Publications of the Astronomical Society of Australia (2020), 1–18

CAMBRIDGE
UNIVERSITY PRESS

ARTICLE TYPE

Report on the Kinematic Modelling Pipeline for WALLABY – Oct. 2023

N. Deg, K. Spekkens

Department of Physics, Engineering Physics, and Astronomy, Queen's University, Kingston ON K7L 3N6, Canada
Author for correspondence: N. Deg, Email: nathan.j.deg@gmail.com.

Abstract

This is a general report on the status of the kinematic modelling pipeline for WALLABY TWG5.

Keywords: keyword entry 1, keyword entry 2, keyword entry 3

1. Introduction

The WALLABY Technical Working Group 5 (TWG5) is aimed at generating kinematic models for as many spatially-resolved WALLABY detections as possible. They have already produced the WALLABY Kinematic Analysis Proto-Pipeline (WKAPP; Deg et al. 2022) for use in the pilot survey. WKAPP uses a combination of the Fully Automated TiRiFiC code (FAT; Kamphuis et al. 2015) and the 3D-Based Analysis of Rotating Objects From Line Observations (3DBarolo; Di Teodoro & Fraternali 2015) to produce its kinematic models. The proto-pipeline has been fairly successful, but it has key limitations that make it unsuited to the full survey: it relies on human examination to accept/reject specific models, it is relatively slow and not fully automated, it is not optimized for WALLABY-like detections, and, most importantly, the uncertainties returned on best fitting parameters are not statistically meaningful. For these reasons, TWG5 has been developing a new kinematic pipeline that is meant to address these issues.

This report is meant to describe the first working version of this full pipeline for TWG5 team members as well as for the broader WALLABY team. As there are a few remaining details to decide upon before fully implementing the pipeline within the WALLABY team, the report is organised in a non-standard way. Sec. 2 briefly describes the overall pipeline structure. Sec. 3 describes the various products produced by the current version of the pipeline. These products have not been finalized, so feedback on them would be appreciated. Sec. 4 presents the set of tests that have been used to validate the pipeline. Sec. 5 explains the fitting process for a single galaxy in detail, while Sec. 6 describes how the uncertainties are generated using a novel bootstrapping technique. Sec. 7 explains how these pieces are combined into the full pipeline. Finally Sec. 8 discusses limitations in the pipeline as well as some possible improvements that can be considered in future versions of the code.

2. General Pipeline Structure

The working title of the pipeline has been the WALLABY Resolved Kinematic Pipeline (WRKP). However, as the code is relatively general and can be applied to other surveys, this working name should be discontinued. Instead, the proposed pipeline name is the 3D-Kinematic Data aNalysis Algorithim for Surveys (3KIDNAS, pronounced as *echidnas*; naming credit to J. Delhaize).

The design of 3KIDNAS was constrained and guided by the priorities of WALLABY. As WALLABY is an untargetted survey, the majority of the detections will be low resolution and low S/N. Most of the detections will be relatively nearby, and estimates on the total number of detections are on the order of $2-3\times10^5$. Thus, the power of WALLABY will lie in the statistics. Taken together, 3KIDNAS must provide robust models with statistically meaningful uncertainties for as many galaxies as possible. To fully leverage the large number of detections, all galaxies must be modelled in the same way, and human intervention must be minimized. This will allow for a more straightforward selection function and comparison with simulations. Interesting and complicated models that explore warps, radial flows, and other phenomena will only be possible for a very small subset of more highly resolved and higher S/N galaxies. At lower resolutions and S/N, the data will only be able to constrain relatively simple 'flat' disk models where the geometry is fixed across the galaxy.

Given these constraints, 3KIDNAS uses a 3D tilted ring (TR) modelling algorithm to generate its models. The core of the fitting algorithm bears a number of similarities to 3DBarolo and FAT but it is fully independent of these existing 3D TR modelling codes. For the sake of modularity, the pipeline is broken up into a number of distinct Fortran programs that feed into each other through python scripts.

The core fitting code, currently called SingleGalaxyFitter takes a cube and associated mask and uses a downhill simplex algorithm (Press et al., 1992) to find the best fitting model. The fitting itself uses the unmasked cube, but, as with FAT and 3DBarolo initial estimates of the model are required beforehand, which is the only place where the mask is used.

In order to obtain uncertainties, we have developed a 3D 'flipping bootstrap' resampler. This has been implemented in Fortran as the BootstrapSampler code. It takes the original data cube, a model cube along with the model's geometric center and position angle, and produces a mock data cube where the residuals have been scrambled about the axes of symmetry for a flat rotating disk in such a way to preserve the spatial

structure of any unmodelled features (i.e. tidal tails, warps, etc.; see Sect. 6 for details). The model uncertainties are obtained by running the *SingleGalaxyFitter* code on 50 bootstrap resampled cubes. In order to match the analysis as closely as possible, new masks are made for each cube using the SoFiA code (Westmeier et al., 2021). The distribution of parameter values across the bootstrapped fits are compared to the best-fit model to produce statistically meaningful uncertainties. This process is implemented in the set of python scripts currently named *WRKP_GalaxyFitDriver* and associated packages.

Finally the full pipeline script and associated package, currently named *WRKP_CatalogueDriver*, applies the *WRKP_GalaxyFitDriver* to all sources listed in some catalogue file. Unlike FAT and 3DBarolo fits are obtained for almost all galaxies regardless of the data quality. The code only fails if no initial estimate of the galaxy model parameters is found. After finishing all the individual fits, the *WRKP_CatalogueDriver* examines each fit to determine whether the model is reliable using a few, well defined acceptance rules. The accepted models are then grouped together and saved along with all model products.

Currently all pieces of the 3KIDNAS pipeline appear to be working with WALLABY data. The pipeline has been tested using a suite of mock MCGSuite models. In addition 3KIDNAS models have been compared to WKAPP models for the WALLABY pilot data release 1 (PDR1, Westmeier et al. 2022) and pilot data release 2 (PDR2, Murugeshan et al., in prep) sources. New scripts have been developed to integrate the pipeline with the larger AusSRC database in order to automatically select all new sources and attempt kinematic modelling. These scripts are currently being tested. Once all products and testing has been accepted and finalized for version 1 of 3KIDNAS the scripts will be modified to add the accepted models to the AusSRC database once the fitting is completed.

3. Pipeline Products

There are a number of products output by 3KIDNAS. Many relate to individual fits, but some products relate to a suite of kinematic models when 3KIDNAS is run in catalogue mode. In pipeline mode, the products of the accepted kinematic models are reorganized and renamed to follow a similar convention as the pilot phase WKAPP products. At this time, there is still time to adjust and finalize the kinematic models produced by 3KIDNAS.

3.1 Pipeline-Mode Products

The most important set of products are those included in the accepted set of kinematic models from a 3KIDNAS run in catalogue mode. In this mode, fits to almost all galaxies are generated (the only exceptions are when the initial parameter estimation fails; see Sec. 5.4 for more details). These fits are then checked against a set of acceptance criteria. Based on testing described in Sec. 4, these acceptance critera are:

1. Have 3 or more rings.
2. Have a model inclination > 25°
3. Have $\sigma_{PA} \leq 20°$
4. Have $\sigma_{v_{sys}} \leq 40$ km s^{-1}
5. Have $sin(i_{max}) - sin(i_{min}) \leq 0.15$
6. Have a rotation curve and inclination such that no point in the model velocity profile is beyond the cubelet velocity range.

Fits that satisfy all criteria are added to the final kinematic catalogue, and a number of data products are stored for each fit. Table 1 lists all of these products. The most important file listed in Table 1 is the catalogue file as it contains the full set of model parameters for all accepted kinematic models. However, it is also possible to access those parameters for an individual fit through the *_AvgModel.txt file. Figure 1 shows an example of one of these best fit files.

```
Object:  ba_4.26.mass_10.39.inc_37.9.pa_188.2.veldisp_6.7.Phi_0.0.A2_0.0.version_0
Date:   2023-5-18
KINVER:          WRKPv1

nBootstraps     5
nBootstrap_Fits 5

Cube Noise
RMS (mJy/beam)  1.49
SN_Int  31.94
SN_Peak         10.19
SN_Avg  1.76
SN_Median       1.61

Geometry Parameters
Param Name       Value   Error
X_model (pixels)        36.14           0.2742
Y_model (pixels)        35.98           0.2459
RA_model (degrees)      -0.0            0.0
DEC_model (degrees)     -0.0            0.0
Inc_model (degrees)     66.8            12.1
PA_model (degrees)      188.6           1.7
PA_model_g (degrees)    187.0           1.7
VSys_model (km/s)       10450.8         1.5
VDisp_model (km/s)      7.5             3.1

Rotation Curve
nR=     5
Rad             VROT_model      e_VRot_model
('')            (km/s)                  (km/s)
7.5             178.7           35.7
22.5            205.3           21.1
37.5            202.1           25.2
52.5            195.4           31.0
67.5            205.7           26.2

Surface Density Profile
nR=     5
Rad             SD_FO_model     e_SD_FO_inc_model
('')            (Msol/pc^2)     (Msol/pc^2)
7.5             4.0             0.53
22.5            3.1             0.55
37.5            3.3             0.51
52.5            1.6             0.56
67.5            1.1             0.47
```

Figure 1. An example model output file for a 3KIDNAS fit to a galaxy.

Figure 2 shows a sample diagnostic plot for a kinematic model. These plots are designed to show the model parameters at a glance. They also provide rough comparisons to the data, although the best comparison product is the *_DiffCube.fits file as it contains the data - model residuals. As with the model output file itself, the diagnostic plot structure can be changed based on team feedback. For the moment map panels, it is worth noting that the moment 0 map uses a linear scale and not a logarithmic scale. As a consequence, the faintest, most low density gas, can be difficult to discern, which ultimately can cause the moment 0 maps to appear to have a smaller spatial extent than the corresponding moment 1 maps.

