Draft version August 31, 2020
Typeset using LaTeX **preprint** style in AASTeX63

An Interpolation Method for Irregularly-Sampled Multi-Dimensional Data Cubes

Daniel Perera[1]

[1]*SOFIA-USRA, NASA Ames Research Center, MS 232–12, Building N232, PO Box 1, Moffett Field, CA 94035–0001, USA*

ABSTRACT

We present a method of interpolating irregularly-sampled multi-dimensional data onto arbitrary coordinates using local polynomial regression. Local regression techniques fit models to a subset of samples in the vicinity of a given coordinate, and may optionally weight samples according to measurement error and/or relative position, resulting in a smooth, self-consistent function that may be evaluated over the entire sample space. Additionally, we derive an optional method for determining local kernel estimators in a single step, approximating an optimal reduced chi-squared statistic on the fit.

Keywords: resampling, interpolation, local polynomial regression, multivariate polynomials, kernel estimation

1. INTRODUCTION

Irregularly sampled data are commonplace in astronomy and other scientific fields, where interpolation onto a regular spaced grid is almost always required for visualization and analysis. However, resampling data in three or more dimensions simultaneously is complicated and computationally expensive, so frequently one resorts to resampling multiple times in a series of steps for each dimension. This process can be time-consuming and prone to propagating errors, introduced in the interpolation procedure, from one dimension into others. It also does not account for the continuity requirements on adjacent values in the multiple dimensions imposed by the physical data collection process. In this paper, we outline a method to carry out simultaneous resampling (interpolation) in multiple dimensions.

1.1. *Motivation*

The FIFI-LS instrument (Colditz et al. 2018; Fischer et al. 2018) used on SOFIA is an example of an integral field spectrograph that generates data cubes in which the original values are not evenly sampled on a regular three-dimensional grid. As described by Fischer et al. (2018), FIFI-LS records flux values at an array of discrete 25 x and y positions on the sky (spaxels), and at an array of 16 wavelengths λ. The spaxels are not regularly spaced on the sky. Furthermore, the set of sampled wavelength values varies from spaxel to spaxel, and some measurements are either missing or noisy. Vacca et al. (2020) [1] give a brief overview of the pipeline developed to convert instrumental

dperera@sofia.usra.edu

[1] See also the FIFI-LS Observer's Handbook and The FIFI-LS Guest Observer (GO) Data Handbook.

values recorded by the instrument into flux estimates at regularly sampled spatial and wavelength coordinates. The initial version of this data processing pipeline dealt with the irregularly sampled cloud of data points in three dimensions by first interpolating the flux values in wavelength at each spatial point onto a regular array, so that each spaxel was assigned the identical wavelength array. Then at each wavelength, interpolation was performed again in the spatial dimension to compute fluxes on a regular spatial grid. In this way, a regularly sampled data cube in three dimensions (space and wavelength) was generated. Although this procedure is effective, it can be slow, and the wavelength interpolation ignores the coupling (due to physical continuity requirements) of data points that are spatially adjacent (and therefore within the instruments point spread function).

1.2. *Local Polynomial Regression (LPR)*

One of the most employed and well understood methods of interpolating data is to model sample measurements as a smooth function of unknown parameters, and then estimate a set of parameters that best fit those data using regression. For the purposes of this paper, polynomials functions are used for their versatility and ease of use.

One could choose to model the entire sample space with a single set of parameters that describe the "True" underlying function of the noisy sample measurements. However, as the complexity of the structure to be modelled increases, so too does the number of parameters required, leading to numerous difficulties. Numerical instability becomes more pronounced in higher dimensions and for higher polynomial orders (Noferini and Townsend (2016)). Gelman and Imbens (2019) show that for polynomials, high order fits have increased sensitivity to noise, and strong dependence on the order of polynomial fit. It is therefore preferable to derive numerous model from subsets of samples in the local vicinity of each point at which a fit is required.

