag{7.2}$$

where A is an amplitude and the rest of the quantities are as described above, the right hand side of Equation (7.1) is constant.

GALFIT uses a power law (cf. Eq. (2.6)—further details Appendix C), to perform the coordinate rotation that creates the appearance of the arms. This powerlaw function *does not* assume a constant pitch angle as the dependence on r does not vanish when the partial derivative is taken (C.4). This framework can therefore provide an alternative description of the pitch angle of these galaxies, one which is allowed to vary across the length of the arm.

Figure 7.5 shows the pitch angle analysis for galaxy (1) from Figure 6.1: a) a scatter plot of the pitch angle with respect to the radius, and b) an overlay of our approach, represented by the green line, on top of the model—de-inclined to face-on due to the degeneracy in the inclination with the disk axis ratio. Additional scatter and overlay plots can be found in Appendix F.

Figure 7.6a is another such plot. We present this overlay to validate our analysis and to demonstrate, at radius 15, the agreement between the derived pitch angle from (7.1) and that of the geometric definition of pitch angle, i.e. the angle between the tangent lines to the spiral and to a circle at that radius. We see excellent agreement between the two measurements—we believe the disagreement is due to our implementation of the estimate of the slope of the spiral tangent.

Figure 7.5b shows a scatter plot of the pitch angles with respect to radius for a galaxy from the 14 set. Figure 7.6b is another such plot. The green lines in each scatter plot indicate the region between the smallest inner radius and largest outer radius between the two longest arms as measured and detected by SpArcFiRe while the horizontal orange line is the approximate pitch angle, pa_alenWtd_avgdomChiralityOnly, with shading to represent the pitch angle uncertainty given by SpArcFiRe. Due to the radial dependence of GALFIT's pitch angle, it is difficult to ascribe a single value with which we can compare to SpArcFiRe.

We intend, in the future, to use the two methods to aid in determining the success of a fit but have not yet found a suitable relationship between them.

The framework is purely mathematical: neither the inner nor outer radius measures we feed to `GALFIT` represent the start and end of an arm (Peng et al., 2010). Furthermore, we are limited in our description of the arm by its representation as an mathematical line. This results in a pitch angle measurement method that's similar to the dominant mode approach of FFT methods as we are limited to evaluating the pitch angle along what is effectively a single arm. This limitation arises from our methodology: we evaluate the coordinate rotation itself on the Fourier transformed grid rather than the resultant, rotated and Fourier transformed ellipse from the image. This implementation therefore fails to capture some of the effects from the Fourier transform, most notably in the regions of the arm which are thicker or where the Fourier mode splits off a sub-arm and most often near the bulge as in Figure 7.6. As a result, the method can misconstrue the value of the pitch angle in the regions of galaxies which feature thick or partial arms or sharp bending features. We must also note an additional limitation to consider when generating the overlay that arises from the underlying ellipse rotation implementation as well as a degeneracy between the power law inclination with the disk axis ratio. In order to generate the overlays in Figure 7.5, we had to de-incline the model to face-on before applying the plot to the image.

It is unclear whether the comparison between the pitch angle values found from `SpArcFiRe` and `GALFIT` can elucidate any information about the success of our models. When examining several models which otherwise satisfied our constraints, we often found `SpArcFiRe` to underestimate the pitch angle, likely in an effort to average out the variation near the bulge and at the edge of the disk. In other cases, the `GALFIT` arm fit simply wasn't accurate enough despite passing the other tests. `SpArcFiRe`'s success in processing the resulting fits is, however, in conjunction with the residual analysis, still the best metric we have for confirming a model to be representative of a spiral galaxy.

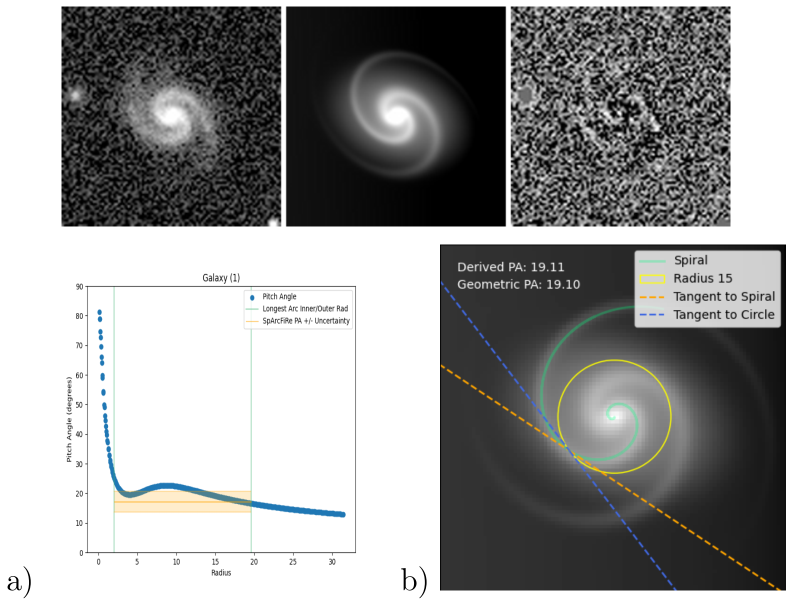


Figure 7.5: The pitch angle scatter plot with respect to radius (a) and the coordinate rotation overlay (b) for galaxy (1) from Figure 6.1, pictured above both. a) The green lines indicate the region between the smallest inner radius and largest outer radius between the two longest arms as measured and detected by `SpArcFiRe`. The horizontal line is the `pa_alenWtd_avgdomChiralityOnly` while the shading represents the uncertainty given by `SpArcFiRe`. b) Along with the coordinate rotation overlay, we plot the tangent to the circle at radius 15 (pixels) and the tangent to the overlay (geometric) to validate the pitch angle measurement produced by solving equation (7.1) at radius 15, both values of which are given in the upper left hand corner.

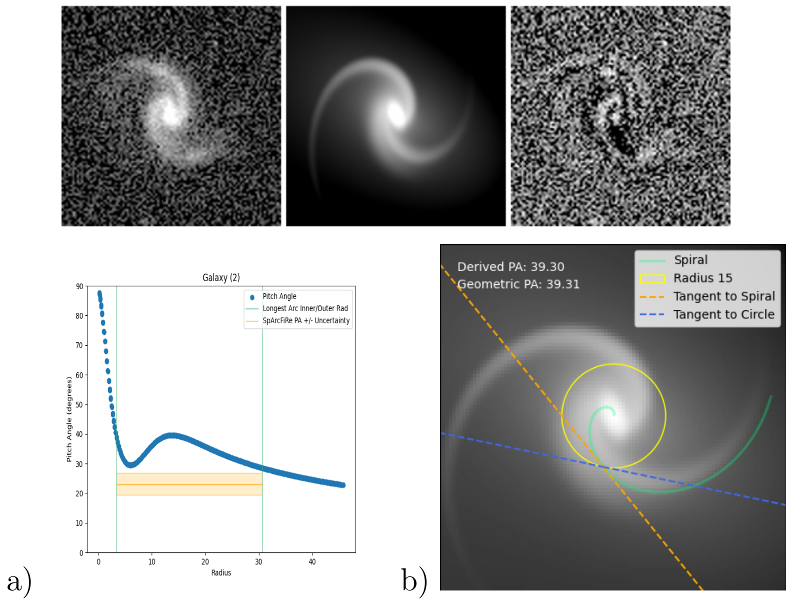


Figure 7.6: The coordinate rotation overlay and pitch angle scatter plot for galaxy (2) from Figure 6.1, showcasing the limits of our implementation.

7.2.1 `GalfitModule`: Calculating the Pitch Angle

Section 7.2 above details the means by which we can produce a measure of the pitch angle according to `GALFIT`'s coordinate rotation function. The script that performs this calculation can be found in the `PitchAngle` submodule of the `GalfitModule`, `calculate_pitch_angle.py`. When called from the command line, `calculate_pitch_angle.py` uses the FITS files output by `GALFIT` to calculate and tabulate—into a pickled `pandas` dataframe—the arrays which describe the pitch angle as it varies along the length of the arm. As columns in the dataframe, these are the pitch angles (`pitch_angle`) and rotation angles (`theta`) on the fourier transformed radial grid (`fourier_rgrid`). We also include a metric, the average pitch angle constrained by a length weighted, dominant chirality estimate of the beginning and ending of the arms in the observation (from `SpArcFiRe` and using the same scheme as detailed in Section 3.3.2) as a column, `avg_constrained_pa`.

`calculate_pitch_angle.py` also has the capability to produce plots related to the calculation of the pitch angle. The first, a scatter plot, as can be seen in Figure F.1, plots the `pitch_angle` against the `fourier_rgrid`. In these plots, we overplot the approximate average beginning and ending of the arms as well as `SpArcFiRe`'s uncertainty of the pitch angle between these two bounds. The second plot we can produce is a series of overlay plots (Figure 7.5)—alternatively identified as validation plots in the code—in which we recover the coordinate rotation function using `fourier_rgrid` and `theta` and plot this function overlaid onto an image of the de-inclined model to validate the accuracy of the function. In addition to this overlay, we provide a 'geometric' calculation of the pitch angle, in which we compute the angle between the tangents to the circle and the spiral, so as to compare with our 'derived' pitch angle value. Using both the overlay and the comparison between the 'geometric' and 'derived' pitch angle values, we can effectively validate our approximation of the pitch angle by this method. To re-iterate the discussion in Section 7.2, the purpose for doing so is twofold: 1) this approach is novel—very few pitch angle calculation methods exist which account for variation across the arm, and 2) our representation/calculation can only effectively capture a single arm. The single arm representation is generally valid for `GALFIT`'s approach given `GALFIT`'s assumption of symmetry by virtue of the underlying ellipse, however, when the amplitude of the fourier modes is high, the model arms become highly anti-symmetric and/or split off into sub-arms, and we cannot capture this behavior by our method.

Chapter 8

Conclusion

We have developed the first method to automatically generate `GALFIT` P, 2010 models on a large sample of spiral galaxies. Our files, as input to `GALFIT` (Section 2.2), are derived from the output of `SpArcFiRe` (Section 2.1), our code for an image-level description of spiral structure (Davis & Hayes, 2014)—Chapter 3. To facilitate this translation and to govern the operation of our processing pipeline, we have developed a Python module, entitled "`GalfitModule`" (Chapter B), that abstracts `GALFIT` parameters (Section B.1.1), components (Sections B.1.2 and B.1.3), and output (Section B.1.4). This module enables us to employ distributed processing (Section 4.5) as well as a variety of fitting techniques—multi-stage (Section 4.3) and parameter search fitting (Section 4.4)—which aids the pipeline in producing models of spiral galaxies which accurately map the visible structure of the original observation. To evaluate the success of these models, we used a series of techniques, including: residual or fit quality analysis (Section 5.1), quantitative spiral arm analysis (Section 5.2), and by-eye assessment to produce estimates on the ability of our pipeline to validate our results (Section 5.3). We have presented our method and results on three sub-samples of spiral galaxies from SDSS DR7, consisting of 14, 1000, and 28912 galaxies identified to be likely spiral by human volunteers as part of GZ1, Section 6.1. Of the largest sample, we estimate to have produced about 54.3% (15,700) successful models consisting of either two (bulge + arms), or three (bulge + disk + arms) components (Section 6.5). This estimate is derived from our by-eye analysis of a subsample of the 28912 galaxies and closely agrees with that found in our sample of 1000 galaxies.

We extracted, from the results of the 1000 and 28912 sample sets, features of the underlying profiles which describe the galactic components, such as the Sérsic index and integrated magnitudes, and evaluate the ability of our pipeline to generate accurate models of galaxies from these features (Sections 6.4.2 and 6.5.2). We compared our Sérsic index results to that of previous studies which created bulge + disk models from SDSS DR7 galaxies and found that our method is likely producing models which may not be readily separable by the bulge + disk convention due to a limited bulge fit. We also compared these results with other bulge, disk, and spiral arm decomposition studies performed on a much smaller scale and found our Sérsic index results to be closer to—-but still inconsistent with—the other studies. Despite this, we found that our estimates for the magnitude of several components were consistent with the results of the above studies. We additionally found general agreement in our estimates of the ratio of spiral arm light to that of the total light of the galaxy in the model.

Finally, we presented a method and results for evaluating the pitch angles of the models as derived from `GALFIT`'s rotation implementation (Section 7.2). We demonstrated the feasibility of the method and highlighted preliminary results for one of the galaxies from the set of 14 (the rest of which can be found in Appendix F).

Chapter 9

Future Work

9.1 Pipeline Improvements

9.1.1 Improving the Models

The results we have presented in Chapter 6 demonstrate the ability of the models produced by our n-component method to fit their respective observations. It is still possible, however, to improve these models in bulk across both NC2 and NC3 and to additionally include other methods and techniques to do so. We anticipate additional methods may include the use of truncation functions, PSFs, or a modification to `GALFIT` source code (Section B) to favor, during its χ^2 fitting process Eq. (2.1), the region beyond the bulge but we must caution here against adding too much complexity. Of techniques, the simultaneous fitting technique (Simard et al., 2002; Simard et al., 2011) is of great interest. This technique works by fitting secondary (nearby) sources rather than simply masking these sources as their contribution to the overall flux in the image can have an effect on the resultant decomposition. `GALAPAGOS-C` (Hiemer et al., 2014) contains a substantial breakdown of this process which could be consulted for its implementation.