The *_Flags.txt file contains extra information about the fit that may be useful as well as some preliminary flags. Currently, it contains the cube RMS, the fit likelihood, the normalized goodness-of-fit, a size flag, and a flag for the initial estimate of

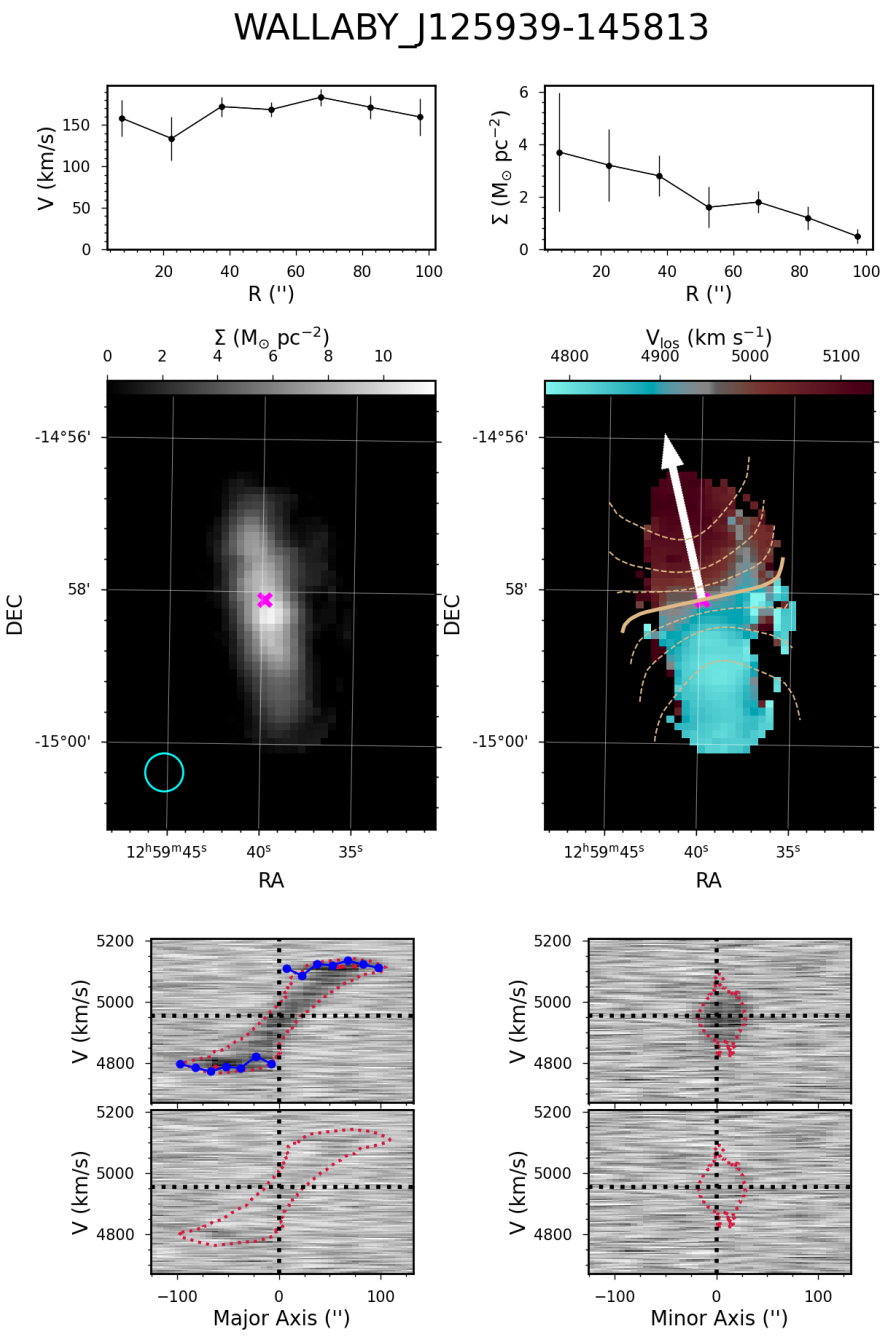

Figure 2. An example model diagnostic plot produced by 3KIDNAS. The topmost line gives the name of the galaxy. The upper left and right panels show the model rotation curve and surface density profile respectively. The left hand moment map shows the data moment 0 map (grayscale), along with the beam (teal circle), and the model center point (magenta x). The right hand map shows the moment 1 map (coloured), the model isovelocity contours (dashed orange lines), the model systemic velocity (solid orange line), and the direction of the position angle (white arrow). In both plots the grid lines show the WCS coordinates. The lower left panels show the major axis position-velocity (PV) diagram of the model (upper) and difference cube (lower), while the right hand panels show the minor axis versions. In the PV diagrams, the dashed red lines show the model PV diagram contours. These contours are set at 1, 3, and 5 times the noise level of the observed PV diagram. The dashed black lines in all PV panels show the model's systemic velocity and center point. The blue points and line in the major axis PV diagram show the projected line of sight velocity ($v_{\rm sys} \pm v(R)\sin(i)$) of the model. Finally, the lower text gives the model's geometric parameters and associated uncertainties.

Table 1. The various data products produced and stored from a catalogue run of 3KIDNAS. The * in the file names is for the specific field/catalogue (for the catalogue file), or the WALLABY name for individual galaxies.

File name	File Type	Description
Single File		
*_KinematicModels.csv	CSV	A catalogue file containing the model parameterization for every successfully modelled galaxy.
Each Galaxy		
*_AvgModel.txt	ascii	A text file containing the fit to a galaxy.
*_DiagnosticPlot.png	png	A diagnostic plot of the fit (see Fig. 2).
*_Flag.txt	ascii	A file containing flags for the fit.
*_ModCube.fits	fits	A realization of the best fitting model.
*_ProcData.fits	fits	A processed version of the observed data cube.
*_DiffCube.fits	fits	The difference between the model cube and the observed cube.
*_PVMajor_Data.fits	fits	The major axis PV diagram of the data.
*_PVMinor_Data.fits	fits	The minor axis PV diagram of the data.
*_PVMajor_Model.fits	fits	The major axis PV diagram of the model cube.
*_PVMinor_Model.fits	fits	The minor axis PV diagram of the model cube.

the center coordinates. It also encodes all the acceptence rules that are checked during the accept/reject step and whether the model passes. For the accepted models, all flags must be passed, but these flags are encoded regardless of the model acceptance. This allows one to check rejected models and determine precisely why they were rejected. More flags/information can be added to this file upon feedback from the team.

The *_ModCube.fits contains a realization of the model at the same resolution as the observed data. The reason for including the *_ProcData.fits file is twofold; the observed source cubelet must be converted from frequency space to velocity space before fitting, and keeping this processed data is helpful for consistency tests.

3.2 Products from Individual Fits

While the data products generated for accepted kinematic models are the most important for the general user, there are additional data products generated for each individual fit before the 3KIDNAS acceptance step. These are the same products that are generated when 3KIDNAS is run for a single galaxy rather than for a catalogue.

Table

Many of the files listed in Table 2 are also present in Table 1 under slightly different names. This is mainly due to changes in the naming convention during the course of the code development and should be addressed. There are two additional files generated for individual fits that are worth discussing. The first is the *_AvgModel_v1.txt file, which contains only the best fitting model without uncertainties. This file is necessary for the bootstrapping process used to calculate the uncertainties (Sec. 6). It is currently kept for an individual fit as it allows for tests/modifications of the uncertainty calculation without having to refit the data itself, as well as for diagnostic purposes. The second file is the *_RTParameters.py file, which is only generated when 3KIDNAS is run in catalogue mode. 3KIDNAS implements catalogue runs using a python wrapper around the individual galaxy fits code. Keeping this file allows for additional tests of individual galaxies after a catalogue run. This is also quite valuable for troubleshooting potential bugs in the fitting code that may arise during a 3KIDNAS catalogue run.

4. Pipeline Tests/Results

3KIDNAS has undergone extensive testing to understand the limits of the code and find areas of future improvement. The tests can be broken up into tests on mock idealized galaxies where the truth is known and comparisons to existing WKAPP fits to WALLABY detections. These more realistic tests allow 3KIDNAS to attempt significantly more complicated objects than those generated by idealized models.

4.1 Idealized Tests

For the idealized tests, we generated three sets of 500 mock galaxies using variations on the Mock Cube Generator Suite (MCGSuite) code (Lewis 2019,Spekkens et al. in prep). MCGSuite generates mock observed data cubes based on a set of scaling relations and observation parameters. The base version of the code, which is publicly availableuses a target HI mass and observed size in beams to construct axisymmetric cubes. We modified the code to include a 'random' model where the mass, size, inclinations, position angle, and velocity dispersion for each mock galaxy is selected from a range of possible values. For our test sample, we generated 500 galaxies with masses between $10^8 \leq M_{\rm HI}(M_\odot) \leq 10^{10.5}$, $2 \leq D_{\rm HI}(\text{beams}) \leq 15$, $15 \leq i(^\circ) \leq 90$, $0 \leq PA(^\circ) \leq 360$, and $6 \leq \sigma_R(\text{km s}^{-1}) \leq 14$. These comprise our fiducial 'Sample 1' for idealized testing.

We further modified the MCGSuite code to generate asymmetric galaxies (see Deg et al. 2023 for a first use of this code). In this modified version, the asymmetry is introduced as an $A1$ Fourier moment in the surface density profile. For our 'Sample 2' we produced a matched sample where there underlying galaxy parameters are those of Sample 1, but a random asymmetry moment is added to the model. In Sample

Table 2. The various data products produced and stored for an individual 3KIDNAS fit.

File name	File Type	Description
*_BSModel.txt	ascii	A text file containing the final fit to a galaxy.
*_BSModel.png	png	A diagnostic plot of the fit (see Fig. 2).
*_AvgModel_v1.txt	ascii	A file containing the best fit to the galaxy without the bootstrap uncertainties.
*_Flags_v1.txt	ascii	A file containing flags for the fit.
*_AvgModel_v1.fits	fits	A realization of the best fitting model.
*_VelCube.fits	fits	A processed version of the observed data cube.
*_DifferenceCube.fts	fits	The difference between the model cube and the observed cube.
*_FitTimeCheck.txt	ascii	A text file giving the runtime of the code for a particular fit.
*_RTParameters.py	python	A python file used to run the individual fit. This file is only produced in catalogue 3KIDNAS runs.
*_PVMajor_Data.fits	fits	The major axis PV diagram of the data.
*_PVMinor_Data.fits	fits	The minor axis PV diagram of the data.
*_PVMajor_Model.fits	fits	The major axis PV diagram of the model cube.
*_PVMinor_Model.fits	fits	The minor axis PV diagram of the model cube.

2, the $0.2 \leq A1 \leq 0.6$. The phase angle of this asymmetric moment relative to position angle is also randomly selected.

Finally, we generated a matched 'Sample 3' of warped galaxies. Again, this required a modification of MCGSuite to implement warps. Rather than using a physical motivation, the warps are described by independent random changes in the inclination and position angle from an inner radius to an outer radius. For the matched sample the start of the warp is between $0.3 \leq R_s\ (R_{\rm HI}) \leq 0.6$. The minimum and maximum changes in inclination and position angle are $10 \leq \delta i(^\circ) \leq 20$ and $30 \leq \delta PA(^\circ) \leq 70$.

Sample 1 is meant to be as realistic as an idealized, axisymmetric set of models can be. By contrast, Sample 2 and Sample 3 were both designed to provide relatively far from axisymmetric models rather than being realistic.

3KIDNAS was run on all 1500 galaxies making up each sample. The resulting 3KIDNAS fits were then classified as a successful, failed, or borderline fit based on a visual comparison to the original MCGSuite rotation curves and surface density profiles. The left-hand column of Figure 3 shows the all the fits for each sample.

At low resolutions, the 3KIDNAS fits tend to have large inclinations. This is due to a bias in the initial estimate of the inclination. 3KIDNAS takes the SoFiA measures of the major and minor axis to get the inclination estimate. To avoid low-inclination biases in more resolved cases, this estimate is then corrected for the beam size. When the galaxy is poorly resolved, this beam correction leads to a large initial estimate of the inclination. And, in those cases, the data is not of sufficient quality to overcome this biased initial estimate. However, it is important to note that using the beam-corrected inclination avoids underestimates of the inclination at moderate resolutions.