LPR is discussed in detail by (Fan and Gijbels 1996), with multivariate extensions provided by (Masry 1996), and expanded upon by (Gu et al. 2015). All methods essentially minimize the following problem via weighted least squares given by (Fan et al. 1996) as

$$\underset{\hat{c}}{min} \sum_{i=1}^{N} \left(y_i - \sum_{j=0}^{p} c_j (x_i - v)^j \right)^2 Wt(\frac{x_i - v}{h}) \tag{1}$$

where $(x_1, y_1), \cdots, (x_N, y_N)$ form an i.i.d sample from a certain population, Wt is a non-negative, symmetric, and bounded weighting function with smoothing parameter (or bandwidth) h, and v is the point of interest. h may be optimized globally or locally for each resampling point by balancing the bias and variance of $\hat{c}$ such that the optimal bandwidth minimizes the Mean Squared Error (MSE) of $\hat{c}(x) - c(x)$

$$\hat{h} = arg\,\underset{h}{min}\,\hat{\text{MSE}}(f(\hat{c}, \boldsymbol{x})) \tag{2}$$

Unfortunately, this is not always effective when resampling irregular data taken from real observations. The optimal bandwidth given by (Gu et al. 2015) assumes that x are i.i.d, with a common density, and errors on the sample measurements (ϵ) are likewise i.i.d with zero mean, and independent of x. These assumptions cannot always hold for FIFI-LS observations on SOFIA when data are combined over multiple flights over long periods of time, leading to uneven sampling density, and calibration errors that can vary substantially between samples.

Additionally, little attention has been given to defining what constitutes the “local” subset of samples in a fit. While the standard method of selecting the N-nearest neighbors is acceptable if x is i.d.d, doing so for irregularly sampled data in which there is no such guarantee is a dangerous tactic to employ. For FIFI-LS observations, the spatial and spectral response of the telescope is known, it is scientifically irresponsible to perform regression on samples for which there is no coupling to the response function. In higher dimensions, the problem of determining local sample subsets using nearest-neighbors becomes increasingly severe as there is no accounting for sufficient sampling along a given dimension, potentially leading to unstable solutions and co-linearity. The following LPR implementation attempts to overcome many of these difficulties while remaining computationally viable.

2. FORMULATION

2.1. *Polynomial Representation of K-Dimensional Data*

We assume that the input data consist of N measurements (samples) y distributed over K dimensions and that the ‘true’ (but unknown) generating function for the measured values is given by f which can be adequately represented by a K-dimensional polynomial function whose order o_k is allowed to vary in each dimension k. Then at the coordinate $\boldsymbol{x}$ we have,

$$f(\boldsymbol{x}) = \sum_{p_1=0}^{o_1} \sum_{p_2=0}^{o_2} \cdots \sum_{p_K=0}^{o_K} \lambda_{(p_1,p_2,\cdots p_K)} c_{(p_1,p_2,\cdots p_K)} x_1^{p_1} x_1^{p_1} \cdots x_K^{p_K} \tag{3}$$

Here, c are the coefficients of the polynomial and p are the exponents of the coordinate variables x for the K dimensions, while $\lambda_{(p_1,p_2,\cdots p_K)}$ is defined as

$$\lambda_{(p_1,p_2,\cdots p_K)} = \begin{cases} 1, & \text{if } \sum_{k=0}^{K} p_k \leq max(o) \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

We can then define a set S containing all polynomial exponent combinations where $\lambda = 1$,

$$S = \{(p_1, p_2, \cdots p_K) | \lambda_{(p_1,p_2,\cdots p_K)} = 1\} \tag{5}$$

and can then re-write equation 3 in the form

$$f(\boldsymbol{x}) = \sum_{s \in S} c_s \Phi_s \tag{6}$$

where the polynomial terms of the equation are given as

$$\Phi_s = \prod_{k=1}^{K} x_k^{s_k} \tag{7}$$

The set s is in lexographic order such that (for $K = 3$) $\{0, 1, 1\}$ comes before $\{1, 0, 0\}$. As an example, when $K = 2$ and $o_k = 2$ for all k we have

$$\begin{aligned}
s_{(p=2,K=2)} &= [(0,0),(0,1),(0,2),(1,0),(1,1),(2,0)]\\
\Phi &= [1, x_2, x_2^2, x_1, x_1x_2, x_1^2]\\
f(\boldsymbol{x}) &= c_1 + c_2x_2 + c_3x_2^2 + c_4x_1 + c_5x_1x_2 + c_6x_1^2.
\end{aligned}$$