In addition to simultaneous fitting, Simard et al. (2011) employ a cross-band fitting technique which works by fitting images in one band before using the decomposition to fit in another band, commonly r and g (e.g. Simard et al., 2011; Lackner & Gunn, 2012; Kim et al., 2016). This is done to minimize fitting errors with the drawback that color gradients cannot be measured between the simultaneously fit bands. In the context of spiral galaxies, comparing color gradients (as in Pour-Imani et al., 2016, also see Section 9.4) can potentially illuminate important morphological characteristics, e.g. pattern speed and star formation rate (Binney & Tremaine, 2008), so this comparison may play a crucial role in future automated studies. Despite this drawback, it is still possible to use simultaneous fitting across bands which may contain little or redundant morphological information to then inform the fit across bands we are interested in comparing, hence why we intend to include it.

Another means of improvement could come from the development done on `SpArcFiRe`, specifically the "mean-image guiding" technique first developed and integrated with `SpArcFiRe` by Alan Barkley-Yeung, an undergraduate student in our lab. Mean-image guiding uses a composite image of a galaxy across different color bands, pixel aligned by an automated algorithm our group developed (Navarrette & Hayes, 2022), and generates a cluster mask of this image (cluster masks are the colored regions in panels **d** and **e** in Figure 2.1) that `SpArcFiRe` can then use to inform its analysis of the galaxy in each individual wave band. This method will prove advantageous in its ability to correct for fitting issues across all wave bands whether they occur due to ill-conditioning of input parameters or whether they are due to physical/observational phenomena such as dust obscuring, source contamination, low signal-to-noise ratio, etc. We therefore expect a mean-guided image to provide an excellent basis for generating our final band-dependent models. We can furthermore use this technique in conjunction with simultaneous fitting to provide increased robustness without sacrificing the ability to compare across wave bands. To do this, we will use `SpArcFiRe` composite image output and `GALFIT`'s subsequent convolution as our template for re-running

GALFIT on the images from each waveband—a relatively trivial procedure and ultimately low in computational cost.

Finally, and of great importance, is the issue of overfitting of the bulge, discussed in Section 6.4.2. We cannot yet say whether the overfitting found is indicative of deeper problems within our translation framework or whether it is a consequence of the constraints of automating GALFIT's spiral arm decomposition method. We anticipate it is the former given the inherent sensitivity of modeling the bulge especially with respect to the strides previous studies have taken to accommodate for it. If it is the latter, it will be necessary to identify the extent to which these models are appropriate for precision morphological analysis. We do not believe this possibility invalidates our method however as these models can still offer a wealth of information about general morphological trends. We therefore intend to explore this issue and potential resolutions in full. We also intend to experiment with the application of our algorithm to existing B+D decompositions as was done in Chugunov et al. (2023). This may solve the overfitting issue or otherwise highlight where we can fix the original pipeline to avoid it.

9.1.2 Improving Validation

One of the biggest challenges that we faced in the development of the pipeline was our inability to automatically validate our results. The methods presented in Chapter 5 are the result of years of work and experimentation, yet there is still much we can do to better assess our results in terms of the false positive (FP) and false negative (FN) rates. In our current framework, we employ our metrics as threshold cutoffs (Sections 5.1 and 5.2)—based on our intuition about the comparison between model and observation—after which a model can longer be considered 'successful'. Although doing so allows us some manner of simplicity, we believe it should be possible to employ a weighting scheme for the four metrics, whether weighted together, i.e. threshold success boolean $\cdot$ weight, or individually, i.e. threshold value (with respect to our best guess for an appropriate value) · weight. Given our current results per the discussion of the cutoff values in Table 6.1, we anticipate that the former would provide flexibility across data sets while the latter would better be able to tune the FP/FN rates. We also anticipate that a weighting scheme would benefit greatly from the use of a neural network or similar regression based analysis as discussed below (Section 9.2).

Beyond the tuning of the metrics are potential improvements in both 1) the preparation of the output for `SpArcFiRe` and, more importantly, 2) in tuning `SpArcFiRe` itself. Of 1) we presented our method of reducing the arm width in preparation for `SpArcFiRe` in Section 5.2. This method appeared to improve the FP/FN rates however it is possible that some other adjustment—or some other choice of arm width—at that stage of the pipeline may improve `SpArcFiRe`'s ability to evaluate the model. As a first attempt, we recommend exploring a means to reduce the light at the center of the rotation in order to provide more contrast, given the possibility of `SpArcFiRe` identifying false arms in that region as in Figures 7.3 and 7.4. This may be achieved by reducing the coordinate rotation's outer radius or by reducing the Sérsic index or half light radius of the underlying Sérsic function, however it is not immediately clear which or what combination of these may work best.

Of 2) it is clear, even in the FP/FN figures (7.4/7.3) that `SpArcFiRe` is capable of successfully tracing out the arms of the models given the limitations of the logarithmic spiral arc formulation. However, we can also identify two issues of note: a) the contrast of the model image and b) the need for consistency of the image standardization between observation and model.

2a) Without a well-defined, bright bulge, the conversion from FITS to PNG flattens the diffusion of light we would otherwise see in a real image of a galaxy. This is an issue we encountered in our rebuttal (Portman et al., 2023) of another work by Hewitt & Treuthardt (2020) in which the authors used a pure disk of light and arms to model a spiral galaxy—without tuning `SpArcFiRe` correctly, they produced erroneous results concerning the performance of `SpArcFiRe`. When tuned correctly, we were able to almost-perfectly analyze their models. This leads us to believe that a solution, or an improvement to the current FP/FN rates *can* be found in either the conversion from FITS to PNG (via `ImageMagick`) or in the parameterization of `SpArcFiRe`'s contrast enhancement/image standardization itself.

2b) The model in Figure 7.3, identified as FN, was labeled 'unsuccessful' due to the pitch angle metric for the set of 14 in Table 6.1. This can only be explained by the difference in the image standardization used by `SpArcFiRe` on both images. This difference can be remedied by one of `SpArcFiRe`'s features: `SpArcFiRe` outputs a file, after performing its analysis, that details the parameterization of the image standardization which can be re-used to exactly duplicate the standardization employed. We attempted to use this technique when preparing our results but saw no clear improvement in the FP/FN rates. We believe, however, that this failure simply requires more time and attention to resolve and should greatly improve the success of the validation method.

We discuss, in Section 7.2, the possibility of finding some means of comparing `GALFIT`'s pitch angle analysis with `SpArcFiRe` as another metric for determining success. We believe this is still worth examining. However, the pitch angle analysis provides another avenue through which the FP/FN rates could be improved: the comparison of the arm overlay as pictured in Figure F.1 with that of `SpArcFiRe`'s arc overlay as pictured in Figure 2.1f. Given the image-level simplicity of both overlays, it should be feasible, whether through computer vision or machine learning, to assess the similarity between the arcs. Finding a method for doing so would save much computation time given that `SpArcFiRe` would no longer have to be run after `GALFIT` has completed its processing.

9.2 Machine Learning

Work is ongoing to incorporate machine learning techniques to supplement the input to `GALFIT` via regression. We are currently prototyping a supervised machine learning regressor using `XGBoost` Chen & Guestrin (2016), trained on good model fits as identified by visual classification, to supplement the input given by `SpArcFiRe`. Our aim is to *supplement* `SpArcFiRe`'s output rather than to replace it entirely to allow for the ability to interpret the decisions made by the regressor. We believe the ability to interpret the results in this context will allow us to glean useful information which may help to improve the pipeline. Additionally, we intend to incorporate a classifier which may better assist us in identifying when `GALFIT` has failed to generate a reasonable model of the arms of a galaxy. We believe we can derive a prototype of this classifier from the regressor by identifying when an output model does not align with what the regressor may deem as a successful model. Azra Zahin has began work on the development of a convolutional neural networks to be used on the output images but this work has halted upon her departure from the project.

9.3 Completion of the `GalfitModule` for Release

As discussed in Chapter B, the `GalfitModule` is incomplete and more work must be done to better generalize the existing framework insofar as the `GALFIT` objects are concerned. Since the `GalfitModule` is deeply intertwined with the fitting pipeline, many of its additional utilities are being updated regularly but the core functionality does not change.

We also plan to update the pipeline of the `GalfitModule` to streamline many of its features and to better incorporate those features into `SpArcFiRe`. This includes updates to its ability to handle even larger data sets, when disk space is a concern, as well as the inclusion of a machine learning framework discussed below. As discussed briefly in section 7, using the 1000 galaxy subset as a proof-of-concept for the generality of our pipeline, we hope to use our method to model the approximately 900,000 galaxies from SDSS DR7 in order to compare our results to other automated DR7 surveys such as Simard et al. (2011); Meert et al. (2014) and to compare our `GALFIT` pipeline to others such as `PyMorph` (Vikram et al., 2010), `GALAPAGOS` (Barden et al., 2012; Häußler et al., 2013), `MORPHOFIT` (Tortorelli & Mercurio, 2023), and more. We also intend to apply the pipeline to other data sets such as later SDSS data releases, LSST (Ivezić et al., 2019), and those from cosmological simulations such as Illustris (Vogelsberger et al., 2014; Pillepich et al., 2018), FIRE (Hopkins et al., 2014, 2018) and NIHAO (Wang et al., 2015).

9.4 Comparison with Theory

We intend to use this pipeline to explore the predictions of spiral density wave theory and to assist in the quantification of spiral galaxy morphology. The pitch angle analysis that we provide here (and which is included in the `GalfitModule`'s utilities) will be the foundation for this work, through which we will be able to test several predictions of the theory and relationships between it and other galactic features. We intend to examine several of the pitch-angle relationships described in Section 2.3 of the review of spiral galaxy literature by Sellwood & Masters (2022), starting with our results for the galaxies of SDSS DR7 and expanding to other data sets when possible.

We expect that, according to spiral density wave theory (Lin & Shu, 1964; Binney & Tremaine, 2008), observations of the arms of a spiral galaxy in different imaging bands should show a difference in their pitch angles, with gas and younger stars at the leading edge and older stars at the trailing edge, and the arms crossing at the co-rotation radius, or the radius at which the angular velocity of the stars in the arms match the pattern speed of the arm (Binney & Tremaine, 2008). Pour-Imani et al. (2016) investigated this phenomena and found that the pitch angles of the galaxies they selected were different across different bands. We believe that our pipeline and pitch angle analysis can also be used for this purpose.

Figure 9.1 shows our preliminary results concerning this comparison: pitch angle scatter plots for r and g band NC3 models from the set of 14—as a reminder, the observations used elsewhere in this thesis are from the r-band (Section 6.1). In these plots, we see that there is consistently a difference between r and g bands. However, Galaxy (1) differs from the others in that the pitch angle in the r-band is mostly *lower* than the g-band. We are unsure what the implications of this inconsistency are—it is possible that, despite Galaxy (1) having the lowest fit quality score across both bands, the arm fit is not as good as in the others or vice-versa. It is also possible, given `GALFIT`'s sensitivity to magnitude, that minor variations in fit parameters could alter the arm fit enough to change these results. A sensitivity analysis of this relationship will be crucial to the validation of this comparison method.

Beyond the sensitivity analysis, much work must be done to handle the variety of behavior we see in the pitch angle. In the g-band pitch angle scatter plot for Galaxy (4), we see a peak/discontinuity at about radius 12.5. This peak is known behavior—it occurs due to a negative power law index which reverses the direction of the winding (Section 2.2). Similarly, in the g-band plot for Galaxy (7), we see a minima around radius 9 which may be due to a relatively strong Fourier amplitude in the m1 mode. It is not clear whether this behavior should be accounted for given `GALFIT`'s sensitivity in the region near the bulge. In both cases, the parameterization of the rotation and Fourier functions holds the key to identifying these behaviors. It may be prudent then to use machine learning to identify the (combination of) parameters that cause 'problem' behaviors to occur and whether or not these problems indicate a poor model fit. In any case, work must be done to determine a means to identify and to potentially correct for these discrepancies.

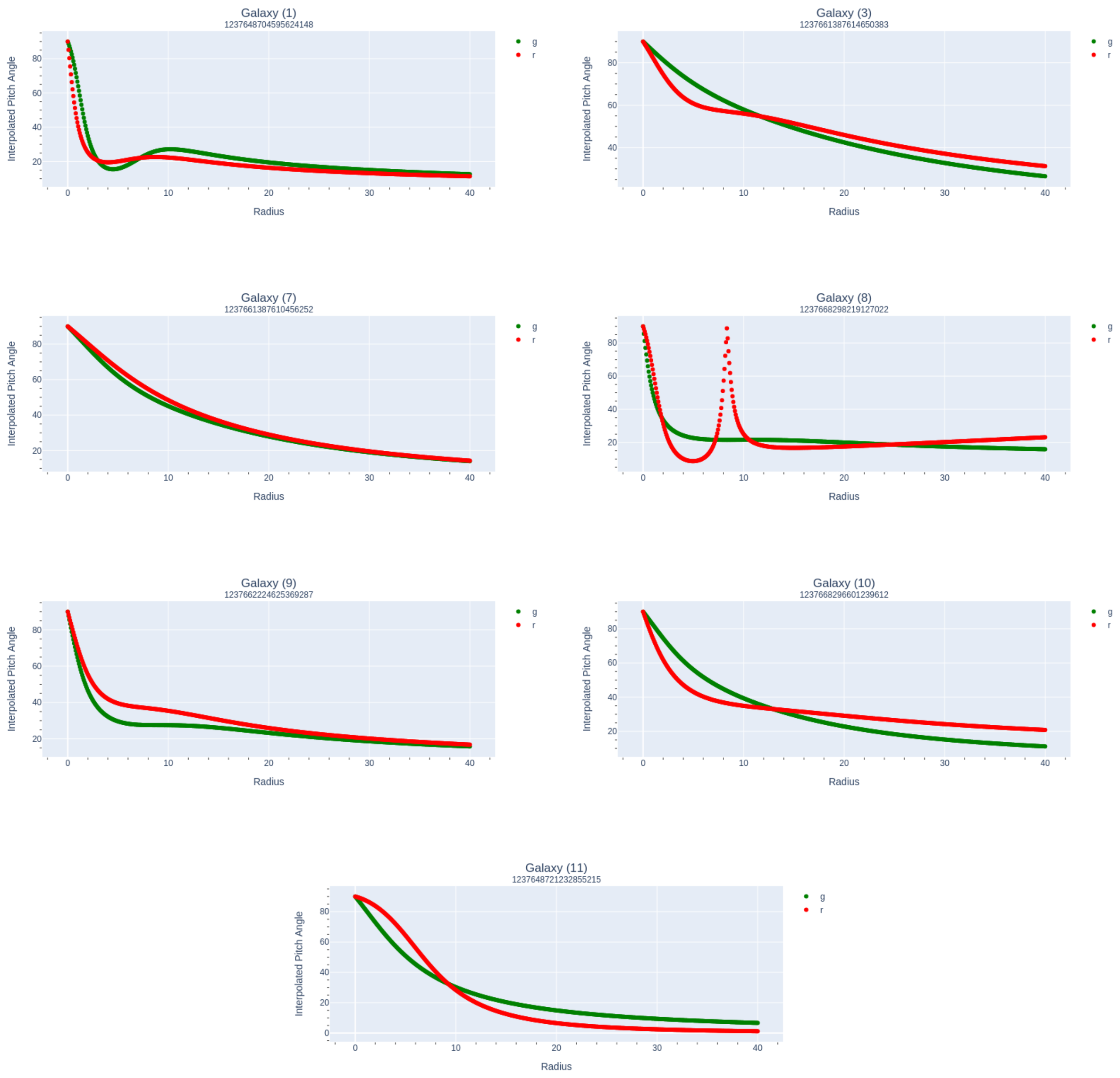


Figure 9.1: Pitch angle scatter plots for four NC3 models from the set of 14 successfully fit in both the r and g SDSS DR7 imaging bands.