The majority of the failures occur when *Ell_Maj*<2 beams. At more moderate resolutions, the failures tend to occur at low inclinations. Based on these results, we developed a set of automated acceptance tests. The specific tests are listed in Sec. 3, but we have relisted them here for convience. In order for 3KIDNAS to consider a kinematic model a success it must

1. Have 3 or more rings.
2. Have a model inclination > 25°
3. Have $\sigma_{PA} \leq 20^\circ$
4. Have $\sigma_{v_{sys}} \leq 40$ km s^{-1}
5. Have $sin(i_{max}) - sin(i_{min}) \leq 0.15$
6. Have a rotation curve and inclination such that no point in the model velocity profile is beyond the cubelet velocity range.

The right-hand column of Figure 3 show only the automatically accepted models. Sample 1 has 217 accepted models with 87% of these being successful fits, Sample 2 has 207 with 83% successful, and Sample 3 has 224 accepted with 88% successful. The acceptance rules can be tuned slightly to increase the total number of accepted models or increase the success rate. These rules are relatively simple, and it may be worth considering more sophisticated approaches in the future.

The similarity of all panels in the right-hand column of Figure 3 is highly promising. It suggests that the adopted flat-disk approach coupled with the automated filtering is sufficient for WALLABY-like sources regardless of possible asymmetries and warps. It is likely that the effects of warps and asymmetries on kinematic models are lower than the uncertainties on the models due to the noise. And, as long as those effects are unbiased, then the accepted models can be used for statistical studies. There are other effects that can be explored with further idealized tests, such as radial flows, tidal tails, etc., but we expect to find similar results; namely that these perturbations on the kinematic models are below the level of the typical WALLABY noise.

While the right-hand column of Figure 3 suggests that warps and asymmetries are not an issue for the overall fits, it is worth exploring whether there are any typical biases in the fits. With the idealized sample, it is possible to calculate the residual for the rotation curve or surface density profile as

$$\Delta = \frac{Y_{fit} - Y_{model}}{\sigma}\ , \qquad (1)$$

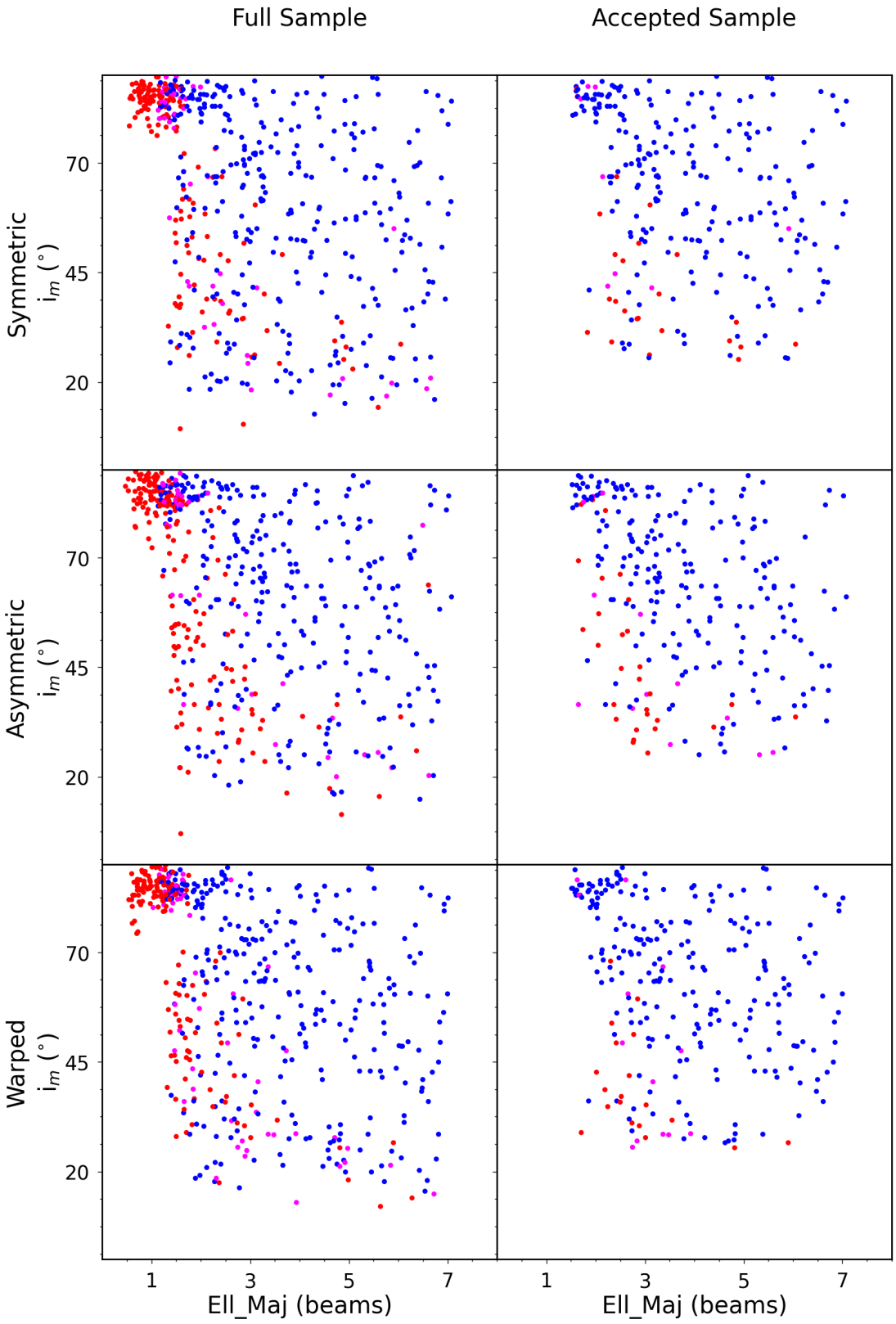

Figure 3. The success and failure of 3KIDNAS on idealized samples. The blue, red, and magneta points indicate successful fits, failed fits, and borderline fits based on visual comparisons to the underlying models. The left hand column shows all fits in a given sample, while the right hand columns shows only those fits that are pass the automated model checks.

where Y_{fit} is the fit data point, Y_{model} is the MCGSuite value at the same radius, and σ is the uncertainty from 3KIDNASFigure 4 shows the rotation and surface density residual curves for all 217 automatically accepted Sample 1 fits (left-hand panels). It is clear that the there is fair amount of fluctuations in the fits, but these are typically on the order of 1 – 2 σ. However, it is also clear that the fits are biased to be high in the inner 1.5 beams.

These points are made clearer in the right-hand panels of Figure 4. They show the averaged residual curve and associated dispersion. The dispersion is $\sim$ 1 σ for all but the innermost beam. This result is critical as it indicates that the 3KIDNAS bootstrap uncertainties are truly representative of the random deviations from the underlying models. In Appendix Appendix 1, Figures 10 and 11 show matching figures for Sample 2 and Sample 3. All three samples show similar results, reinforcing the previous point that the biases due to asymmetries and warps are not at a detectable level for the majority of the WALLABY detections.

Examining the right-hand rotation curve panel of Figure 4 shows that the failed rotation curves typically have fits that are below the model values. This is typically caused by underestimating the inclination. Interestingly, the surface densities are typically recovered well regardless of the success or failure of the model. The solid black lines in these two panels illustrate that the overall accepted sample is of sufficient quality for statistical studies of galaxies over the majority of their extent.

Figure 4 does indicate that the inner 1.5 – 2 beams of the rotation curves and surface density profiles are typically elevated relative to the underlying model. The cause of this bias lies in the initial estimates of the rotation curve and surface density. A first estimate of these profiles is generated by fits to the moment 0 and moment 1 maps of the masked data cube. These first estimates are beam smeared. In order to avoid the effects of beam smearing, a beam correction is applied to these profiles. And it is these beam corrected profiles that are used as the initial estimates in 3KIDNAS. It is clear that this correction is failing in the innermost region. However, tests using the beam smeared estimates without any correction produces stronger biases in the innermost regions.

4.2 WALLABY PDR1 and PDR2

Tests on idealized galaxies are very useful, but real galaxies are much more complex. So, as a base comparison, 3KIDNAS was run an all WALLABY PDR1 and PDR2 sources. 3KIDNAS generated 394 models across from the complete set of sources. However, many of these are models of the same galaxy seen in multiple releases (i.e. Hydra TR1 and Hydra TR2). Taking repeated sources into account, 3KIDNAS produced 295 unique kinematic models from the $\sim$ 2400 unique WALLABY detections. For comparison WKAPP produced 236 unique models for the same set of sources, meaning that the current version of 3KIDNAS has produced 59 more models, which corresponds to a 25% in the modelling success rate.

Figure 5 shows the full suite of rotation curves and surface density profiles for the 3KIDNAS models. These span a large range of masses, rotation velocities, and surface density profiles. One issue seen in the right-hand panel that the surface density profiles do not always reach the 1 $M_\odot$ pc^{-2} threshold. In 3KIDNAS, the rotation curve and surface density profile are intrinsically linked, which is not the case in WKAPP. Ultimately, WKAPP extracts the surface density profile from the moment 0 map of the galaxy (based on the averaged geometry of the FAT and 3DBarolo fits). This allows it to extend further out that the rotation curve.

The extent of the 3KIDNAS profiles is set during the initial estimate of the rotation curve and surface density profiles. Since 3KIDNAS fits all rings simultaneously, once the initial number of rings is set, they cannot be increased or decreased. We have implemented a parameter in 3KIDNAS that cuts off the profiles once they reach a certain surface density limit based on the cubelet noise. This can be used to increase the number of rings in some cases. However, we have found that including too many rings leads to model instability and many more fitting failures. It is possible to explore and optimize this parameter in the future.

5. Detailed Fitting

The core of 3KIDNAS is the *SingleGalaxyFitter* program. This is the program that generates the fit to the model. It can itself be broken into a number of pieces; initial estimates, mock cube generation, model-data comparisons, and the fitting algorithm. Each of these portions are modular and stand on their own. Accordingly, each portion can be replaced in future versions of 3KIDNAS in order to incorporate improved algorithms and code.

Figure 6 shows a flow chart for the *SingleGalaxyFitter* code. The construction of the mock cubes in the flow chart is described in Sec. 5.1. The calculation of the likelihood for a given model is described in Sec. 5.2. The full minimizer used is described in detail in Sec. 5.3, while the initial estimates are described in Sec. 5.4.

5.1 Mock Cube Generation

The goal of *SingleGalaxyFitter* is to compare TR models to some observed data cube. This requires the ability to construct a mock data cube from a TR model. *SingleGalaxyFitter* accomplishes this through a few steps and computational objects. The code itself has been written in Fortran for computational speed.