Similarly, for $K = 3$ where $p_1 = 1$, $p_2 = 2$, and $p_3 = 3$ we have

$$\begin{aligned}
s_{(p=[1,2,3],K=3)} = \quad & [(0,0,0),(0,0,1),(0,0,2),(0,0,3),(0,1,0),(0,1,1),(0,1,2),\\
& (0,2,0),(0,2,1),(1,0,1),(1,0,2),(1,1,0),(1,1,1),(1,2,0)]\\
\Phi = \quad & [1, x_3, x_3^2, x_3^3, x_2, x_2x_3, x_2x_3^2, x_2^2, x_2^2x_3, x_1x_3, x_1x_3^2, x_1x_2, x_1x_2x_3, x_1x_2^2]\\
f(\boldsymbol{x}) = \quad & c_1 + c_2x_3 + c_3x_3^2 + c_4x_3^3 + c_5x_2 + c_6x_2x_3 + c_7x_2x_3^2 + c_8x_2 + c_9x_2^2x_3 +\\
& c_{10}x_1x_3 + c_{11}x_1x_3^2 + c_{12}x_1x_2 + c_{13}x_1x_2x_3 + c_{14}x_1x_2^2
\end{aligned}$$

2.2. *Resampling Algorithm*

With the above representation for f we now wish to evaluate $f(\boldsymbol{x})$ at a point $\boldsymbol{x} = \boldsymbol{v}$. We assume that f has been sampled at N data (or measurement) points $\boldsymbol{y}_i$, $i = 1, \cdots, N$. Throughout, we refer to the measurement values $\boldsymbol{y}_i$ at coordinates $\boldsymbol{x}_i$ as samples, and the coordinates $\boldsymbol{v}$ as resampling points. To compute $f(\boldsymbol{v})$ we consider the N data values within a window region set Ω centered on $\boldsymbol{v}$. This set comprises values within an ellipsoidal hyper-surface. We assume each measured value $\boldsymbol{y}_i$ has an associated noise ϵ_i. In this case,

$$f(\boldsymbol{x}_i) + \epsilon_i = y_i, \tag{8}$$

and the task of determining $f(\boldsymbol{v})$ becomes one of estimating the S coefficients c of f from N noisy measurements. Using the above notation we have

$$\mathbf{c} \cdot \boldsymbol{\Phi} + \boldsymbol{\epsilon} = \mathbf{y} \tag{9}$$

which can be expressed in matrix notation as

$$\begin{bmatrix} 1 & \Phi_{(2,1)} & \Phi_{(3,1)} & \cdots & \Phi_{(S,1)} \\ 1 & \Phi_{(2,2)} & \Phi_{(3,2)} & \cdots & \Phi_{(S,2)} \\ 1 & \Phi_{(2,3)} & \Phi_{(3,3)} & \cdots & \Phi_{(S,3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \Phi_{(2,N)} & \Phi_{(3,N)} & \cdots & \Phi_{(S,N)} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_S \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} \tag{10}$$

where $\Phi_{(m,i)}$ is the m^{th} element of the set of Φ for sample point i. We now wish to solve for the set of S coefficients c. The standard procedure is to use the least-squares formalism, by which the estimated values for the coefficients, $\hat{c}$, are given by Fan et al. (1996) as

$$\hat{\mathbf{c}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{y} \tag{11}$$

or, in the case of a weighted fit,

$$\hat{\mathbf{c}} = (\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{W}\mathbf{y} \tag{12}$$

where $\mathbf{W}$ is an NxN diagonal weight matrix in which the diagonal elements are the weights for each measurement point. Once the $\hat{c}$ values have been determined, $f(v)$ can be evaluated:

$$\Phi_s(\boldsymbol{v}) = \prod_{k=1}^{K} v_k^{s_k} \tag{13}$$

$$f(\boldsymbol{v}) = \sum_{s \in S} \hat{c}_s \Phi_s(\boldsymbol{v}) \tag{14}$$

The conditional bias and variance on $\hat{c}$ are given by Fan et al. (1996) as

$$\text{Bias}(\hat{c}|\Phi) = E(\hat{c}|\Phi) - c = (\Phi^T W \Phi)^{-1} \Phi^T W r \tag{15}$$

$$\text{Var}(\hat{c}|\Phi) = C_{\hat{c}} = (\Phi^T W \Phi)^{-1} (\Phi^T \Sigma \Phi)(\Phi^T W \Phi)^{-1} \tag{16}$$

where the residuals on the fit are given as

$$r = \Phi \hat{c} - y \tag{17}$$

and

$$\Sigma = diag(w_\delta^2(x_{i,m}) \sigma_x^2(x_i)) \tag{18}$$

$\sigma_x(x)$ is the conditional variance of y given x, and $w_\delta(x_{i,m})$ is the positional (distance) weighting for sample i with respect to resampling point m (equivalent to the Wt function in equation 1).