Bibliography

Abazajian, K. N., Adelman-McCarthy, J. K., Agüeros, M. A., et al. 2009, Astrophysical Journal, Supplement, 182, 543, doi: 10.1088/0067-0049/182/2/543

Adelman-McCarthy, J. K., Agüeros, M. A., Allam, S. S., et al. 2006, The Astrophysical Journal Supplement Series, 162, 38, doi: 10.1086/497917

Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., et al. 2022, apj, 935, 167, doi: 10.3847/1538-4357/ac7c74

Barden, M., Häußler, B., Peng, C. Y., McIntosh, D. H., & Guo, Y. 2012, Monthly Notices of the Royal Astronomical Society, 422, 449

Bell, E. F., Naab, T., McIntosh, D. H., et al. 2006, The Astrophysical Journal, 640, 241

Benson, A. J., Frenk, C. S., & Sharples, R. M. 2002, Astrophysical Journal, 574, 104, doi: 10.1086/340925

Bertin, E., & Arnouts, S. 1996, Astronomy & Astrophysicss, 117, 393, doi: 10.1051/aas:1996164

Binney, J., & Tremaine, S. 2008, Galactic Dynamics, 2nd edn. (Princeton, New Jersey, USA: Princeton University Press)

Blanton, M. R., Kazin, E., Muna, D., Weaver, B. A., & Price-Whelan, A. 2011, The Astronomical Journal, 142, 31, doi: 10.1088/0004-6256/142/1/31

Bouwens, R. J., Illingworth, G., Oesch, P., et al. 2015, The Astrophysical Journal, 803, 34

Byun, Y. I., & Freeman, K. C. 1995, Astrophysical Journal, 448, 563, doi: 10.1086/175986

Chen, T., & Guestrin, C. 2016, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM), doi: 10.1145/2939672.2939785

Chugunov, I. V., Marchuk, A. A., Mosenkov, A. V., et al. 2023, Monthly Notices of the Royal Astronomical Society, 527, 9605, doi: 10.1093/mnras/stad3850

Davis, B. L., Berrier, J. C., Shields, D. W., et al. 2012, Astrophysical Journal, Supplement, 199, 33, doi: 10.1088/0067-0049/199/2/33

Davis, D. R. 2014, PhD thesis, University of California, Irvine

Davis, D. R., & Hayes, W. B. 2012, in 2012 IEEE Conference on Computer Vision and Pattern Recognition, 1138–1145

Davis, D. R., & Hayes, W. B. 2014, Astrophysical Journal, 790, 87, doi: `10.1088/0004-637x/790/2/87`

de Souza, R. E., Gadotti, D. A., & dos Anjos, S. 2004, ApJs, 153, 411, doi: `10.1086/421554`

de Vaucouleurs, G. 1948, Annales d'Astrophysique, 11, 247

Dewsnap, C., Barmby, P., Gallagher, S. C., et al. 2023, The Astrophysical Journal, 944, 137, doi: `10.3847/1538-4357/ac9400`

Díaz-García, S., Salo, H., Knapen, J. H., & Herrera-Endoqui, M. 2019, Astronomy & Astrophysics, 631, A94, doi: `10.1051/0004-6361/201936000`

Domínguez Sánchez, H., Margalef, B., Bernardi, M., & Huertas-Company, M. 2021, Monthly Notices of the Royal Astronomical Society, 509, 4024, doi: `10.1093/mnras/stab3089`

F. Hroch. 2019, fitspng, `http://integral.physics.muni.cz/fitspng/`

Freeman, K. C. 1970, Astrophysical Journal, 160, 811, doi: `10.1086/150474`

Gao, H., & Ho, L. C. 2017, The Astrophysical Journal, 845, 114, doi: `10.3847/1538-4357/aa7da4`

Grosbøl, P., Patsis, P. A., & Pompei, E. 2004, Astronomy & Astrophysics, 423, 849, doi: `10.1051/0004-6361:20035804`

Guo, Y., McIntosh, D. H., Mo, H. J., et al. 2009, Monthly Notices of the Royal Astronomical Society, 398, 1129, doi: `10.1111/j.1365-2966.2009.15223.x`

Hart, R. E., Bamford, S. P., Willett, K. W., et al. 2016, Monthly Notices of the Royal Astronomical Society, 461, 3663, doi: `10.1093/mnras/stw1588`

Hart, R. E., Bamford, S. P., Hayes, W. B., et al. 2017, Monthly Notices of the Royal Astronomical Society, 472, 2263, doi: `10.1093/mnras/stx2137`

Hayes, W. B., Davis, D., & Silva, P. 2016, Monthly Notices of the Royal Astronomical Society, 466, 3928, doi: `10.1093/mnras/stw3290`

Hewitt, I., & Treuthardt, Patrick, P. D. 2019, treuthardt/P2DFFT: P2DFFT Version 5.2.2, v5.2.2, Zenodo, doi: `10.5281/zenodo.3238638`

Hewitt, I. B., & Treuthardt, P. 2020, Monthly Notices of the Royal Astronomical Society, 493, 3854, doi: `10.1093/mnras/staa354`

Hiemer, A., Barden, M., Kelvin, L. S., Häußler, B., & Schindler, S. 2014, Monthly Notices of the Royal Astronomical Society, 444, 3089

Hopkins, P. F., Kereš, D., Oñorbe, J., et al. 2014, Monthly Notices of the Royal Astronomical Society, 445, 581, doi: `10.1093/mnras/stu1738`

Hopkins, P. F., Wetzel, A., Kereš, D., et al. 2018, Monthly Notices of the Royal Astronomical Society, 480, 800, doi: 10.1093/mnras/sty1690

Häussler, B., McIntosh, D. H., Barden, M., et al. 2007, The Astrophysical Journal Supplement Series, 172, 615, doi: 10.1086/518836

Häußler, B., Bamford, S. P., Vika, M., et al. 2013, Monthly Notices of the Royal Astronomical Society, 430, 330, doi: 10.1093/mnras/sts633

Inc., P. T. 2015, Collaborative data science, Montreal, QC: Plotly Technologies Inc. https://plot.ly

Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, Astrophysical Journal, 873, 111, doi: 10.3847/1538-4357/ab042c

Kartaltepe, J. S., Mozena, M., Kocevski, D., et al. 2015, The Astrophysical Journal Supplement Series, 221, 11

Kelvin, L. S. e. a. 2012, Monthly Notices of the Royal Astronomical Society, 421, 1007

Kendall, S., Kennicutt, R. C., & Clarke, C. 2011, Monthly Notices of the Royal Astronomical Society, 414, 538, doi: 10.1111/j.1365-2966.2011.18422.x

Kent, S. M. 1985, Astrophysical Journal, Supplement, 59, 115, doi: 10.1086/191066

Kim, K., Oh, S., Jeong, H., et al. 2016, The Astrophysical Journal Supplement Series, 225, 6, doi: 10.3847/0067-0049/225/1/6

Kormendy, J. 1977, Astrophysical Journal, 217, 406, doi: 10.1086/155589

Lackner, C. N., & Gunn, J. E. 2012, Monthly Notices of the Royal Astronomical Society, 421, 2277, doi: 10.1111/j.1365-2966.2012.20450.x

Lin, C. C., & Shu, F. H. 1964, Astrophysical Journal, 140

Lingard, T., Masters, K. L., Krawczyk, C., et al. 2021, Monthly Notices of the Royal Astronomical Society, 504, 3364, doi: 10.1093/mnras/stab1072

Lingard, T. K., Masters, K. L., Krawczyk, C., et al. 2020, The Astrophysical Journal, 900, 178, doi: 10.3847/1538-4357/ab9d83

Lintott, C., Schawinski, K., Bamford, S., et al. 2011a, Monthly Notices of the Royal Astronomical Society, 410, 166, doi: 10.1111/j.1365-2966.2010.17432.x

—. 2011b, Monthly Notices of the Royal Astronomical Society, 410, 166, doi: 10.1111/j.1365-2966.2010.17432.x

Lisker, T. 2008, The Astrophysical Journal Supplement Series, 179, 319, doi: 10.1086/591795

Läsker, R., Ferrarese, L., & van de Ven, G. 2013, The Astrophysical Journal, 780, 69, doi: 10.1088/0004-637X/780/1/69

Masters, K. L., Lintott, C. J., Hart, R. E., et al. 2019, Monthly Notices of the Royal Astronomical Society, 487, 1808, doi: 10.1093/mnras/stz1153

Meert, A., Vikram, V., & Bernardi, M. 2013, Monthly Notices of the Royal Astronomical Society, 433, 1344

—. 2014, Monthly Notices of the Royal Astronomical Society, 446, 3943, doi: 10.1093/mnras/stu2333

—. 2015, Monthly Notices of the Royal Astronomical Society, 455, 2440, doi: 10.1093/mnras/stv2475

Mutlu-Pakdil, B., Seigar, M. S., Hewitt, I. B., et al. 2018, Monthly Notices of the Royal Astronomical Society, 474, 2594, doi: 10.1093/mnras/stx2935

Navarrette, A. L., & Hayes, W. 2022, Astronomy and Computing, 40, 100611, doi: https://doi.org/10.1016/j.ascom.2022.100611

Peng, C. Y., Ho, L. C., Impey, C. D., & Rix, H.-W. 2010, The Astronomical Journal, 139, 2097, doi: 10.1088/0004-6256/139/6/2097

Pillepich, A., Springel, V., Nelson, D., et al. 2018, Monthly Notices of the Royal Astronomical Society, 473, 4077, doi: 10.1093/mnras/stx2656

Portman, M. E., Mesforoush, S., & Hayes, W. B. 2023, Monthly Notices of the Royal Astronomical Society, 526, 830

Pour-Imani, H., Kennefick, D., Kennefick, J., et al. 2016, The Astrophysical Journal Letters, 827, L2, doi: 10.3847/2041-8205/827/1/L2

Ringermacher, H. I., & Mead, L. R. 2009, Monthly Notices of the Royal Astronomical Society, 397, 164, doi: 10.1111/j.1365-2966.2009.14950.x

Robotham, A. S. G., Taranu, D. S., Tobar, R., Moffett, A., & Driver, S. P. 2016, Monthly Notices of the Royal Astronomical Society, 466, 1513, doi: 10.1093/mnras/stw3039

Savchenko, S., Marchuk, A., Mosenkov, A., & Grishunin, K. 2020, Monthly Notices of the Royal Astronomical Society, 493, 390, doi: 10.1093/mnras/staa258

Savchenko, S. S., & Reshetnikov, V. P. 2013, Monthly Notices of the Royal Astronomical Society, 436, 1074, doi: 10.1093/mnras/stt1627

Schade, D., Lilly, S. J., Le Fevre, O., Hammer, F., & Crampton, D. 1996, Astrophysical Journal, 464, 79, doi: 10.1086/177301

Seigar, M. S., Block, D. L., Puerari, I., Chorney, N. E., & James, P. A. 2005, Monthly Notices of the Royal Astronomical Society, 359, 1065, doi: 10.1111/j.1365-2966.2005.08970.x

Sellwood, J., & Masters, K. L. 2022, Annual Review of Astronomy and Astrophysics, 60, 73, doi: 10.1146/annurev-astro-052920-104505

Sersic, J. L. 1968, Atlas de Galaxias Australes (Observatorio Astronomico)

Sheth, K., Regan, M., Hinz, J. L., et al. 2010, Publications of the Astronomical Society of the Pacific, 122, 1397

Silva, P., Cao, L. T., & Hayes, W. B. 2018, Galaxies, 95

Simard, L., Mendel, J. T., Patton, D. R., Ellison, S. L., & McConnachie, A. W. 2011, The Astrophysical Journal Supplement Series, 196, 11, doi: `10.1088/0067-0049/196/1/11`