A TR model consists of a series of rings. Each ring has a surface density, rotation velocity, velocity distribution, center point, inclination, and position angle at minimum. They can also include parameters like the disk thickness, radial velocity, vertical velocity, as well as a dv/dz term. Given the focus of 3KIDNAS on low resolution, low *S/N* observations, and, given that these additional terms act generally as pertubations upon the base model, *SingleGalaxyFitter* currently does not have the ability to fit these additional parameters. And, while it is technically possible to run *SingleGalaxyFitter* in a mode that would consider warps, no testing has been done for such models. The entire focus thus far has been on flat disk models

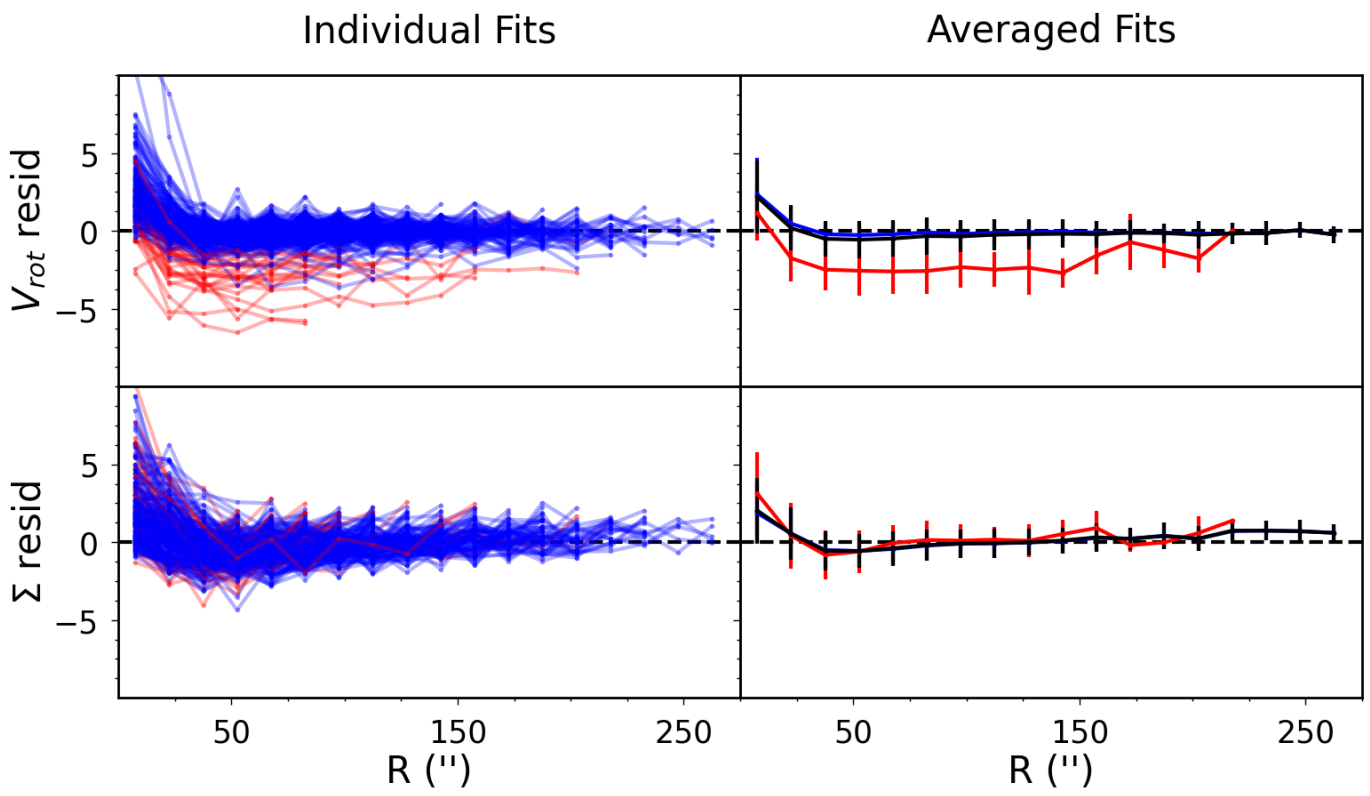

Figure 4. The residuals of the automatically accepted 3KIDNAS rotation curves (top row) and surface density profiles (bottom row) for Sample 1. The left hand column shows all 217 individual residual curves, while the right hand column shows the averaged residual curve. The blue lines show the successful fits from visual classification, while the red lines show the failed fits. In the right hand column, the black line shows the averaged residual curve for the entire sample.

where the geometric parameters remain constant across all rings.

As an intermediate step, the TR model parameters are used to generate a set of tracer particles. For a given ring, the particle positions are drawn randomly in angle and from a sech^2 distribution for the vertical position. Their radial coordinate is drawn from an equal area distribution. Explicitly, this is

$$R_i^2 = \mathrm{rand}\left(R_{max}^2 - R_{min}^2\right) + R_{min}^2 \,, \tag{2}$$

where R_{min} and R_{max} are the inner and outer edges of the ring, and rand is a random number drawn from a uniform distribution between 0 and 1.

The particle positions are then converted from cylindrical coordinates to Cartesian coordinates centered where the Z axis is perpendicular to the galaxy disk. Then the ring's inclination and position angle are used to project the particle positions into observed coordinates. Again, explicitly

$$x_{p,i} = x_i \cos(PA) - (y_i \cos(INC) - z_i \sin(INC)) \sin(PA) \,, \tag{3}$$

$$y_{p,i} = x_i \sin(PA) + (y_i \cos(INC) - z_i \sin(INC)) \cos(PA) \,, \tag{4}$$

where $(x_{p,i}, y_{p,i})$ are the projected positions of the i'th on the sky, while (x_i, y_i, z_i) are the particle's Cartesian coordinates. In this coordinate definition, the positive X-axis aligns with the major axis of the model. Finally, the projected positions are recentered to the ring's center coordinate position.

Moving to the velocities, the only component that is necessary to calculate is the line-of-sight velocity relative to the observer. The contribution to this from the rings rotational velocity is

$$v_{los,rot,i} = v_{rot} \cos(\theta_i) \sin(INC) \,, \tag{5}$$

where θ_i is the particle's angular position (that was selected randomly during the position calculation). Then the observed line-of-sight velocity for the particle is

$$v_{los,i} = v_{los,rot,i} + v_{sys} + \mathrm{rang}\sigma \,, \tag{6}$$

where rang is a random number drawn from a Gaussian distribution with a width of one. It is straightforward to include the contributions from any radial or vertical motions to the line-of-sight velocity using the appropriate projections.

Finally each particle must be given some flux. In *SingleGalaxyFitter*, all particles in a particular ring are assigned equal flux. So, the flux for a given particle is

$$F_i = \frac{\Sigma A}{N} \,, \tag{7}$$

where Σ is the ring's surface density, A is the area of the ring, and N is the total number of particles in the ring. The number of particles in the ring itself is determined

$$N = \rho_p \Sigma^p * n_{\mathrm{pixel}} \,, \tag{8}$$

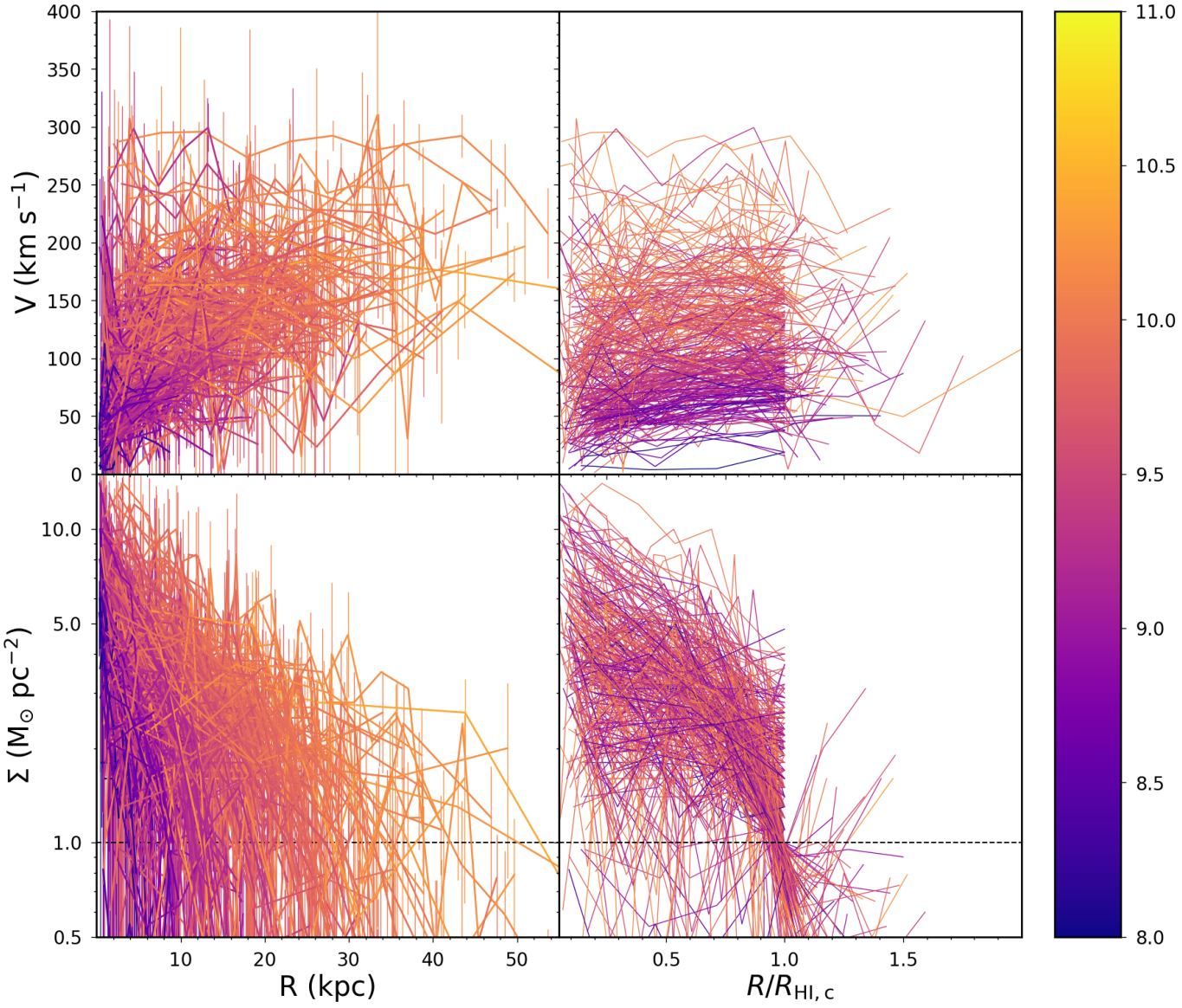

Figure 5. All 3KIDNAS rotation curves (top row) and surface density profiles (bottom row) for WALLABY PDR1 and PDR2 detections. The left-hand panels show the curves in physical space using Hubble flow distances to convert from angular sizes to kiloparsecs. The right-hand panel shows the same curves scaled by $R_{\rm HI}$. In cases where the surface density does not decrease to 1 $M_\odot$ pc^{-2}, the outermost radial point is used as a $R_{\rm HI}$ for this scaling. The colors of each line correspond to the total HI mass of the galaxies from the original SoFiA source detection analysis.