2.3. *Error Propagation and Estimation*

The Mean Squared Error (MSE) of a fit at point v is given by Fan et al. (1996) as

$$\text{MSE}\{f(v)\} = (\text{Bias}\{f(\boldsymbol{v}) \,|\, \hat{c}_v\})^2 + \text{Var}\{f(\boldsymbol{v}) \,|\, \hat{c}_v\} \tag{19}$$

and variance on the fit at $\Phi(v)$ may be propagated as

$$\text{Var}\{f(\boldsymbol{v}) \,|\, \hat{c}_v\} = \Phi(v)^T C_{\hat{c}} \Phi(v) \tag{20}$$

We also derive a data driven estimate of the variance-covariance matrix from the residuals on the fit r using

$$\text{Var}\{\hat{c} \,|\, \Phi, r\} = R_{\hat{c}} = \chi_r^2 (\Phi^T W \Phi)^{-1} \tag{21}$$

where χ_r^2 is the reduced chi-squared statistic calculated as

$$\chi_r^2 = \frac{|\Omega|}{tr(W)(|\Omega| - |S|)} r^T W r \tag{22}$$

Note that $|S|$ is the total number of parameters that are being estimated in for the fit, where S is given in equation 5, and $|\Omega|$ are the total number of samples in the "local" region of v, so the quantity $|\Omega| - |S|$ gives the degrees of freedom in our fit. The variance on a fit at v according to residuals is then given as

$$\text{Var}\{f(\boldsymbol{v}) \,|\, r\} = \Phi(v)^T R_{\hat{c}} \Phi(v) \tag{23}$$

3. IMPLEMENTATION

The resampling algorithm takes N samples with coordinates $\boldsymbol{x}$ and generates local polynomial fits at M resampling points with coordinates $\boldsymbol{v}$. In this case, the "local" region for a single resampling point m is defined by a K-dimensional ellipsoid centered on $\boldsymbol{v}_m$. The principle axes of the ellipsoid are given as a user supplied parameter ω so that equation of the ellipsoid centered on point m is given as

$$\sum_{k=1}^{K} \frac{(x_k - v_{m,k})^2}{\omega_k^2} = 1 \tag{24}$$

The window region of point m contains the set of samples Ω_m where sample i is included when

$$i \in \Omega_m \mid \sum_{k=1}^{K} \frac{(x_{i,k} - v_{m,k})^2}{\omega_k^2} \leq 1 \tag{25}$$

This window region will be constant for all resampling points. Therefore, it should be chosen carefully as it defines the subset of samples that are to be considered "local" for the LPR of each resampling point. Although there is no strict limit to the size of the window in each dimension, it should be chosen so that there are sufficient samples to perform a polynomial fit of the desired order in most cases (see section 3.2.2). Since ω is constant, determining ω for measurements consisting of variable sample densities can be difficult. It may be necessary to set ω to a large value in order to ensure fits occur at certain resampling points.

If the instrumental response of the observing device is of importance, then the window size should be chosen to represent a maximum distance for any coupling effects. For example, FIFI-LS reductions use a default window of $3 \times \text{fwhm}_{\bar{\lambda}}$ in the spatial dimensions. If distance weighting is enabled (see section 3.3), and $\sigma_x \ll \omega$, there is no significant consequence for the final result with increased ω. Computationally, however, the window size increases the number of operations by $\mathcal{O}(\omega^K)$ assuming uniform sampling density, and should be given extra consideration in higher dimensional reductions. Figure 1 gives a graphical representation of the window in 3 dimensions.

3.1. *Search Problem*

In many cases, both M and N are large, and deriving Ω for all M resampling points is computationally expensive. To shorten processing time, the reduction is performed in parallel with each thread performing a search on set of samples that are guaranteed to be close to another set of resampling points. This is accomplished by dividing up the sample coordinate space into regularly spaced K-orthotopes (blocks) with the width in dimension k equal to ω_k. Binning resampling points and samples into blocks is a quick and easy procedure. For any given block, the search for samples within the window region of any resampling point in that block should be limited to the block itself, and all immediately neighboring blocks as shown in figure 2.