Simard, L., Willmer, C. N. A., Vogt, N. P., et al. 2002, Astrophysical Journal, Supplement, 142, 1, doi: `10.1086/341399`

Sonnenfeld, Alessandro. 2022, A&A, 659, A141, doi: `10.1051/0004-6361/202142786`

Sparke, L. S., & Gallagher, John S., I. 2007, Galaxies in the Universe: An Introduction (Cambridge University Press)

Stoughton, C., Lupton, R. H., Bernardi, M., et al. 2002, The Astronomical Journal, 123, 485, doi: `10.1086/324741`

The ImageMagick Development Team. 2020, ImageMagick, 7.0.10. `https://imagemagick.org`

The Pandas Development Team. 2020, pandas-dev/pandas: Pandas, latest, Zenodo, doi: `10.5281/zenodo.3509134`

Tortorelli, L., & Mercurio, A. 2023, Frontiers in Astronomy and Space Sciences, 10, 989443

van der Wel, A., Bell, E., Häussler, B., et al. 2012, The Astrophysical Journal Supplement Series, 203, 24

Vikram, V., Wadadekar, Y., Kembhavi, A. K., & Vijayagovindan, G. V. 2010, Monthly Notices of the Royal Astronomical Society, 409, 1379, doi: `10.1111/j.1365-2966.2010.17426.x`

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261, doi: `10.1038/s41592-019-0686-2`

Vogelsberger, M., Genel, S., Springel, V., et al. 2014, Monthly Notices of the Royal Astronomical Society, 444, 1518, doi: `10.1093/mnras/stu1536`

Von Der Linden, A., Best, P. N., Kauffmann, G., & White, S. D. M. 2007, Monthly Notices of the Royal Astronomical Society, 379, 867, doi: `10.1111/j.1365-2966.2007.11940.x`

Wadadekar, Y., Robbason, B., & Kembhavi, A. 1999, Astronomical Journal, 117, 1219, doi: `10.1086/300754`

Wang, L., Dutton, A. A., Stinson, G. S., et al. 2015, Monthly Notices of the Royal Astronomical Society, 454, 83, doi: `10.1093/mnras/stv1937`

Willett, K. W., Lintott, C. J., Bamford, S. P., et al. 2013, Monthly Notices of the Royal Astronomical Society, 435, 2835, doi: `10.1093/mnras/stt1458`

Yu, S.-Y., & Ho, L. C. 2018, Astrophysical Journal, 869, 29, doi: `10.3847/1538-4357/aaeacd`

Yu, S.-Y., & Ho, L. C. 2020, The Astrophysical Journal, 900, 150, doi: 10.3847/1538-4357/abac5b

Yu, S.-Y., Ho, L. C., Barth, A. J., & Li, Z.-Y. 2018, The Astrophysical Journal, 862, 13, doi: 10.3847/1538-4357/aacb25

Appendix A

Sample `GALFIT` Input File

A sample input file like those generated as part of our study. This file displays the general structure of the file `GALFIT` takes as input as well as the components we have selected to use throughout the study to model galaxies for the NC3 structure. We cannot encode our multi-stage fit nor the parameter search fitting technique in this simple representation but these two methods are simply modifications of this input.

```
===================================================================
# IMAGE and GALFIT CONTROL PARAMETERS
A) galaxy_name.fits # Input data image (FITS file)
B) galaxy_name_galfit_out.fits # Output data image block
C) none # Sigma image name (made from data if blank or 'none')
D) galaxy_name_psf.fits # Input PSF image and (optional) diffusion kernel
E) 1  # PSF fine sampling factor relative to data
F) galaxy_name_star-rm.fits # Bad pixel mask
G) none  # File with parameter constraints (ASCII file)
H) 37      85   37   85 # Image region to fit (xmin xmax ymin ymax)
I) 50      50    # Size of the convolution box (x y)
J) 24.8    # Magnitude photometric zeropoint
K) 0.396   0.396    # Plate scale (dx dy)   [arcsec per pixel]
O) regular    # Display type (regular, curses, both)
P) 0    # Choose: 0=optimize, 1=model, 2=imgblock, 3=subcomps
====================================================================
```

Figure A.1: GALFIT input file header for Galaxy ID 1237667429564547261. Descriptions of each parameter can be found as a comment in the figure.

```
===================================================================
# INITIAL FITTING PARAMETERS
#
#   For component type, the allowed functions are:
#       sersic, expdisk, edgedisk, devauc, king, nuker, psf,
#       gaussian, moffat, ferrer, and sky.
#
#   Hidden parameters will only appear when they're specified:
#       Bn (n=integer, Bending Modes).
#       C0 (diskyness/boxyness),
#       Fn (n=integer, Azimuthal Fourier Modes).
#       R0-R10 (coordinate rotation, for creating spiral structures).
#       To, Ti, T0-T10 (truncation function).
#
# ------------------------------------------------------------------
#   par)    par value(s)    fit toggle(s)    # parameter description
# ------------------------------------------------------------------

# Component number: 1
 0) sersic              # Component type
 1) 60.5588 60.5802 0 0 # Position x, y
 3) 14.4900     1       # Integrated magnitude
 4) 12.0600     1       # R_e (effective radius)   [pix]
 5) 1.1600      1       # Sersic index n (de Vaucouleurs n=4)
 9) 0.6500      1       # Axis ratio (b/a)
10) 23.0200     1       # Position angle (PA) [deg: Up=0, Left=90]

# Component number: 2
 0) sersic              # Component type
 1) 60.5588 60.5802 0 0 # Position x, y
 3) 15.0000     1       # Integrated magnitude
 4) 8.9878      1       # R_e (effective radius)   [pix]
 5) 4.0000      1       # Sersic index n (de Vaucouleurs n=4)
 9) 0.6000      1       # Axis ratio (b/a)
10) -14.9366    1       # Position angle (PA) [deg: Up=0, Left=90]

# Component number: 3
 0) sersic              # Component type
 1) 60.5588 60.5802 0 0 # Position x, y
 3) 15.0000     1       # Integrated magnitude
 4) 10.9878      1       # R_e (effective radius)   [pix]
 5) 4.0000      1       # Sersic index n (de Vaucouleurs n=4)
 9) 0.6000      1       # Axis ratio (b/a)
10) -14.9366    1       # Position angle (PA) [deg: Up=0, Left=90]
```

```
R0) power               # PA rotation func. (power, log, none)
R1) 0.0000      0       # Spiral inner radius [pixels]
R2) 25.8517     0       # Spiral outer radius [pixels]
R3) -60.0000    1       # Cumul. rotation out to outer radius [degrees]
R4) 1.4824      1       # Asymptotic spiral powerlaw
R9) -51.0979    1       # Inclination to L.o.S. [degrees]
R10) 90.0000     1      # Sky position angle
F1) 0.0500  45.0000 1 1 # Azim. Fourier mode 1, amplitude, & phase angle
F3) 0.0500  25.0000 1 1 # Azim. Fourier mode 3, amplitude, & phase angle

# Component number: 4
 0) sky                # Component type
 1) 1000.00   1        # Sky background at center of fitting region [ADUs]
 2) 0.00      1        # dsky/dx (sky gradient in x)     [ADUs/pix]
 3) 0.00      1        # dsky/dy (sky gradient in y)     [ADUs/pix]

 ================================================================
```

Figure A.2: Example of `GALFIT` input file components used in this study for a NC3 model. The first component is parameterized to represent the bulge using a Sérsic profile. The second component is parameterized to represent the disk using a Sérsic profile. The third component is parameterized similarly to the disk but is modified by the power function (R0 – R10) which performs a coordinate rotation on the this component to represent the spiral arms. The fourier modes (F1–F5) further refine the spiral arm fit. Finally, the fourth component, the sky, estimates the brightness of the background sky. For a complete example of a `GALFIT` input file with all profiles available, please see `GALFIT`'s website.

Appendix B

GalfitModule

GALFIT is written in C++ and can be downloaded as either an executable from GALFIT's webpage or built from source linked in the about page of GALFIT's Facebook group.[1] In this work, we opt to use the executable version of GALFIT and have developed an object-oriented library in python to handle its input and output entitled GalfitModule. The GalfitModule is currently packaged with SpArcFiRe on GitHub but plans to release it as an independent package are underway. In its current state, the GalfitModule is written in two parts: the first, to generically handle GALFIT components, models, and output via object oriented programming, and the second, to explicitly handle the naming conventions and choices of SpArcFiRe to automate the batch processing of spiral galaxies, i.e. the pipeline. In the current version of the code, the pipeline has taken precedence so some choices and restrictions related to modeling spiral galaxies are currently hard coded. We will disentangle the two and generalize the module by the time of its formal release.

[1]Prior to the author's discovery of the available source code, it is indicated on the website that source would not be distributed due to the use of proprietary algorithms. The choices made in developing the current iteration of the GalfitModule are informed by this limitation.

B.1 Fundamental Classes

The object oriented segment of the `GalfitModule` is the basis upon which much of the pipeline relies. We utilize four main class types: parameters, components, containers, and fits handlers to process the information through the pipeline. These four types, in that order, form the main operating hierarchy of the `GalfitModule`. These class types are submodules of the `Classes` submodule of the `GalfitModule` and are stored in two forms: as a `.py` file and as a jupyter notebook (`.ipynb`) file. The latter is used for documentation and for prototyping and in all cases, outputs the `.py` version of the file in the last cell of the notebook. All notebooks, and therefore all `.py` files, contain unit testing code. In the case of the notebooks, when all cells are run sequentially, the unit tests must pass before the notebook can be exported to a standalone python script. This unit testing standard applies to the `.py` files generated, therefore, all submodules can also be run as standalone scripts for unit testing purposes.

B.1.1 `GalfitParameter`

The parameters, stored in the `Parameters` submodule and based on a parent `GalfitParameter` class, are abstracted `GALFIT` parameters, written primarily to store numeric values and other extraneous information used for creating standalone feedme files, and to set parameters as fixed or free during the fitting process. The required arguments, attributes, methods, and overrides of this class are:

REQUIRED ARGUMENTS

· `value` – the generic type value of the parameter to be stored by the object.

ATTRIBUTES

· `value` – the `value` from REQUIRED ARGUMENTS.

- `fix` – abstracts the binary value (represented as integer 1's or 0's) which indicates to `GALFIT` whether to hold the value fixed during fitting.
- `parameter_number` – stores the number which corresponds with the parameter being instantiated as used by the `GALFIT` function.
- `parameter_prefix` – stores the prefix used in generating the `GALFIT` feedme which corresponds to the chosen `GALFIT` function.
- `comment` – stores the comment associated with the parameter which describes the parameters use.
- `component_name` – stores the name of the component the parameter belongs to.
- `component_number` – stores the number of the component the parameter belongs to.

  OVERRIDES

- `__str__` – overrides `str()`. Is used for outputting a string formatted in the style of the `GALFIT` feedme.
- `__repr__` – overrides `repr()`. Is used for outputting a string containing solely the `value` of the object. We choose to do so to provide what we anticipate to be the desired output in the case of debugging, in contrast to the full feedme-style output of the `str` override.

The non-required argument attributes have default values but can also be specified by keyword arguments to the new object.

Child classes of `GalfitParameter` are written to accommodate other `GALFIT` features (or anything which is contained in the feedme file) such as header parameters and multi-valued parameters, i.e. position, fourier modes, etc. These multi-valued parameters are generally described by purpose-built classes to accommodate various differences in outputting to string and other attributes of convenience but all inherit the attributes and methods of a parent `MultiParameter` class which also inherits the attributes and methods of the `GalfitParameter`.

The non-inherited attributes of the `MultiParameter` class are:

- `x`
- `y`
- `fix_x`
- `fix_y`

The `MultiParameter` uses the built-in `NamedTuple` python module to enable flexible value and fix value retrieval. For example, the x and y values of the `position` parameter (itself otherwise described by a unique child class of `MultiParameter`) can be retrieved by using any of the following syntax:

```
multiparameter_obj.value

(multiparameter_obj.value[0], multiparameter_obj.value[1])

(multiparameter_obj.x, multiparameter_obj.y)
```

This same syntax applies to the `fix` attribute.

Several other classes are defined in the `Parameters` submodule which inherit the attributes and methods from `GalfitParameter` and `MultiParameter`. In the `Parameters` submodule, we also include functions which create python dictionaries, corresponding to `GALFIT` functions (i.e. Sérsic, power, etc.), with keys as the parameter names and values as the instantiated parameter objects loaded with "defaults" for the various attributes. The `Components` submodule takes significant advantage of these functions. Furthermore, the dictionary structure is fundamental to the operation of `Components` and enables increased flexibility in a similar manner to that of the `NamedTuple` usage discussed above.

B.1.2 `GalfitComponent`

The components, stored in the `Components` submodule and based on a parent `GalfitComponent` class, are abstracted `GALFIT` component functions, with Sérsic, Power, Fourier, and Sky components currently available. The header, although not a `GALFIT` *function* is also treated as a component being a child class of `GalfitComponent`. The `GalfitComponent` class is written to generically handle the parameters associated with a component as well as meta information about the components. It is also designed to provide flexibility by the dictionary which stores parameters and to handle I/O at the level appropriate for components (see the `Container` and `FitsHandlers` sections below). The required arguments, attributes, methods, and overrides of this class are:

REQUIRED ARGUMENTS

· `component_type` – the type of component being created. String.

ATTRIBUTES

· `component_type` – the `component_type` from REQUIRED ARGUMENTS.

· `parameters` – stores the parameters that inform the component/function as a dictionary. This dictionary can be accessed as key/value or more directly as an attribute of the class, i.e.

```
sersic_obj.magnitude.value
```

or

`sersic_obj.parameters[‘magnitude’].value`

- `component_name` – stores the name of the component as a string.