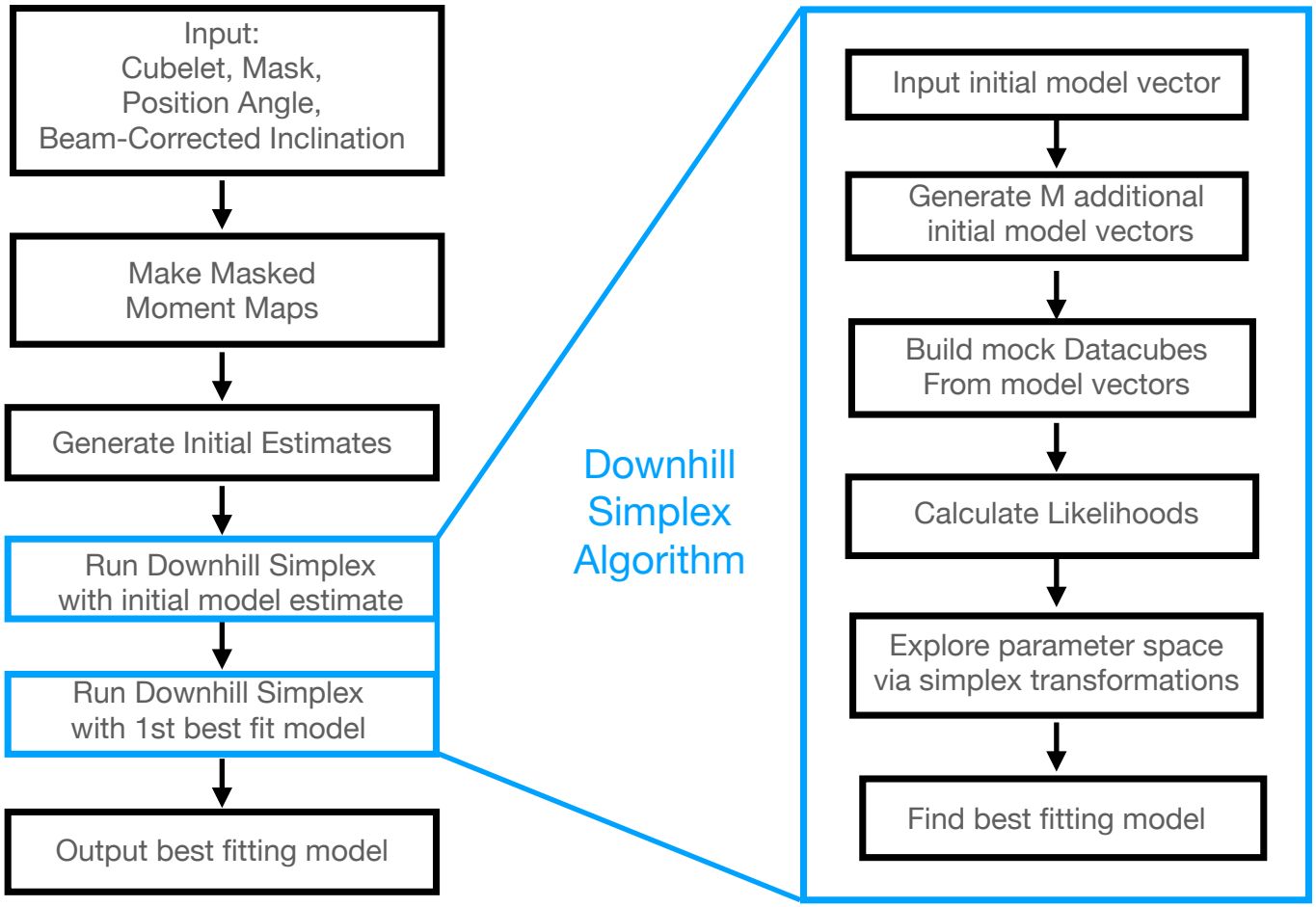

Figure 6. A flow chart describing the *SingleGalaxyFitter* code.

where ρ_p is the 'particle surface density', p is the 'particle density mode', and $n_{\rm pixel}$ is the number of pixels in the ring. ρ_c and c are both user-set parameters. Setting $c = 0$ provides a constant particle density for all pixels. From our tests, we find that $\rho_p = 100$ provides sufficient particle densities for WALLABY models. A lower particle density will speed up the calculations significantly, but can cause issues due to numerical noise.

Once the set of tracer particles are generated, they can be placed into a data cube structure. Essentially, the projected position and velocity are converted into datacube 'cell' coordinates. If those coordinates of a particular particle lie inside the cube, the flux from that particle is added to the particular cell in which it falls. If the model somehow extends beyond the cube structure, only the portion of the flux inside the cube is counted.

Finally, the model cube is convolved with the user defined beam. For this, we use the FFTW library[a] (Frigo & Johnson, 2005). Each channel is independently convolved with the beam under the assumption that the channels are independent. It is possible to include either a second set of 1D convolution to account for any channel correlations, or to modify the convulution to use a 3D 'beam' kernel instead.

To summarize,

1. A TR model consists of M rings.
2. Each ring is described by the parameters Σ v_{rot}, σ, X_c, Y_c, v_{sys}, PA, and INC.
3. For each ring, randomly generate a set of particles.
4. The particle positions in the galaxy's coordinate system are randomly drawn based on the ring parameters.
5. The particle's are then projected to observed coordinates.
6. The particle's observed line-of-sight velocities are then calculated.
7. The projected positions and velocities of all particle's are converted into datacube cell coordinates.
8. The flux from the particle's are added to the appropriate cells in the cube.
9. The filled cube is then convolved with the beam using the FFTW library.

5.2 Model-Data Comparison

Once a model cube is constructed it is necessary to compare it to the observed cube. The simplest method is to use the χ^2 statistic as a proxy for the goodness of fit;

$$\chi^2 = \sum_i \left(\frac{(M_i - D_i)^2}{\sigma^2} \right) , \tag{9}$$

where M_i, D_i, and σ_i are the model flux in a particular cell, D_i is the corresponding observed flux, and σ is the noise in the observed cube. One complication to this formulation is the potential presence of bad pixels or channels in the observed cube. Those are typically set to be NaN's, which will cause this calculation to fail. *SingleGalaxyFitter* avoids this issue by not including any cells with NaN's in the summation. However, it is important to note here that this comparison does not mask the data. That is, the fit includes the entire cube and not just the portions that the source-finding algorithm considers as part of the object.

[a] https://www.fftw.org/

5.3 Minimization Algorithm

With the method to construct model cubes from TR models and to compare those cubes to an observed cube set, the next core piece of *SingleGalaxyFitter* is the minimization algorithm that will search through different possible models to find the one that best fit to the data. There are many different minimization algorithms that could potentially work, ranging from full MCMC analyzes, to gradient descent methods, to grid searches. In this version of 3KIDNAS we have chosen to implement the Downhill Simplex minimization algorithm (Press et al., 1992).

Downhill Simplex minimization is relatively simple, straightforward to implement, quick, and does not require the calculation of any derivatives. In the Downhill Simplex, $M + 1$ points are placed in the model parameter space, where M is the number of parameters being fit. For example, a 3 ring flat disk model will have 6 free parameters from the velocity and surface density profiles plus 5 free geometric parameters. For such a model, the minimizer will require 12 model vectors. The likelihood, $\mathcal{L}$ of each of these 12 models will need to be calculated. As discussed in Sec. 5.2, the likelihood function adopted in *SingleGalaxyFitter* is the χ^2 statistic.

Once all $M + 1$ point likelihoods are calculated they are sorted from lowest to highest likelihood. Starting with the lowest likelihood point, a number of possible new points are considered based on geometric reflections, expansions, or contractions across the surface of other points. This process is repeated iteratively until the set of points converge in likelihood space. The system is considered conserved when

$$f_{tol} > \frac{2|\mathcal{L}_{max} - \mathcal{L}_{min}|}{|\mathcal{L}_{max} + \mathcal{L}_{min}| + \epsilon} , \tag{10}$$

where f_{tol} is a tolerance parameter and ϵ is an arbitrary small number to keep from dividing by zero. For the system to be converged the difference between the best and worst likelihoods normalized by the average likelihood must be below f_{tol}.

It is recommended in Press et al. (1992) that the Downhill Simplex method should be run twice when fitting. The second run should initialize the $M + 1$ points based on the best-fitting model of the first run and use a lower f_{tol}. This is meant to avoid any anomalous stopping and confirm allow for a more precise finding of the best-fitting model. In practice, *SingleGalaxyFitter* uses $f_{tol} = 0.005$ for the first run and $f_{tol} = 0.001$ for the second run.

As noted above, Downhill Simplex requires $M + 1$ models to be initialized. The first run sets the first model to be the one found by the initial estimates (see Sec. 5.4), while the second run uses the best fitting model from the first run. The other M models are generated by perturbations about the first model. Explicitly, the perturbation for a given model parameter, P, is

set to

$$\Delta P = (2\mathrm{ran} - 1)\lambda P_{\mathrm{range}} \ , \qquad (11)$$

where $\lambda = 1$ for the first run and 0.5 for the second run, ran is a random number between 0 and 1, and P_{range} is the range allowed for that particular parameter. For the inclination and position angle $P_{\mathrm{range}} = 10°$. For the central position $P_{\mathrm{range}} = 5$ pixels. The range for the systemic velocity is $P_{\mathrm{range}} = 20$ km s^{-1}, while the range on the rotation velocity parameters is $P_{\mathrm{range}} = 50$ km s^{-1} and $P_{\mathrm{range}} = 5$ km s^{-1} for the velocity dispersion. Finally the range on the surface density parameters is $P_{\mathrm{range}} = 1$ M$_\odot$ pc^{-2}.

To summarize, the minimization algorithm employed by *SingleGalaxyFitter* is

1. Input an initial model to the minimizer.
2. Generate M additional models based on the initial model where M is the number of parameters being fit.
3. Run the Downhill Simplex with $f_{tol} = 0.005$.
4. Generate anothe M models based on the best-fitting model from the first fit.
5. Run Downhill Simplex again with $f_{tol} = 0.001$.
6. Output the best fitting model once convergence is achieved.

5.4 Initial Estimates

The Downhill Simplex minimizer relies upon an initial model estimate. Due to the flatness of the likelihood space, this estimate is critical to the success of the fit. We generally find that the initial estimate imprints a signature upon the final fit, so better initial parameter estimates led to better fits. Moreover, by using better estimates, the minimizer will require fewer likelihood calls, which will in turn speed up the code.

SingleGalaxyFitter takes user input inclination and position angle values for the initial estimates of the inclination and position. In 3KIDNAS these estimates are taken from the SoFiA estimates, but the inclination uses a beam correction. Explicitly, the initial PA estimate is set to the SoFiA `kin_pa` value, and the inclination estimate is

$$INC_{est} = \cos^{-1}\left(\frac{\mathtt{ell\_min}^2 - B^2}{\mathtt{ell\_maj}^2 - B^2}\right) \ , \qquad (12)$$

where B is the beam size.

Once the observed cube and corresponding mask are loaded in, the moment 0 and moment 1 maps, as well as the velocity profiles are constructed from the *masked* datacube. It is important to stress that the masked datacube is only used for the initial estimates, and the full observed datacube is used for the actual fitting. The initial center point is calculated using an iterative center-of-brightness on the masked moment 0 map. This approach is used as a simple center-of-mass can be affected by extended tidal features. Using an iterative approach keeps these features from affecting the center estimate, The procedure for iterative center-of-mass is

1. Calculate the center of brightness for the entire masked moment 0 map.
2. Recenter all pixel coordinates about the calculated center and calculate their distance from the center, R.
3. Set $R_{Max,i} = R_{max}/(i+1)$ where R_{max} is the length of the diagonal across the moment 0 map, and i is the current iteration step in the procedure.
4. Recalculate the center using only pixels within $R_{Max,i}$ of the current center.
5. Compare the previous and current centers. If they are within 0.5 pixels of each other the calculation has converged.
6. If the centers are separated by more that 0.5 pixels, repeat steps 2-5.
7. If there is no convergence within 10 iterations, accept the current center and flag the model.