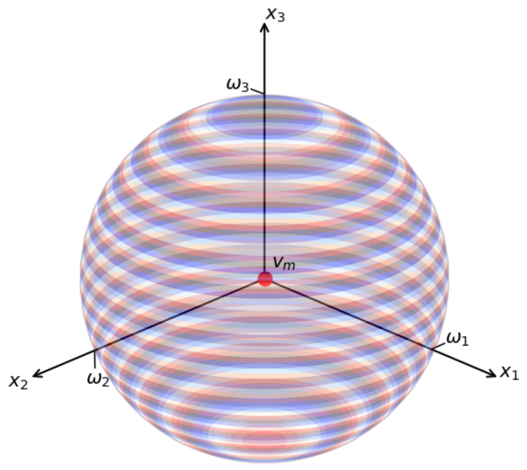


Figure 1. The window region centered around point $\boldsymbol{v}_m$ in 3-dimensions. All samples falling within the shaded region will be included in the fit. The principle axes (ω) of the ellipsoid defining the bounds of the window are supplied by the user.

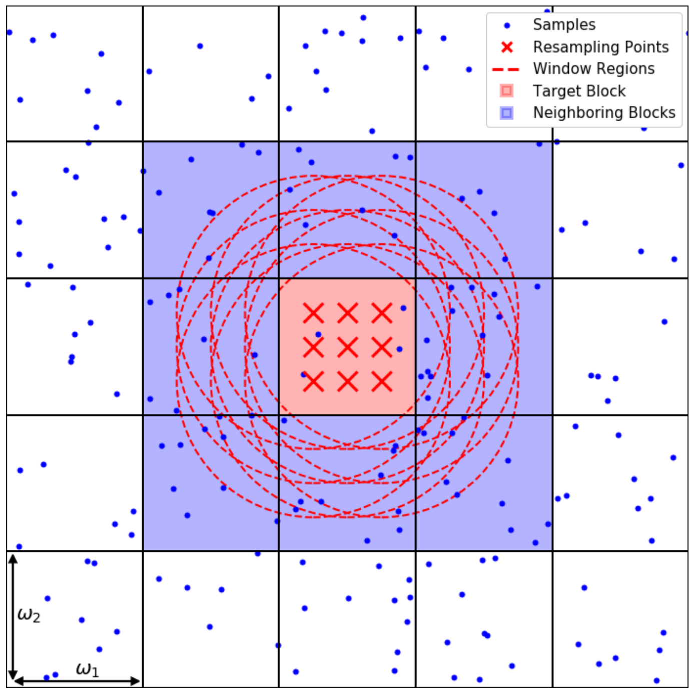


Figure 2. Division of a 2-D sample space into blocks. The search for samples inside the window region of all resampling points within the target block is limited to immediately neighboring blocks.

The procedure used to bin samples and points into blocks simply by assigning an integer index to any given coordinate $\boldsymbol{x}$ in K-dimensions using

$$\text{block}_{i,k} = \left\lfloor \frac{x_{i,k} - min(\boldsymbol{x_k})}{\omega_k} \right\rfloor \tag{26}$$

for coordinate i in dimension k where $\lfloor \cdots \rfloor$ indicates the floored value. All neighboring blocks (the "neighborhood") including the block itself may then be found by simply applying the kernel

$$\text{neighborhood}_{i,k} = \text{block}_{i,k} + [-1, 0, 1] \tag{27}$$

Once the block indices for all samples and points have been found, reductions are performed in parallel over all blocks. Each thread in the parallel reduction is sent a block of resampling points along with its neighborhood of samples as shown in figure 3. Every resampling point in a thread is then processed in serial, first determining which samples in the neighborhood are inside the window region Ω_m of point m, and then deriving a fitted value at that location (figure 4).