- `component_number` – stores the component number. This attribute is important for functions which modify other components, i.e. Power functions, and allows us to retain the correct function pairings should an issue arise.

- `param_prefix` – stores the prefix used in generating the `GALFIT` feedme which corresponds to the chosen `GALFIT` component/function.

- `_start_dict/end_dict` – internal use attribute. Stores the strings which indicate the start and end of the dictionary ingested from a `GALFIT` output FITS file header.

- `_start_text/end_text` – internal use attribute. Stores the strings which indicate the start and end of the dictionary ingested from a `GALFIT` input feedme file.

- `_section_sep` – internal use attribute. Stores the string which is used as the generic separator in the `GALFIT` feedme.

METHODS

- `update_parameter_dict_with_dict/list` – used for updating the parameter attribute (dictionary) with another dictionary/list. Importantly, this function does allow updating of the parameter attribute with parameters which do not already exist in the component.

- `update_from_log` – used for updating a component from the log output to STDOUT by `GALFIT` as a result of the fitting process.

· `to/from_pandas` – used for reading in from or outputting the component to a pandas dataframe (The Pandas Development Team, 2020). Assumes and/or creates column headers in the format: `[parameter_name]_[component_type]_[component_number]`

· `from_file` – used to handle and pre-process input FITS or feedme files for ingestion into the component object.

· `from_file_helper_dict/list` – used for extracting component information from a dictionary or a list after the pre-processing of a FITS header or feedme file that occurs in `from_file`.

· `to_file` – is used to print a component to a file. Uses the component's string override to achieve this and hence achieves generalizability. If other components are specified as positional arguments, they are printed in addition to the component calling the method to file.

OVERRIDES

· `__str__` – overrides `str()`. Is used for outputting the entire component as a string formatted in the style of the `GALFIT` feedme.

· `__repr__` – overrides `repr()`. Is used for outputting the entire component as a string, in the same format as __str__, but containing solely the `value` of the parameters inside.

The other non-internal attributes have default non-values (i.e. 0, empty string, or string with a space) but can also be specified by keyword arguments to the new object.

Child classes of `GalfitComponent` are written to accommodate the specific `GALFIT` components/functions referenced above. All but the `Header` class require `component_number` as a positional argument upon creation. These child classes take advantage of the parameter loading functions described in the `Parameters` section above (B.1.1) to fill the object upon creation. These defaults can still be overwritten during creation through keyword arguments. The `Fourier` and `Header` classes diverge the most from the other child classes in the specific formatting and conventions necessary for both.

The `Fourier` class in particular has several special attributes and methods which we detail here in anticipation of this scheme forming the basis of additional components to be added in the future, i.e. bending modes and truncation). Besides the `component_number` and other inherited keyword attributes, the `Fourier` class can also take a dictionary as a key word argument (with key n) which provides the desired fourier modes, amplitudes, and phase angles in the form `{[fourier mode]:[amplitude], [phase angle])}`. The additional attributes and methods are:

ATTRIBUTES

· `amplitudes/phase_angles` – stores the amplitude/phase angle `value`s in list format. Note: these attributes alone **do not** contain the corresponding *mode* information but rely on the user to have this information upon using.

METHODS

· `sort_parameters` – used for sorting the fourier modes via `parameters` into numerical order given potential issues which may arise from keyword argument input or default overriding.

· `include_fn` – used for creating/updating fourier mode `parameter`s from a dictionary.

As is the case with the `Parameters` submodule, we provide a `load_all_components` function in `Components` which creates a dictionary containing a default set of all available components. The `ComponentContainer` and, to a lesser extent, `FitsHandler` submodules take significant advantage of this function. It is (again) fundamental to the flexibility of their implementation.

B.1.3 `ComponentContainer`

The containers, stored in the `Containers` submodule and based on a parent `ComponentContainer` class, are developed to generically hold multiple components as attributes of the container. If the user desires, upon creation of a container, they can specify the names of physical structures coinciding with the specific components via keyword arguments in order to abstract the structures into the container object. This again takes advantage of python's dictionary key/value structure. If the names are not specified or the default structure is not used, the keys default to `[component_name]_[iterator]`. Its main attributes, methods, and overrides are as follows:

ATTRIBUTES

· `components` – abstracts the components being stored in the container into a dictionary.

METHODS

· `to_tuple/list` – used to create a tuple/list out of the components stored in the container.

· `to/from_pandas` – used to combine the dataframes output by or ingest dataframes into the components classes into/from one unified dataframe.

OVERRIDES

· `__str__` – overrides `str()`. Is used for joining all component strings into a single string.

· `__repr__` – overrides `repr()`. Is used for joining all component `repr` strings into a single string.

Although the `ComponentContainer` contains some utility in the basic transformations described above, its two child classes `FeedmeContainer` and `OutputContainer`, are designed to perform I/O to/from STDOUT or file and thus enjoy the most usage in the pipeline.

The `FeedmeContainer` is designed to abstract a `GALFIT` feedme file. It has a single additional attribute, `path_to_feedme` which stores the path to the feedme file used to generate the object or otherwise the path to write the feedme to. A keyword argument `load_default`, which defaults to true, can be used to specify loading the default (as used in this study: header, bulge, disk, disk for arms, arms, fourier, and sky) component set. Setting this keyword argument to true will not override any components passed in as arguments during the object creation. Care must be taken when using this option with components fed in during object creation as misnaming will cause extra components to be stored in the container. The `FeedmeContainer` also contains many additional methods and overrides which allow it to handle file I/O. These are:

· `sort_components` – used to sort components in their host dictionary in numerical order by their `component_number`. Uses the hierarchy: Sérsic, power, fourier, bending, truncation to determine combination order.

· `to_file` – used to print the components as a full feedme to a file. Uses the components' string overrides to achieve this and hence achieves generalizability. If components are specified as positional arguments, this method is identical to that found in the `GalfitComponent` class.

· `from_file` – used to ingest a feedme in its entirety and parse the subsequent components into the container. This method employs several inner functions which we detail below.

`from_file` INNER FUNCTIONS

- `check_matches` – used to validate ingested components against components already existing in the container. This function assigns new key values to these components, in the form of `[component_name_iterator]`, before adding them to the `components` attribute dictionary. This function is written with compatibility for both the input from FITS headers and text.

- `from_fits` – used to parse FITS headers as output by `GALFIT` which contain the model parameterization. Once parsed, the function sends the appropriate file chunks to the individual component's `from_file_helper` methods. Note that FITS headers have a limited character width so some of the `GALFIT` header information found in the FITS header may be truncated—typically file path information as is found in header items A – F.We elect to leave these potentially incomplete header entries to their default values as handled by the `GalfitHeader` class from `Components`.

- `from_text` – used to parse `GALFIT` text feedme files. Once parsed, the function sends the appropriate file chunks to the individual component's `from_file_helper` methods.

OVERRIDES

- `__str__` – overrides `str()`. Is used for joining all component strings into a single string in the style of the `GALFIT` feedme. As compared to the method in the `ComponentContainer`, this is designed to output components which modify others directly next to the modified component, i.e. power and Sérsic.

The primary utility of the FeedmeContainer in this pipeline is to carry information from the conversion from SpArcFiRe to GALFIT and to hold the feedme information in the abstracted FITS file output described in the FitsHandlers section below.

The OutputContainer is the main driver of the multi-step fitting scheme (Section 4.3) as it takes as input and handles GALFIT's output to STDOUT/STDERR, and populates or updates the component attributes accordingly. The ability to handle GALFIT output and success/failure dynamically is a necessary feature for efficient distributed computing in our implementation, Section 4.5. It is a child class of the FeedmeContainer—its additional attributes and methods are as follows:

ATTRIBUTES

- success – stores a binary value indicating whether GALFIT successfully completed its decomposition. Defaults to false.
- galfit_out/err_text – used to store the output texts from stdout/stderr as extracted from the python subprocess.CompletedProcess object which results from running GALFIT as a subprocess in python. If the option store_text is specified, the text is stored. If not, default text indicates the need to use the option store_text if storing output is desired.

METHODS

- check_success – local function. Is used to parse output and determine whether GALFIT succeeded or not. Updates the success attribute with the results of this operation.
- update_components – local function. Updates components from the information output to STDOUT by GALFIT given success is true using the components' update_from_log method. Currently requires the component list to be specified ahead

of time (or the default to be used) due to its inheritance of the `FeedmeContainer` class. This may change in a future release.

- `__str__` – overrides str(). When called, prints out the output from STDOUT stored in `galfit_out_text`.

B.1.4 `FitsHandlers`

The fits handlers, stored in the `FitsHandlers` submodule, generically abstract FITS files of the kind used and output by `GALFIT`, with most of the base I/O utility being derived from `Astropy`'s FITS handling package `io.FITS`. This submodule uses a `HDU` class as its foundation which abstracts the name of an HDU, its header information, and the image data associated with the HDU and stores these as attributes. On top of this, the parent FITS handling class we employ is `FitsFile` which is designed to generically store HDU data but which contains some default choices convenient for use with `GALFIT`. Its main attributes and methods are:

REQUIRED ARGUMENTS

- `filepath` – the filepath to the FITS file being ingested.

ATTRIBUTES

- `filepath` – the filepath from REQUIRED ARGUMENTS.

- `gname` – the galaxy name of the file being ingested. This attribute can be set as a key word argument during object creation but defaults to extracting the filename from `filepath`.

- `names` – a list of the names of the images being ingested to be used as the key to the dictionary holding the ingested `HDU` objects. This attribute can be set as a key word argument during object creation but defaults to `['observation']` for convenience.

- `num_imgs` – the number of FITS file HDUs present in the file being ingested. This *must* equal the length of `names` or an error will be raised.

- `all_hdu` – a dictionary containing the names (keys) and populated `HDU` objects from the ingested FITS files (values).

- `observation` – this attribute refers to the "observation", if present in the dictionary, and is offered for convenience. Its inclusion here is based on the assumption that the `FitsFile` class will be predominantly used to handle observation files used as input for `GALFIT`.

  METHODS

- `close` – a destructor used to fully close FITS files according to `Astropy` convention.[2]

- `to_png` – a method to generically generate and tile images using `fitspng` (F. Hroch, 2019) and `ImageMagick` (The ImageMagick Development Team, 2020). If the `FitsFile` object only contains a single image, this method will output a resized version of the image with dimensions 175x175. Sets several attributes which store the filepath to the generated pngs—we do not describe them here as they are provided merely for convenience but do not play a crucial role in our pipeline.

  OVERRIDES

[2]See **Warning** in https://docs.astropy.org/en/stable/io/fits/#working-with-large-files

- `sub` – overrides built-in subtract. Can be used to compare single or multiple image blocks given that the two `FitsFile` objects contain the same number of `HDU` objects. If not, the code will raise an error.

The primary fits handler class we employ is an `OutputFits` class, child of `FitsFile`, which conveniently holds the header and data of each image block from `GALFIT`'s output: observation, model, and residual, and generates a feedme via the `FeedmeContainer` class from the model header. Importantly, the `OutputFits` class has the capability to calculate our measures of the residual (Section 5.1) and to update the output FITS file header with these values to be retrieved later. We take advantage of the header updating feature to collate our results during the analysis portion of our pipeline, see Figure 4.1. Its main attributes and methods—beyond those it inherits from `FitsFile`—are:

REQUIRED ARGUMENTS

- `filepath` – the filepath to the FITS file being ingested.

ATTRIBUTES

- `model` – this attribute abstracts the model HDU generated by `GALFIT`.
- `residual` – this attribute abstracts the residual HDU generated by `GALFIT`.
- `header` – this attribute abstracts the header of the model and is populated on object instantiation. The attribute is offered for convenience on the assumption that a user requesting a header from an `OutputFits` object is likely referring to the header of the model.
- `data` – this attribute abstracts the image data of the model and is populated on object instantiation. The attribute is offered for convenience on the assumption that a user

requesting data from an `OutputFits` object is likely referring to the image data of the model.

- `feedme` – this attribute abstracts the feedme into a `FeedmeContainer`. It is populated on object instantiation as extracted from the model header.

- `bulge_mask` – this attribute abstracts the bulge mask that `SpArcFiRe` uses to process the galaxy. On object creation, it is created as a ones-array (no masking) with shape equivalent to that of the model image data array. This attribute is used in our residual analysis and can be updated by the `generate_bulge_mask` method.

METHODS

- `generate_bulge_mask` – a method to generate a bulge mask from `SpArcFiRe`'s analysis. Stores the resultant bulge mask in the `bulge_mask` attribute. Returns the bulge mask data array.

- `generate_masked_residual` – a method to calculate our metric for the residual (Section 5.1). Incorporates the star mask generated by SourceExtractor (Bertin & Arnouts, 1996) and the bulge mask if desired. Sets many attributes in the object related to this analysis. Returns the `masked_residual_normalized`.

ATTRIBUTES SET BY `generate_masked_residual` METHOD

- `masked_residual` – abstracts the data array representing the residual masked by the star mask—this is referred to as the `crop_mask` in the code to indicate that the star mask used is cropped to the coordinates specified during the `GALFIT` decomposition.

- `kstest` – abstracts the results from running the Kolmogorov–Smirnov (KS) test as implemented from the `SciPy.stats` package (Virtanen et al., 2020). Stores the object generated by the `SciPy.stats` kstest method. Can be written to the FITS header by the option `update_fits_header`.

- `norm_observation` – abstracts the 2-norm of the observation image data array.
- `norm_model` – abstracts the 2-norm of the model image data array.
- `norm_residual` – abstracts the 2-norm of the residual image data array.
- `masked_residual_normalized` – abstracts the masked residual data array normalized by the minimum between the observation and model 2-norms (via `SciPy.linalg.norm` Virtanen et al., 2020).
- `nmr` – abstracts the 2-norm of the `masked_residual_normalized`. The abbreviation refers to "norm of the masked residual", abbreviated so to avoid confusion with the `masked_residual_normalized`. Can be written to the FITS header by the option `update_fits_header`.