It is rare for the iterative center of brightness to take more than 4 iterations to converge.

The masked profile is used to obtain an initial estimate of the systemic velocity. First, the profile is smoothed using a boxcar average and the edges are estimated based on the positions of the largest positive and negative slopes of the smoothed profile. Those edges are used to provide an initial estimate of v_{sys}. Then, if the profile is Gaussian, the peak flux is found, or, if the profile is double-horned, the peak fluxes on both the approaching and receding sides of the profile are found (where the approaching and receding sides are determined by the estimated v_{sys}). From those peaks, the edges are found using 30% of the peak flux. Finally the v_{sys} estimate is set as $v_{sys} = (v_{low} + v_{high})/2$, where v_{low} and v_{high} are those 30% edges.

Moving to the velocity dispersion, 3KIDNAS uses simple flat disk models with constant velocity dispersions. The range of possible vleocity dispersions for an HI disk tends to be low as most candidates for modelling are rotation supported. For simplicity, the initial estimate is set to 8 km s^{-1}, and the Downhill Simplex minimizer explores a wide range of possible velocity dispersions during the fitting process.

The initial estimates for the surface density and velocity profile are the most difficult to obtain. They are also critical as this is where the total number of rings of the TR model is set. To begin, the user sets a parameter giving the number of rings per beam. In general, 3KIDNAS uses 2 rings per beam, but this can be adjusted. Then the maximum number of possible rings are initialized by finding the largest distance between the estimated center point and the edge of the moment maps and dividing it by the ring size. Next, initial profiles a generated by creating a one beam thick slice along the moment 0 and moment 1 maps. The slice is further divided into bins that are one ring thick. The surface density and line-of-sight velocity in each bin is calculated as the average of the pixels. The matched approaching and receding bins are then averaged together to make a radial profile for the surface density and rotation velocity. These are then corrected using a simple cos and sin correction for the surface density and rotation velocity respectively.

These first pass radial profiles require a number of checks and analysis to convert them to initial estimates for use in the minimization. First, each ring must be checked to see if it is

indeed modelable. If a ring contains no flux (for instance, if the radial bin is in the masked region), it is flagged. For rings with flux, the estimated surface density is compared to the noise level given as $\Sigma_{lim} = \sigma\delta V$, where δV is the channel width. If $\Sigma(R) < f_{\rm noise}\Sigma_{lim}$, where $f_{\rm noise}$ is a user defined limit, that ring is also flagged. Based on testing, $f_{\rm noise}$ is set to 3 as extending too far beyond this limit can lead to instability in the minimizer. Additionally, if the estimated surface density is larger than 50 $M_\odot$ pc^{-2} or the estimated velocity is below 0 km s^{-1} or above 300 km s^{-1}, they are replaced by those respective limits. If, after this initial check, no rings are unflagged, the fit has failed and the minimizer will not run.

From this point, there are two possibilities for fitting. The first is that this is the first time *SingleGalaxyFitter* is called in 3KIDNAS. In that case, only the unflagged rings are used for kinematic modelling. However, if this is one of the bootstrapped fits (see Secs. 6 and 7), it is necessary to use the same rings as the initial fit. In that case, even flagged rings can be used. If a ring in these bootstrapped fits is flagged, the nearest interior profile values are used for the flagged ring. If this flagged ring is the innermost ring, then *SingleGalaxyFitter* searches for the closest unflagged ring to use.

At this point the profiles are beam corrected. For each point on the ring, a beam corrected radius is calculated as

$$R_c^2 = R^2 - (B/2)^2 \,. \tag{13}$$

The initially estimated profile is linearly interpolated to find the estimated profile values at R_c. These corrected values are used to construct a beam corrected profile. It is these beam corrected profiles that are used as the initial estimates of the surface density and rotation velocity for the Downhill Simplex minimizer.

6. Flipping Bootstrap

Estimating the uncertainties is non-trivial given the complex nature of the models. It is further complicated as the models, while complex, may not fully describe the system. In order to deal with these residuals, we have built a novel 'flipping bootstrap' algorithm.

In bootstrap resampling, the core idea is to take generate multiple mock realizations of a data set by subtracting the model from the data, scrambling the residuals, and then re-adding the model to make the bootstrapped sample. These models can then be fit repeatedly and the distribution of those bootstrapped fits can be used as an estimate of the uncertainty. This method has been used with great success when the uncertainties are driven by random, uncorrelated noise. Unfortunately, in our case, the noise is correlated, and, there may be non-random residuals from unmodelled effects, like tidal tails, radial flows, warps, asymmetries, and more.

Another important point to consider is the spatial-velocity structure of the residuals. For example a tidal tail may appear in only one or two velocity channels. Since our modelling does not consider such effects, it will leave a residual. A simple method of scrambling residuals that keeps spatial structures correlated would be to swap velocity channels at random. However, the new location of the tidal tail residual may no longer be connected to the main body of the galaxy. This will have a fairly different effect on the fitting than a connected tidal tail.

In order to preserve the spatial-velocity structure of the residuals, as well as to keep their relative position to the model constant, the flipping bootstrap algorithm allows two possible types of swaps; a flip across the major axis or a flip across the minor axis coupled with a flip across the systemic velocity. For a cube with N channels and, allowing for M correlated channels of residuals, the total number of possible samples is

$$T = \left(\frac{N}{M}\right)! \times \left(\frac{N}{2M}\right)! \,. \tag{14}$$

The first term is from the major axis flips, while the second term is from the minor axis flips. The division by M is due to the reduction of possible flips by grouping channels together. Despite the large number of independent samples, it is worth noting that a particular residual, like a tidal tail, can only appear in one of 4 locations in a given sample.

The flipping bootstrap algorithm requires a model cube along with the model center point (including the velocity) as well as the position angle. In practice the algorithm acts as

1. Generate the residual cube by subtracting the model from the data.
2. Use the model center and position angle to calculate all cell coordinates in $(R, \theta, \delta V)$ units where θ is the angle of the pixel from the position angle.
3. Go through each channel and randomly select whether it will flip across the major axis. If $M \geq 2$, then both the current channel and the next $M-1$ channels are flipped simultaneously, and the flipping check will only begin again at the $M+2$ channel.
4. When flipping across the major channel, the flip paired coordinates for a given cell are $(R, 360^\circ - \theta, \delta V)$. The flux from the paired cell is then placed in the current cell. This mapping is done for all cells in a channel that is being flipped across the major axis.
5. Once all channels have been cycled through and determined whether they will be flipped or not, a similar process is performed for the minor axis flips.
6. For the minor axis flip, the paired flux is located at $(R, -\theta, -\delta V)$. Again, all cells have their flux replaced with their paired flux.

During the flipping process it is possible for the a cell's particular flipped coordinates to lie outside the cubelet. In such a case, no change to that cell's residual flux occurs.

Figure 7 shows an example of the bootstrap sample construction for WALLABY J100903-290239. This galaxy has an extended tidal tail. The gif shows a comparison of the data, model, residuals, flipped residuals, and reconstructed mock cube across most channels. In this case, a number of interesting behaviours are seen. Firstly, much of the most extended tail is not flipped as it runs into the edges of the cube. However, this is in a region that the model does not consider, so it should not strongly bias the results.

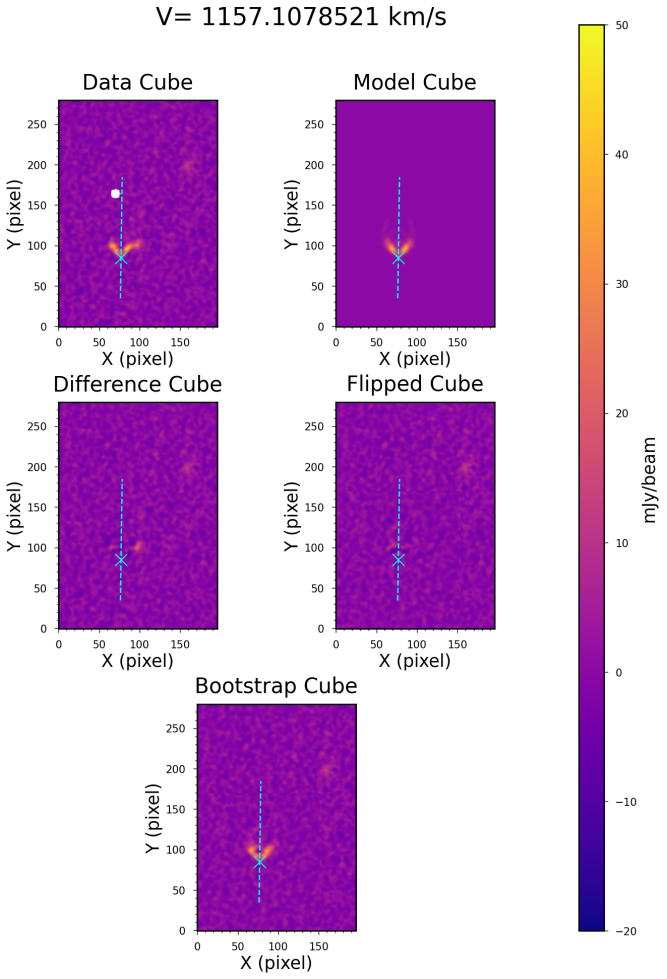

Figure 7. An example of the process of applying the 'flipping bootstrap' to WALLABY J100903-290239. This gif compares the data, model, residuals, flipped residuals, and bootstrapped cube on a channel-by-channel basis.

Fitting a set of bootstrap resampled cubes generates a distribution of model fits. In principle, that distribution should represent the uncertainties in the fit. In order to check whether this is true, we generated 100 realizations of the same idealized mock galaxy. These 100 cubelets all have the same underlying model, but different noise realizations. Fitting each of those 100 cubelets individually provides an estimate of the 'true' uncertainty that is expected based on the noise level (which is set to 1.6 mJy/beam). We then took the fit to the first realization and used it to generate 100 bootstrap sample cubelets. The distribution of fits to the 100 bootstrap resampled cubelets is compared to the distribution of fits to the 100 model realizations in Figure 8.

There are a number of important results apparent in Figure 8. The first, and perhaps most critical, is that the size of blue and red shaded regions are approximately the same. This means that the width of the distribution of bootstrapped fits matches the distribution width expected given the underlying noise. In most cases the blue and red regions overlap, but this is not the case for the inclination. There, the bootstrapped resamples are different than the initial fits. More importantly, they differ from the underlying model used to generate the bootstraps (indicated by the solid red lines in Figure 8). When calculating the uncertainty, this difference should be considered. Accordingly, we set the uncertainty as

$$\sigma^2 = \mathrm{rms}^2 + (M - \mathrm{avg})^2 \ , \qquad (15)$$

where M is the initial model fit, while avg and rms are the mean and standard deviation of the bootstrap distribution of fits. In this example we used 100 fits, but through testing, we've found that 50 bootstrap fits is sufficient to describe the model uncertainties.