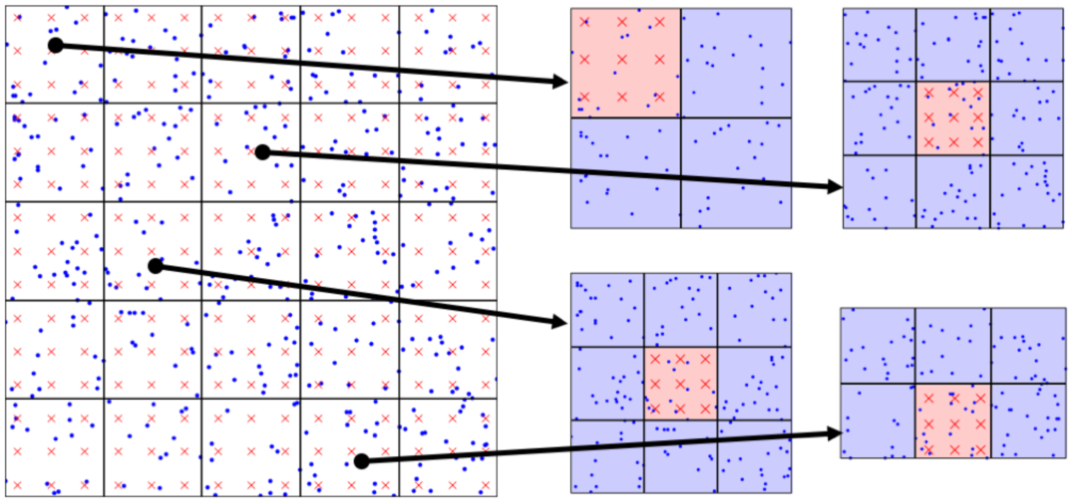


Figure 3. Left: Samples (blue dots) and resampling points (red crosses) divided into blocks for parallel reduction. Right: Four threads of the parallel reduction, each containing a single block of resampling points (red grid squares) and its surrounding neighborhood of samples (blue grid squares).

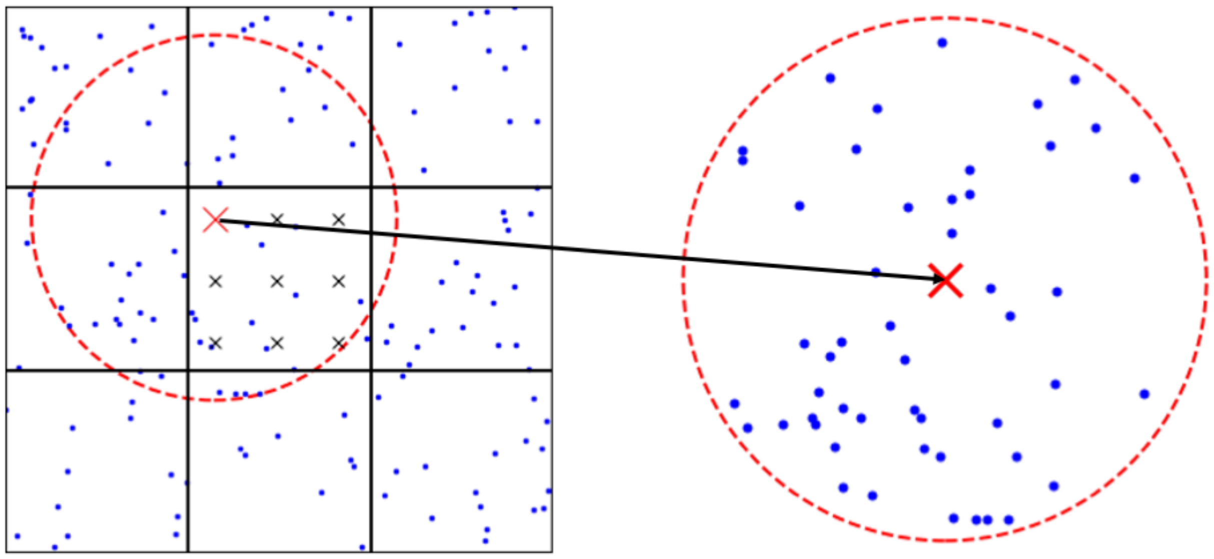


Figure 4. Left: A single thread of the parallel reduction containing a set of resampling points (crosses) and samples (blue dots). Each resampling point will be processed in series. Right: All samples within the window region (Ω_m) of point m at coordinate v_m will be included in the fit.

3.2. *Distribution Checks*

Before deriving a fit from the samples in the window region Ω_m of about coordinate v_m of resampling point m, a number of checks may be performed. These ensure that a polynomial fit of the desired order is possible, and the position of the resampling point relative to the sample distribution will result in a representative fit.

3.2.1. *Edge Rejection*

Polynomial fits (especially higher order polynomials) generally become increasingly less representative of the samples from which they were derived away from the center of the sample distribution. A simple one-dimensional example shown in figure 5 illustrates this deviation.

Away from the center of the distribution ($\bar{x} = 0$), both the deviation and error on the fit begin to increase significantly, and it would be undesirable to attempt a fit in such regions. Therefore, the user may supply an "edge threshold" parameter ($\beta_{edge} > 0$) effectively defining an "edge" around the center of the distribution. No fitting will be permitted for any resampling point located outside of this edge, and the value of the fit for any rejected point will be set to a user-defined failure value.