B.2 Regression Testing

Regression tests have been incorporated into the `GalfitModule` in the `RegTest` submodule. The regression test can be run by calling the script `RegTest.py` from this submodule. The test operates by first creating a `TestOutput` directory, where all subsequent files will be dropped, and calling each class module in a subprocess shell so as to run their individual unit tests and capture their output. This output is compared to known output in the `TestData` directory and any errors are reported to a text file entitled `OutputError.txt`. Furthermore, the code runs `GALFIT` on 11 observations, checking the input files for consistency with the 'correct' files and the output files for having been generated. It is not possible, at this time, to compare the image data in the `GALFIT` output files directly as different computing set-ups have been shown to produce different results, some of which cannot be avoided by setting a tolerance threshold. We anticipate that it may be possible to circumvent this issue by using GitHub actions and a docker environment and intend to explore this option in the future. The results of the reg test are summarized in `STDOUT` and output to the `UnitTestStdOutput.txt` and `OutputError.txt` files. If verbosity is enabled, all information written to these files along with the `STDOUT` from `GALFIT` (which is otherwise captured) is output to `STDOUT`.

B.3 Additional Utilities

The `GalfitModule` contains additional utilities in the `Utilities` submodule that aid in the operation of the pipeline or that can be/are used in conjunction with the pipeline to produce and assess results. In several cases, these utilities augment the processes built into the main scripts and are demarcated as "utilities" to indicate that they can be run as standalone or that it may be appropriate to use these scripts outside the scope of the main pipeline.

We summarize here several utilities which do not require in-depth explanation but which a user may find useful or otherwise convenient:

- `extract_model_from_galfit_output.py` – a script that extracts solely the model and header information from the FITS files output by `GALFIT` into a single image block FITS file.

- `generate_model_only_fits.py` – a script that can be used to turn off `GALFIT`'s optimization stage and produce models directly from the . This utility is most desirable in the context of refining the S2G procedure and in some aspects of the analysis.

- `generate_png.py` – a script that can be used to generate observation, model, residual tiled PNGs for models output by `GALFIT`. This is essentially a convenience wrapper for the `to_png` method of the `FitsFile` class and is most often used for when models cannot be generated due to compatibility issues with `fitspng` or `ImageMagick` when the pipeline is run.

- `label_images.py` – a script that allows a user to label output result images (converted to jpg) by sorting them into pre-specified directories. All by-eye labeling in Chapter 6 has been done using this script.

- `symlink_data.py` – a script that prepares a directory with the symlinked data necessary to run the pipeline. The data that are symlinked are the input observations, placed in the default input directory `sparcfire-in`, and the `csv`s from a `SpArcFiRe` data run, placed in individual galaxy folders in the default output directory `sparcfire-out`, all of which will be created if not already done so.

- `no_opt` – a subdirectory of `Utilities` which contains scripts to generate and tile `GALFIT` models which don't undergo the optimization process. This is used in Section 6.3 to demonstrate our translation `SpArcFiRe` output to `GALFIT` parameterization as a good initial guess for our modeling procedure.

Appendix C

Deriving the Pitch Angle Measurement from `GALFIT`

We present the full calculation used to determine the pitch angle via `GALFIT`. Expanding on the functional form of Equation (2.6), and following a condensed version of that presented in (P, 2010),

$$A = \frac{2 \times \mathrm{CDEF}}{|\theta_{\mathrm{out}}| + \mathrm{CDEF}} - 1.00001 \tag{C.1}$$

$$B = \Big[2 - \tanh^{-1}(A)\Big]\left(\frac{r_{\mathrm{out}}}{r_{\mathrm{out}} - r_{in}}\right) \tag{C.2}$$

$$\tanh\big(r_{in}, r_{\mathrm{out}}, \theta_{incl}, \theta_{pa}^{sky}; r\big) \equiv$$

$$0.5 \times \left\{ \tanh\left[B\left(\frac{r}{r_{\mathrm{out}}} - 1\right) + 2\right] + 1\right\} \tag{C.3}$$

We use this expanded form to find the derivative of Equation (2.6)

$$\theta_1(r) = \frac{1}{2}\left\{\tanh\left[B\left(\frac{r}{r_{\text{out}}} - 1\right) + 2\right] + 1\right\} \tag{C.4}$$

$$\theta_2(r) = \theta_{\text{out}}\left[\frac{1}{2}\left(\frac{r}{r_{\text{out}}} + 1\right)\right]^{\alpha} \tag{C.5}$$

$$\theta(r) = \theta_1(r) \cdot \theta_2(r)$$

$$\therefore \frac{\partial \theta}{\partial r} = \theta_1(r)\frac{\partial \theta_2}{\partial r} \cdot \theta_2(r)\frac{\partial \theta_1}{\partial r} \tag{C.6}$$

where

$$\frac{\partial \theta_1}{\partial r} = \frac{B}{2r_{\text{out}}\cosh^2\left(2 - B + B\frac{r}{r_{\text{out}}}\right)}$$

$$\frac{\partial \theta_2}{\partial r} = \frac{\alpha\theta_{\text{out}}}{2^{\alpha}}\frac{\left(1 + \frac{r}{r_{\text{out}}}\right)^{\alpha}}{r_{\text{out}} + r}$$

As the code is implemented in `Python`, we catch Overflow errors due to division by 0.

Appendix D

Additional `GALFIT` Results Quantiled

We present the image, model, residual results from the NC2 and NC3 results from the 1000 and 29k data sets, selected by residual score quantiles, in intervals of 20%. The images from the 1000 data set are those selected by-eye. The images from the 29k data set are selected from the results which are deemed successful by our combined success metric.

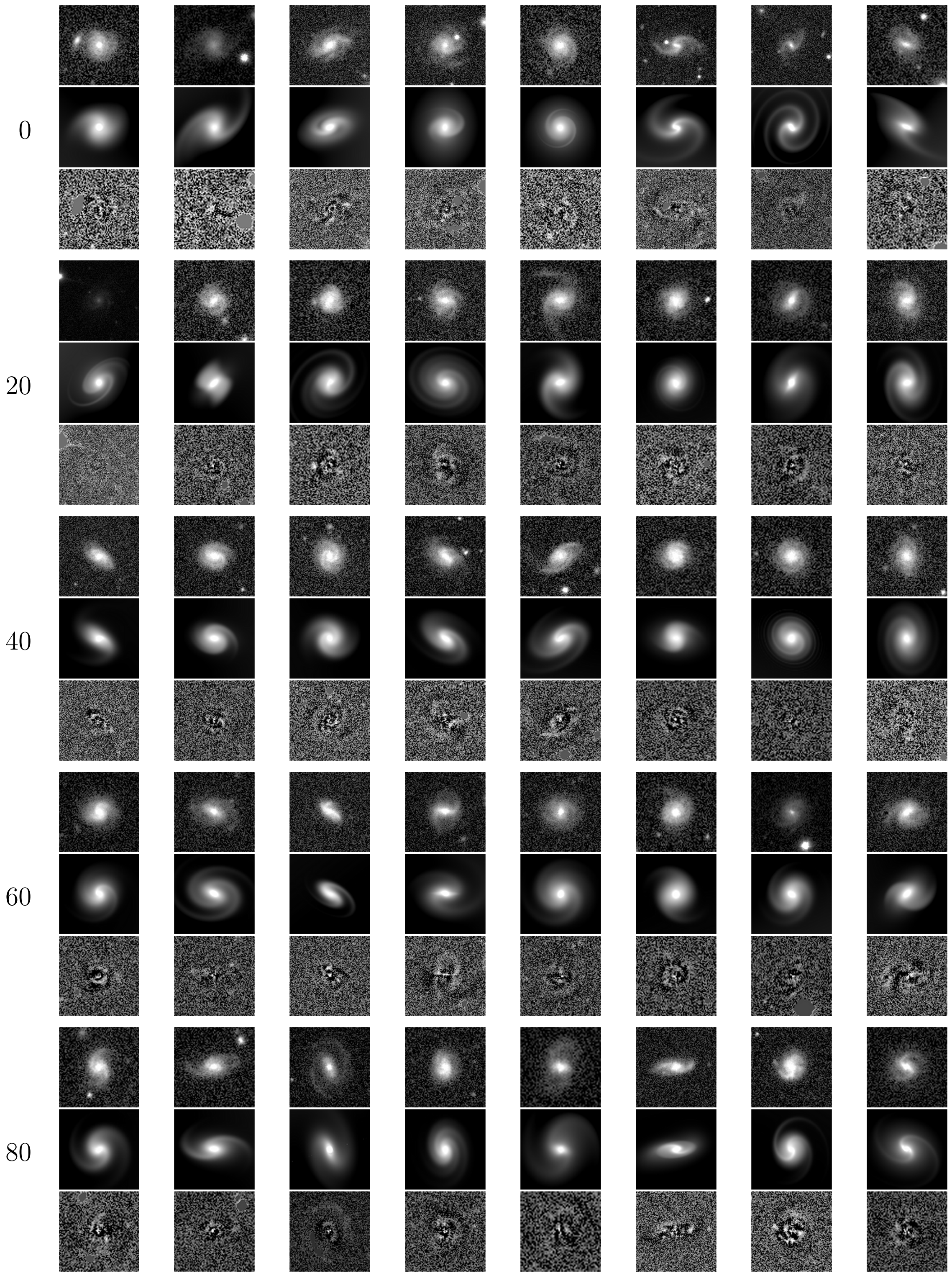


Figure D.1: Models from the by-eye successful sample of 1000 galaxies from NC2 selected by quantile, with representations at every 20%, of the residual metric. Each image is observation, model, residual, vertically oriented.

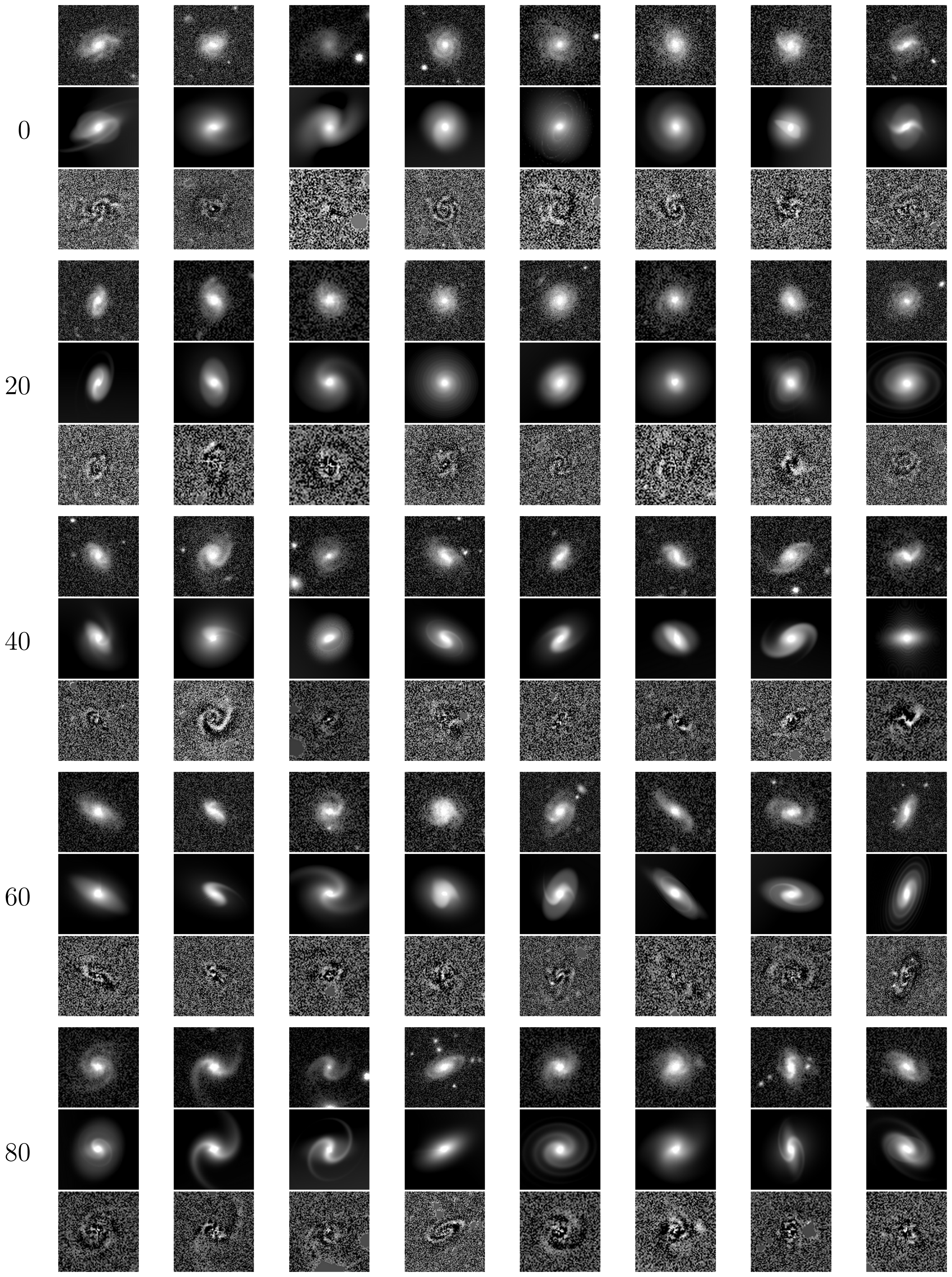


Figure D.2: Models from the by-eye successful sample of 1000 galaxies from NC3 selected by quantile, with representations at every 20%, of the residual metric. Each image is observation, model, residual, vertically oriented.