While we have focused on incorporating the 'flipping bootstrap' into 3KIDNAS, it is worth noting that it can be utilized with any other modelling algorithm that generates mock cubes. To that end, the flipping bootstrap has been implemented as a discrete program called *BootStrapResampler*. It, like *SingleGalaxyFitter* is written in Fortran for improved computational speed.

7. Complete 3KIDNAS Modelling

The combination of *SingleGalaxyFitter* (described in Sec. 5), *BootStrapResampler* (described in Sec. 6), and the existing SoFiA code provide the core of 3KIDNAS. They are tied together through the use of python scripts to create the full 3KIDNAS pipeline. The core scripts are currently named *WRKP_GalaxyFitDriver* and *WRKP_CatalogueDriver*.

Figure 9 shows the 3KIDNAS flowchart, broken up between the *WRKP_GalaxyFitDriver* and *WRKP_CatalogueDriver* scripts. *WRKP_GalaxyFitDriver* generates the 3KIDNAS model for a single galaxy. It begins by taking in the galaxy cubelet, mask, and shape estimates. For WALLABY, those estimates are extracted from the SoFiA catalogue (this extraction occurs in *WRKP_CatalogueDriver*). Those inputs are then passed to *SingleGalaxyFitter* to obtain the best fit model. The best fitting model is then used to generate N bootstrap

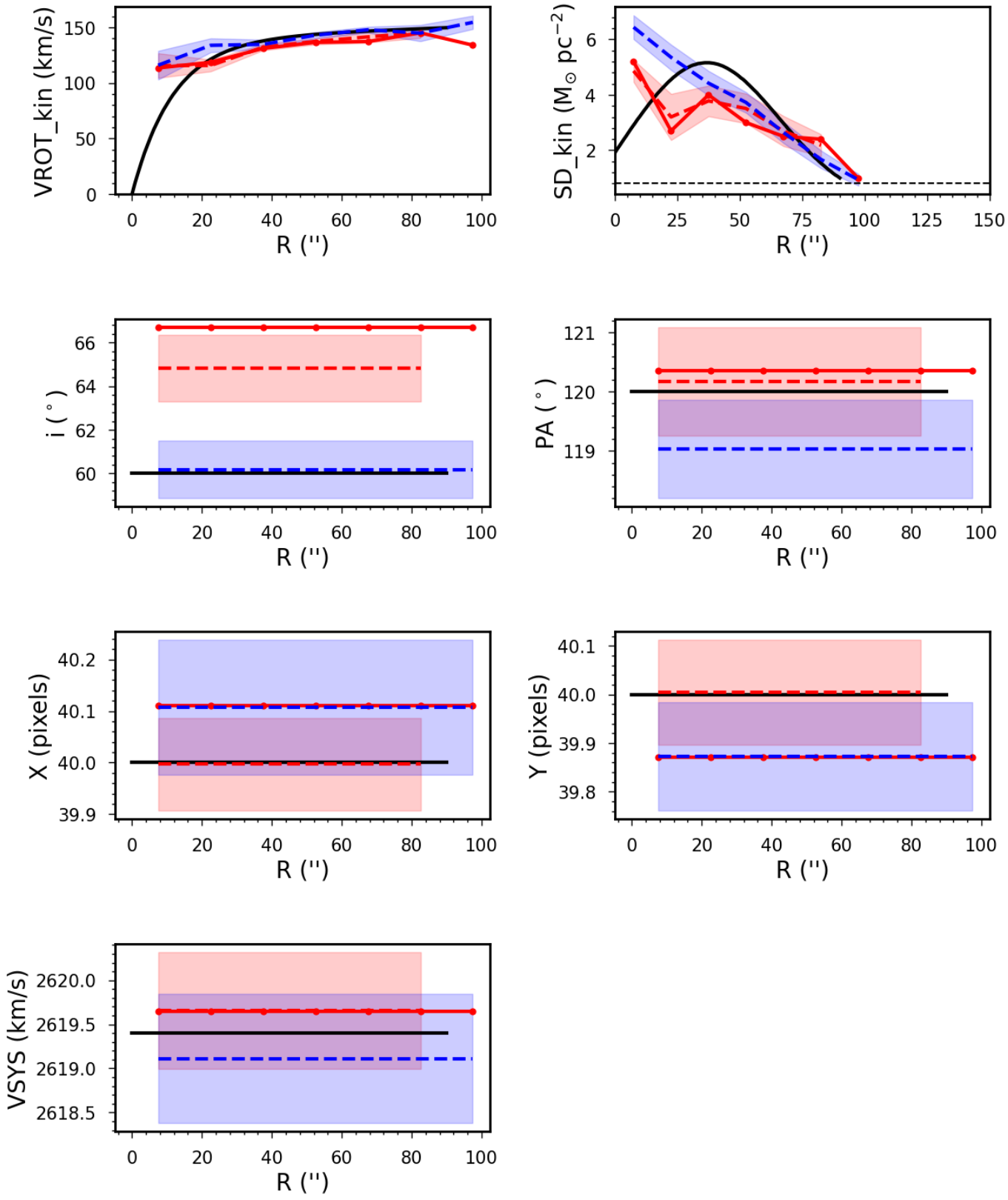

Figure 8. A comparison of 100 fits to different realizations to the same underlying model (red) to the results of fits to 100 bootstrap resampled cubes (blue). The black lines show the true model. The dashed lines and shaded regions show the average and one sigma distribution of fits. The solid red line shows the first fit to the independent samples. That model is used as the basis for the bootstrap samples.

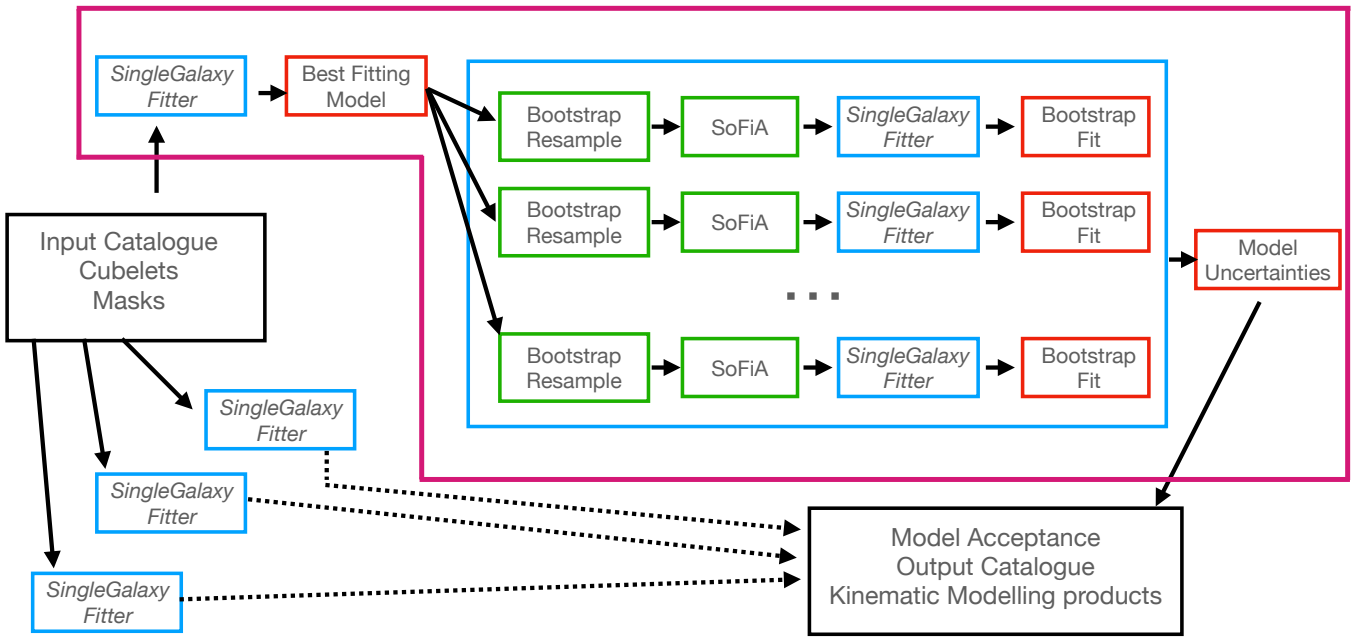

Figure 9. A flowchart describing the overall structure of 3KIDNAS. The large magenta box encloses the content of *WRKP_GalaxyFitDriver*, while the portions outside the box are included in *WRKP_CatalogueDriver*.

cubelets (as discussed in Sec. 6 we find that setting $N = 50$ is sufficient). Next SoFiA is used on each bootstrapped cubelet to obtain a mask and geometry estimate, which are in turn passed through *SingleGalaxyFitter* to obtain N bootstrapped models. The bootstrap fitting has been parallelized as the process is 'embarassingly parallel'. The distribution of bootstrapped fits are then compared to the best fitting model to obtain the uncertainties. In Figure 9 the contents of *WRKP_GalaxyFitDriver* are contained in the large purple box.

WRKP_CatalogueDriver is the used to generate models for an entire SoFiA catalogue. It takes in the SoFiA catalogue (currently in a FITS file format), along with the full set of cubelets and masks. It then runs the *WRKP_GalaxyFitDriver* script for each galaxy. As with the bootstrap portion of *WRKP_GalaxyFitDriver*, this process is embarassingly parallel, so the script runs these in parallel. Thus, the total number of threads used in a 3KIDNAS run is $N_{\rm threads} = N_{\rm galaxy} \times N_{\rm bootstrap}$ where $N_{\rm galaxy}$ is the number of galaxies analyzed in parallel at a time and $N_{\rm bootstrap}$ is the number of bootstrap fits run in parallel at a time. Once all the fits are complete, *WRKP_CatalogueDriver* goes through each fit and checks whether it satisfies the acceptable modelling criteria (given in Sec. 4.1). If a model is acceptable, the appropriate products are added to the acceptable models folder and the model parameters themselves are added to a catalogue file.

8. Limitations and Future Improvements

3KIDNAS appears to be working quite well based on our early results. It currently takes less than one day to analyze the output for a particular field using computing resources from the Canadian Advanced Network for Astronomical Research (CANFAR). This is a marked improvement over WKAPP which takes closer to a week to run. We have generated additional scripts for WALLABY that allow 3KIDNAS to check AusSRC for new, unmodelled detections, and then automatically run 3KIDNAS for those particular sources. Once the version one of the pipeline is finalized, we will set up the scripts necessary to automatically push those models to AusSRC for immediate use by the WALLABY team.

Despite this, there are a number of limitations and weaknesses in 3KIDNAS that should be noted. And there are many areas where 3KIDNAS can be improved. And, while these are all important to consider, address, and implement in the future, the current priority is determining whether the version of 3KIDNAS presented in this report is ready to replace WKAPP for the full survey.

The most obvious limitation of 3KIDNAS is that it is tuned to flat disk models. With some small modifications, it would be possible to consider more complex models, such models are not an anticipated use case for 3KIDNAS. The majority of WALLABY detections are low resolution and low S/N. Complex models, considering warps, radial flows, and more, cannot be constrained by the typical WALLABY detection. And, as shown in Sec. 4, the perturbations caused by such

features are below the typical model uncertainties. Moreover, the focus of 3KIDNAS is on statistical studies of the properties of galaxies. That requires the generation of models for as many galaxies as possible using identical methods. This drives the focus on flat disk models. Nonetheless, there are certainly a subset of detections that are resolved enough to explore complex models. For those cases, we currently recommend using other modelling codes that are optimized for such studies.