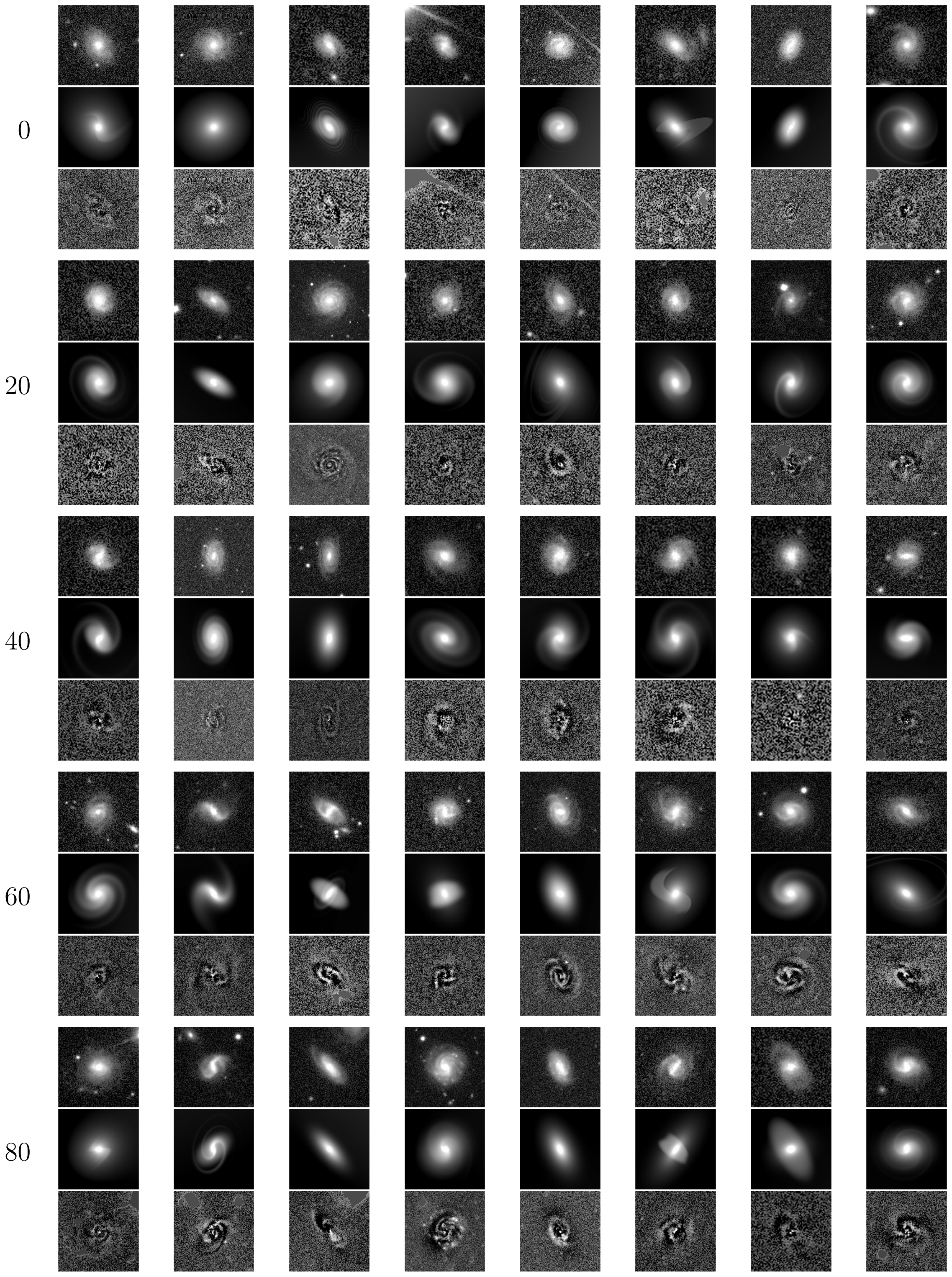


Figure D.3: Models from the by-metric successful sample of 29k galaxies from NC2 selected by quantile, with representations at every 20%, of the residual metric. Each image is observation, model, residual, vertically oriented.

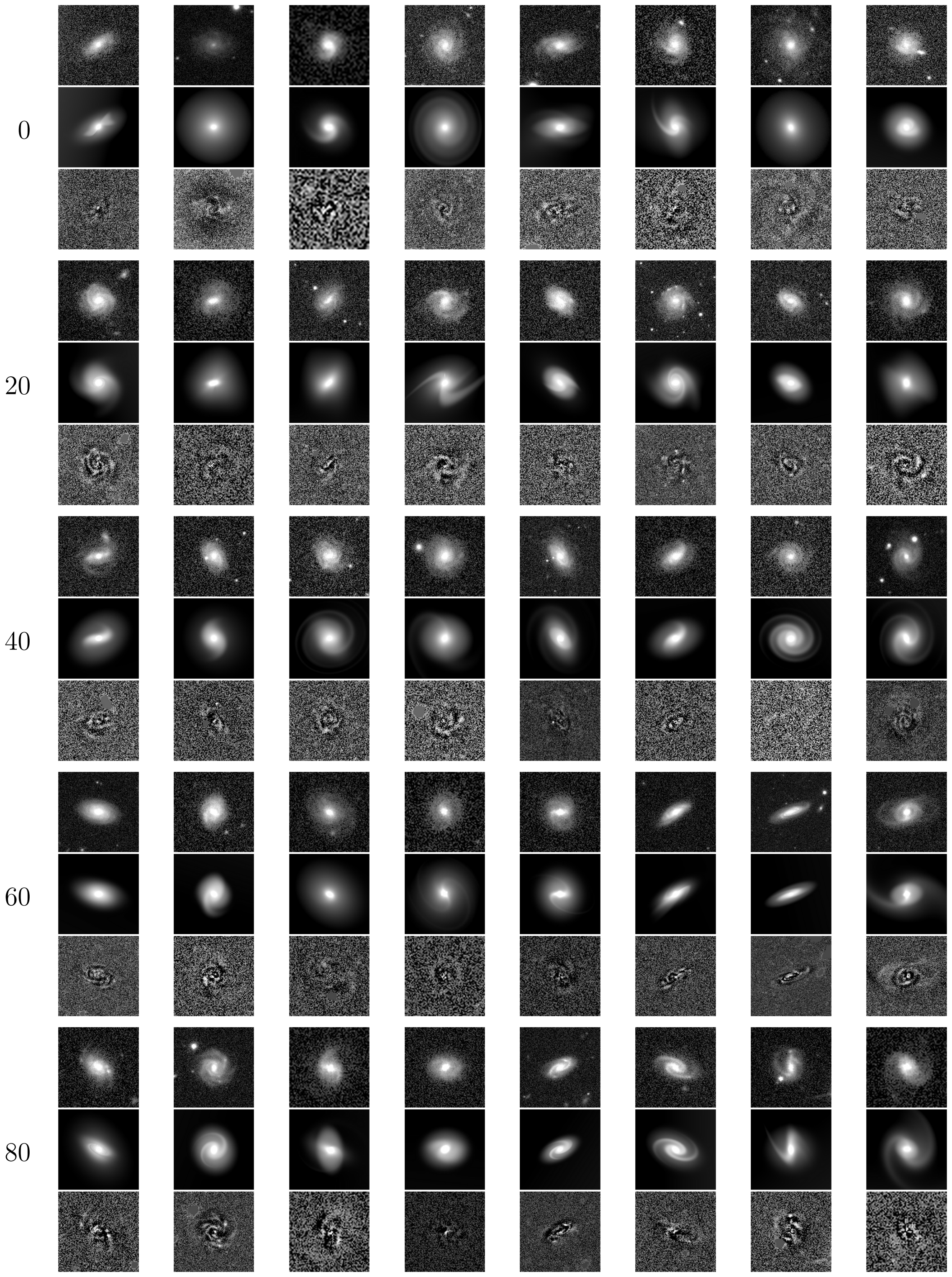


Figure D.4: Models from the by-metric successful sample of 29k galaxies from NC3 selected by quantile, with representations at every 20%, of the residual metric. Each image is observation, model, residual, vertically oriented.

Appendix E

Pairwise Parameter Scatter Plots

We present pairwise scatter plots for several of the parameters discussed in Sections 6.4.2 and 6.5.2 for the NC2 + NC3 results for the 1000 and 29k sample sets. Note that the axes limits on these results are artificially reduced in order to highlight trends in high density regions of the plots.

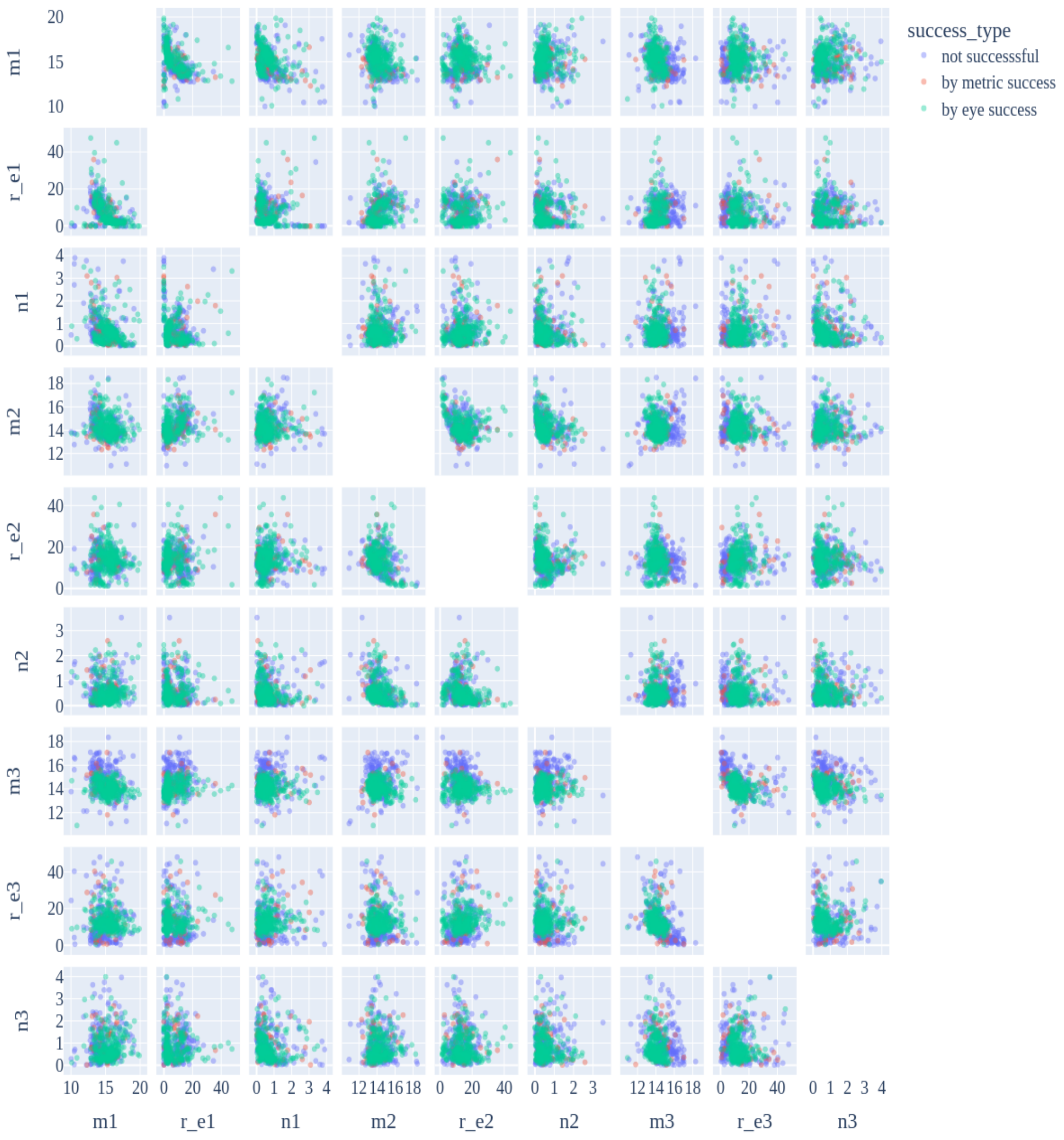


Figure E.1: Pairwise scatter plots for the Sérsic: magnitude, effective radius, and Sérsic index for components 1, 2, and 3 (if applicable) of the 1000 galaxy sample set. The markers in each scatter plot indicate the success level with color priority decreasing from by-eye successful to successful to not successful, i.e., by-eye successful models are always colored green regardless of their by metric determination. This implies that the 'by metric success' markers in these plots are false positives and all 'not successful' markers are true negatives given that all 1000 galaxies have been labeled by-eye. False negatives cannot be seen.