The next weakness to discuss in 3KIDNAS is the use of the Downhill Simplex algorithm for minimization. This is a relatively simple algorithm that rapidly explores parameter space. However, there are other, more efficient algorithms that could be explored in the future in order to increase the speed of the code. More importantly however, is the dependence of the Downhill Simplex algorithm on the initial estimates. The flat likelihood space allows the code to converge rapidly, but it also imbues the code with a strong dependence on the initial conditions and also means that the model may converge to a false minima. There may be other algorithms that are less dependent on the initial conditions, including gradient descent methods, cloud based sampling, and full MCMC analysis. While we feel that the Downhill Simplex coupled with our flipping bootstrap is sufficient for WALLABY observations, we will certainly consider other algorithms in the future.

Related to the the weaknesses of the minimizer are possible biases due to the initial estimates. While using the Downhill Simplex algorithm, it is critical that the initial estimates be reasonably robust. Sec. 5.4 discusses how the various initial estimates are derived in detail, and almost all of them could be improved. Of particular import is the calculation of the beam-corrected profiles, as Sec. 4 shows that these are leaving an imprint in the innermost beam of the final models. Additionally, the noise cutoff adopted (which does tend to stabilize the models), also means that many models do not reach $R_{\rm HI}$. As long as the minimizer depends on the initial estimate (as some methods may not), it is critical that the initial estimates be reasonable.

Another possible improvement to future versions of the code is the model acceptance rules. The rules adopted in Sec. 4 are relatively simple and lead to a $\sim$ 13% contamination in the accepted model catalogue. It may be that these rules can be modified, or additional rules included that would improve the purity of this catalogue without significantly reducing the number of accepted models. Alternatively, some machine learning algorithms may be adopted that could provide improvements in the final catalogue.

The above limitations and weaknesses comprise the largest set of issues with 3KIDNAS, but it is not exhaustive. There are likely additional biases in the code that we have not yet found in our tests. One possible avenue for future work and exploration is to apply 3KIDNAS to a set of mock observations from various cosmological simulations. That will allow for some detailed comparisons and quantifications of issues, as well as provide another avenue to compare the results from the WALLABY survey to these simulations in order to extract additional science.

Despite all these issues and limitations, it is important to reiterate that 3KIDNAS seems to be doing well at modelling WALLABY detections. It is obtaining 25% more models than WKAPP when used on the same sets of observations, and these models are produced faster than WKAPP with more robust uncertainties and fewer biases. The use of flat disk models does not appear to be affecting the models when considering mock WALLABY observations (although it will be useful to detemine when this approximation will no longer hold). The code is at an advanced stage of readiness and has already been shared with WALLABY TWG5 for additional examination and testing.

References

Deg, N., Perron-Cormier, M., Spekkens, K., et al. 2023, MNRAS, 523, 4340
Deg, N., Spekkens, K., Westmeier, T., et al. 2022, PASA, 39, e059
Di Teodoro, E. M., & Fraternali, F. 2015, MNRAS, 451, 3021
Frigo, M., & Johnson, S. G. 2005, Proceedings of the IEEE, 93, 216, special issue on "Program Generation, Optimization, and Platform Adaptation"
Kamphuis, P., Józsa, G. I. G., Oh, S. . H., et al. 2015, MNRAS, 452, 3139
Lewis, C. 2019, PhD thesis, Queen's University at Kingston, Canada
Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992, Numerical recipes in FORTRAN. The art of scientific computing
Westmeier, T., Kitaeff, S., Pallot, D., et al. 2021, MNRAS, 506, 3962
Westmeier, T., Deg, N., Spekkens, K., et al. 2022, PASA, 39, e058

Appendix 1. Supplemental Figures

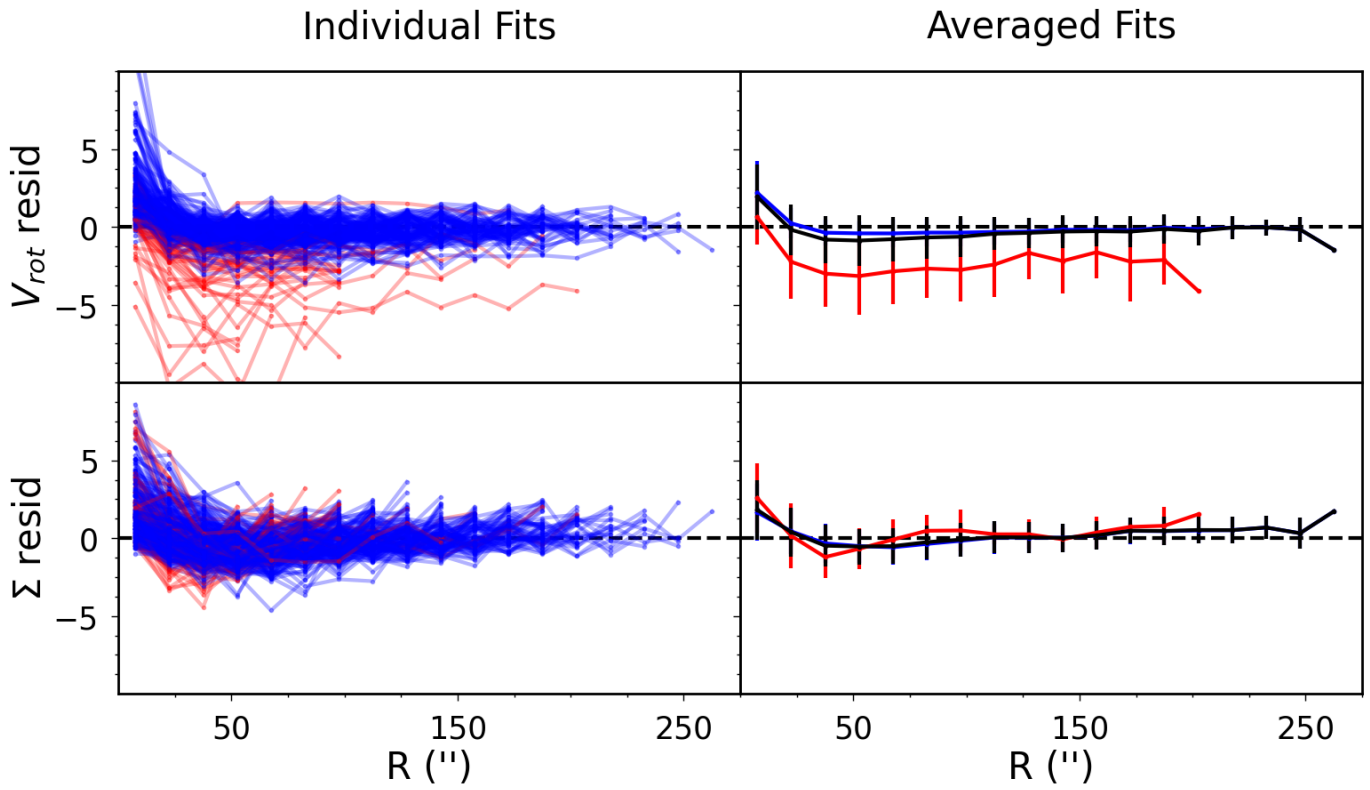

Figure 10. The residuals of the automatically accepted 3KIDNAS rotation curves (top row) and surface density profiles (bottom row) for Sample 2. Panels and lines are as in Figure 4.

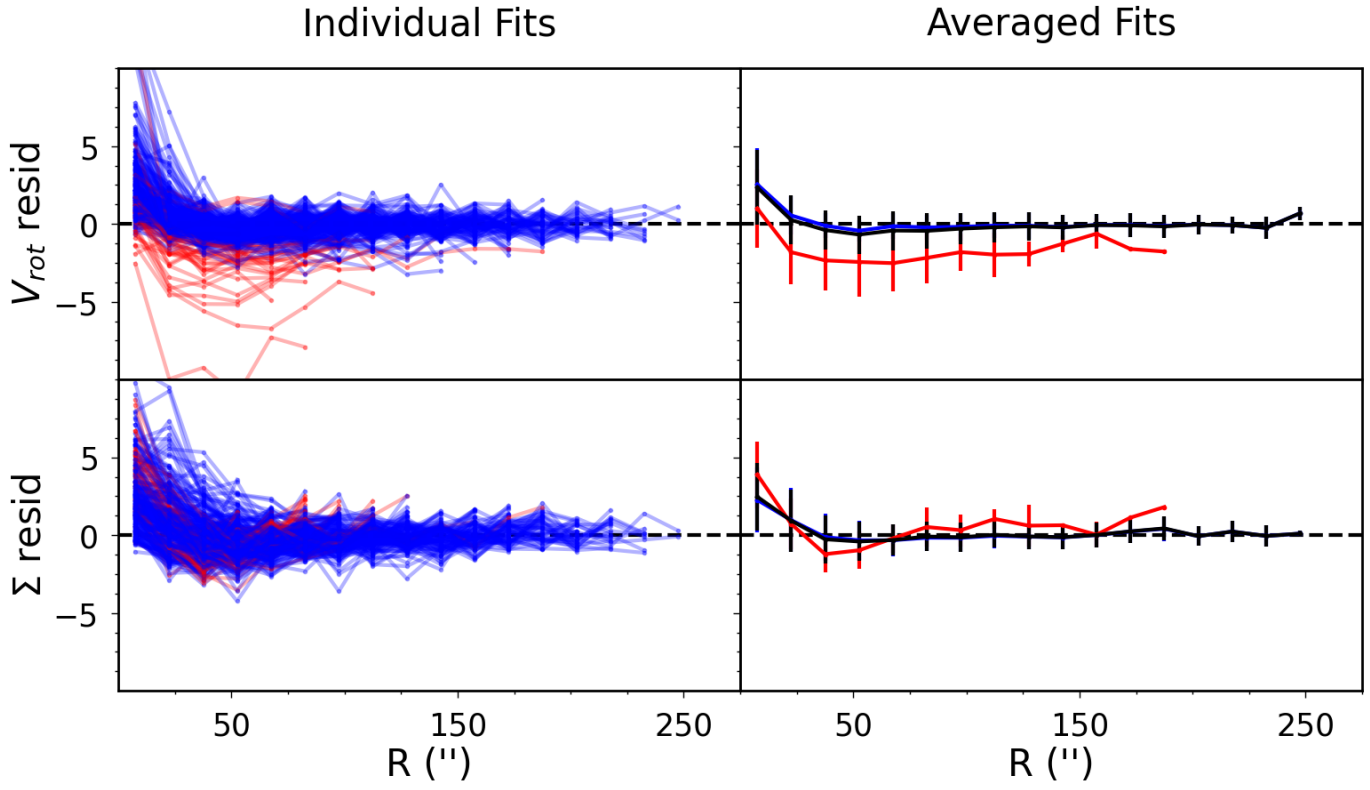

Figure 11. The residuals of the automatically accepted 3KIDNAS rotation curves (top row) and surface density profiles (bottom row) for Sample 3. Panels and lines are as in Figure 4.