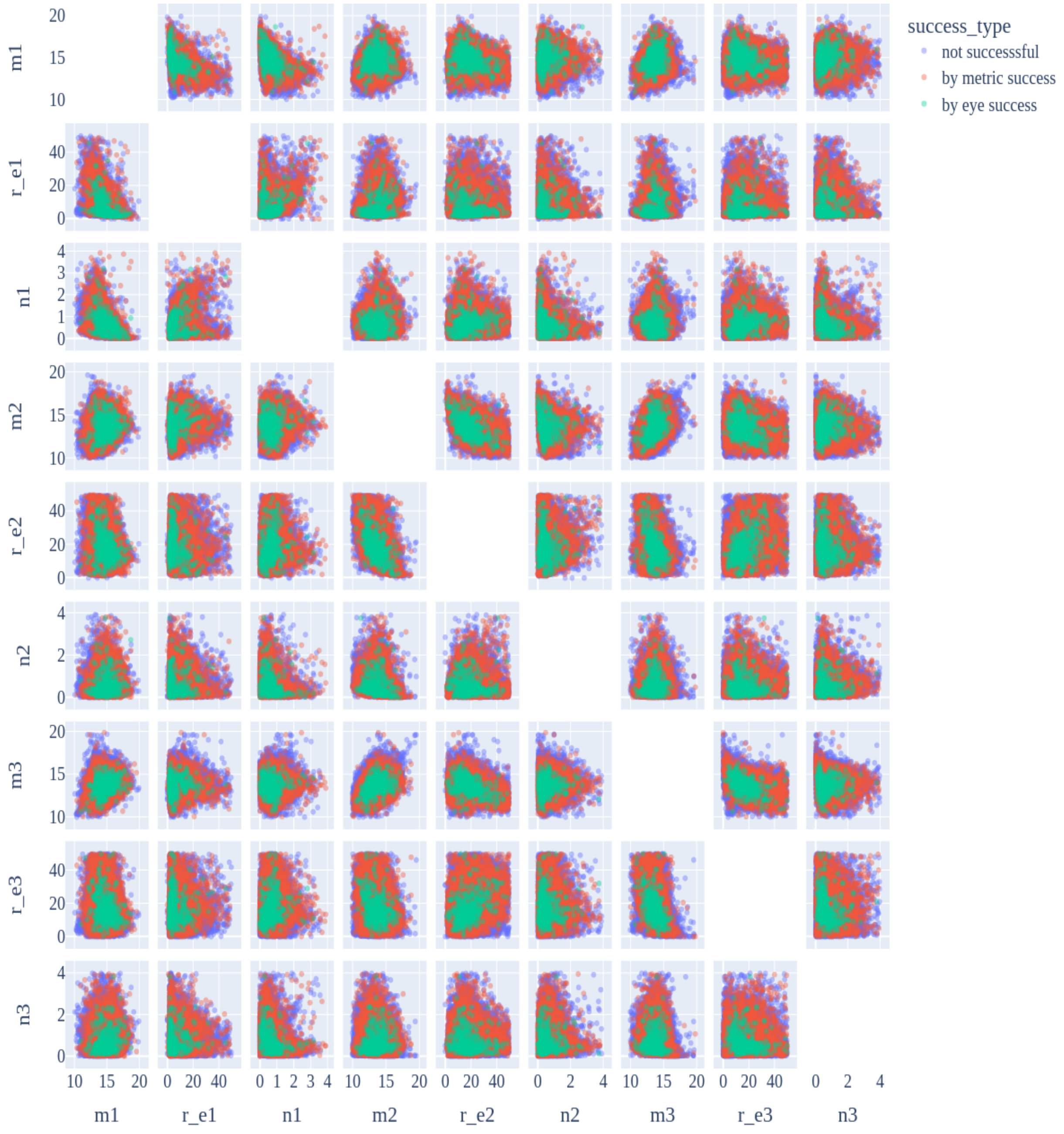
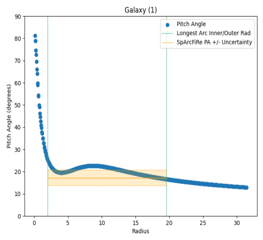


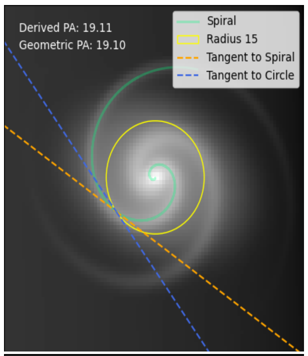
Figure E.2: Pairwise scatter plots for the Sérsic: magnitude, effective radius, and Sérsic index for components 1, 2, and 3 (if applicable) of the 29k galaxy sample set. The markers in each scatter plot indicate the success level with color priority decreasing from by-eye successful to successful to not successful, i.e., by-eye successful models are always colored green regardless of their by metric determination.

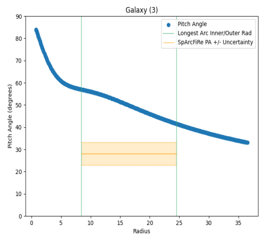
Appendix F

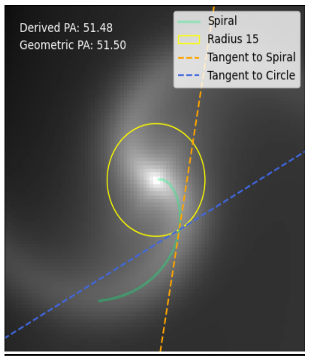
Additional Pitch Angle Plots

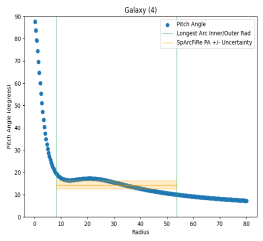
We present the full set of scatter plots and spiral arm overlay (validation) plots for the sample of 14 NC3 galaxies.

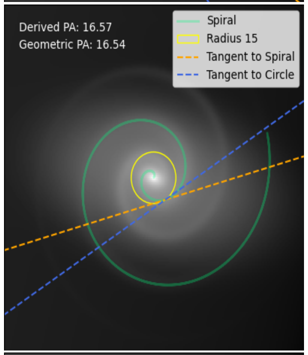

Galaxy (1)
Pitch Angle
Longest Arc Inner/Outer Rad
SpArcFiRe PA +/- Uncertainty
Pitch Angle (degrees)
Radius

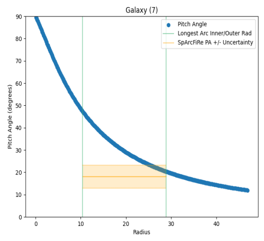

Derived PA: 19.11
Geometric PA: 19.10
Spiral
Radius 15
Tangent to Spiral
Tangent to Circle

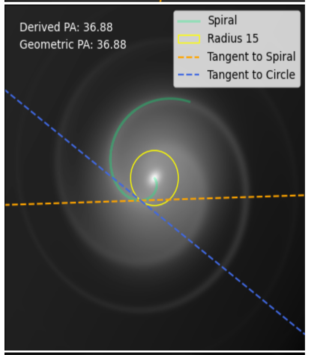

Galaxy (3)
Pitch Angle
Longest Arc Inner/Outer Rad
SpArcFiRe PA +/- Uncertainty
Pitch Angle (degrees)
Radius

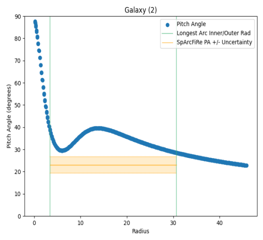

Derived PA: 51.48
Geometric PA: 51.50
Spiral
Radius 15
Tangent to Spiral
Tangent to Circle

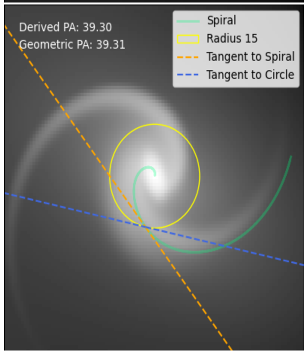

Galaxy (4)
Pitch Angle
Longest Arc Inner/Outer Rad
SpArcFiRe PA +/- Uncertainty
Pitch Angle (degrees)
Radius

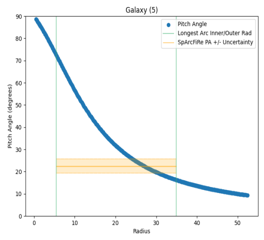

Derived PA: 16.57
Geometric PA: 16.54
Spiral
Radius 15
Tangent to Spiral
Tangent to Circle

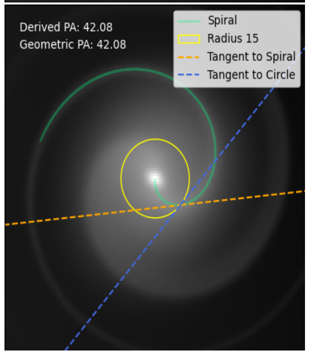

Galaxy (7)
Pitch Angle
Longest Arc Inner/Outer Rad
SpArcFiRe PA +/- Uncertainty
Pitch Angle (degrees)
Radius

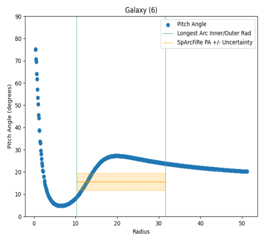

Derived PA: 36.88
Geometric PA: 36.88
Spiral
Radius 15
Tangent to Spiral
Tangent to Circle

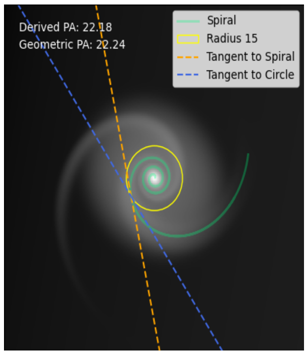

Galaxy (2)
Pitch Angle
Longest Arc Inner/Outer Rad
SpArcFiRe PA +/- Uncertainty
Pitch Angle (degrees)
Radius


Derived PA: 39.30
Geometric PA: 39.31
Spiral
Radius 15
Tangent to Spiral
Tangent to Circle


Galaxy (5)
Pitch Angle
Longest Arc Inner/Outer Rad
SpArcFiRe PA +/- Uncertainty
Pitch Angle (degrees)
Radius


Derived PA: 42.08
Geometric PA: 42.08
Spiral
Radius 15
Tangent to Spiral
Tangent to Circle


Galaxy (6)
Pitch Angle
Longest Arc Inner/Outer Rad
SpArcFiRe PA +/- Uncertainty
Pitch Angle (degrees)
Radius


Derived PA: 22.18
Geometric PA: 22.24
Spiral
Radius 15
Tangent to Spiral
Tangent to Circle

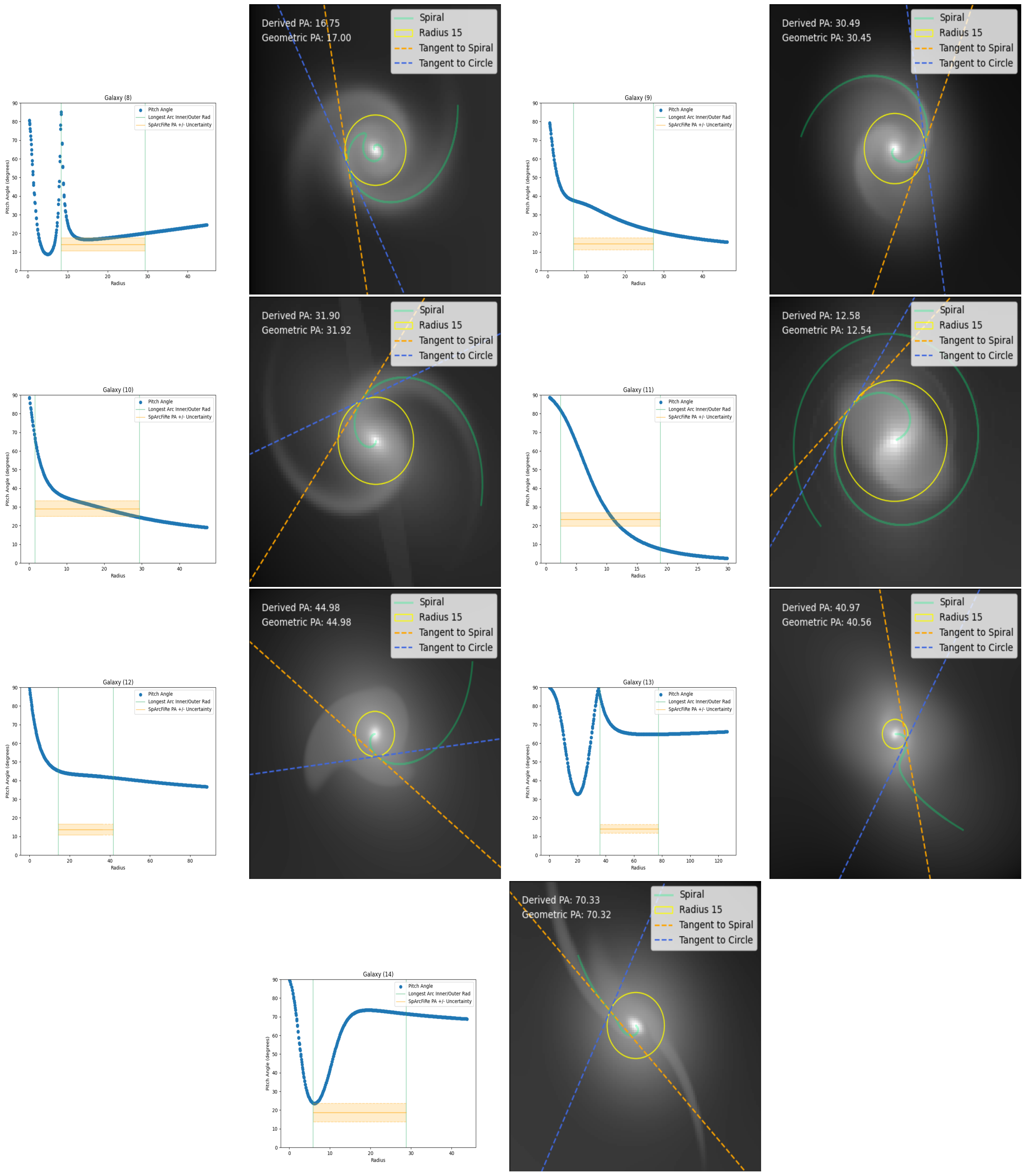


Figure F.1: Pitch angle scatter and overlay plots for the sample of 14 NC3 galaxies.