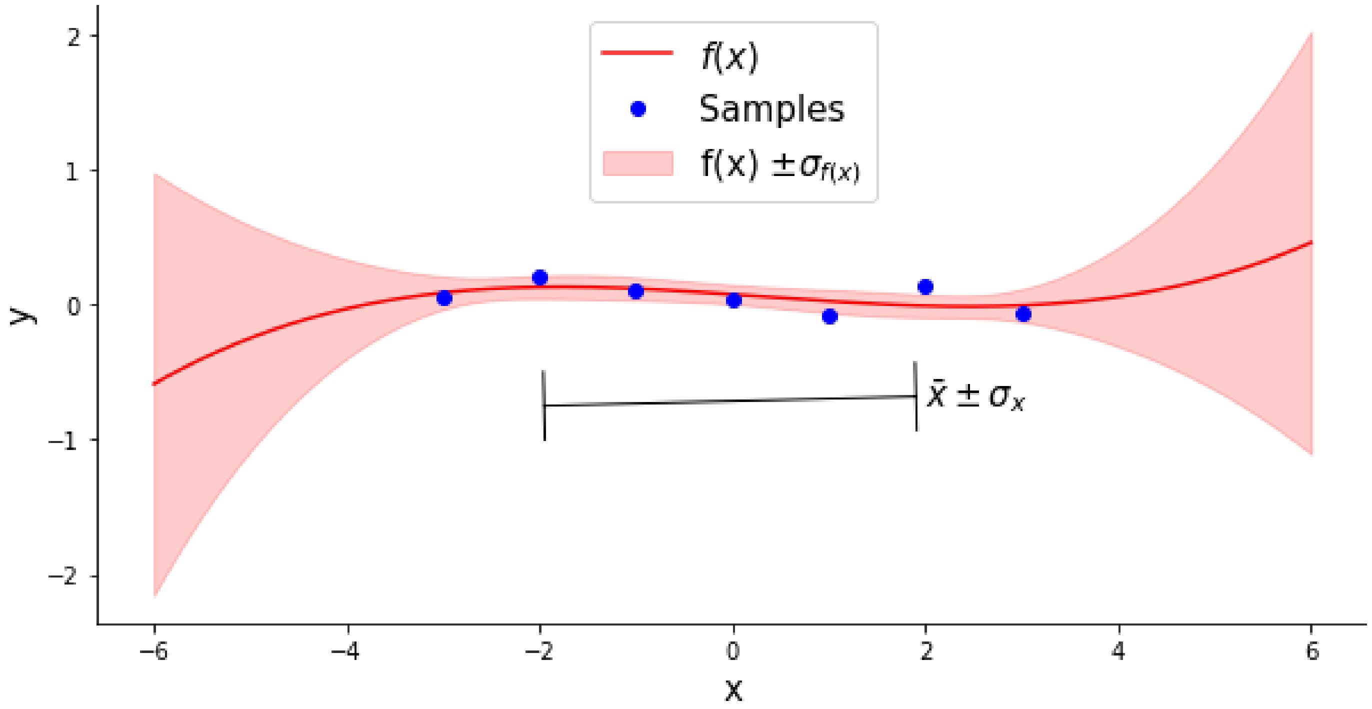


Figure 5. A 3rd order polynomial fit (red) to the samples (blue). The shaded red area is the 1σ error on the fit.

For a single dimension, we define the edge threshold parameter such that

$$f(x) = \begin{cases} \hat{c}_m \cdot \Phi(v_m), & \text{if } |v_m - \bar{x}| \leq \sigma_x/\beta_{edge} \\ \text{Failure value}, & \text{otherwise.} \end{cases} \tag{28}$$

As β_{edge} increases, the acceptable fitting region becomes more concentrated about the center of the distribution ($\bar{\boldsymbol{x}}$). In multiple dimensions, the covariance of the sample distribution ($\boldsymbol{\Sigma_x}$) is used, allowing the additional benefit of preventing fits away from colinear type distributions. For $K > 1$, the following exclusion is applied:

$$f(x) = \begin{cases} \hat{c}_m \cdot \Phi(v_m), & \text{if } \sqrt{(\boldsymbol{v}_m - \bar{\boldsymbol{x}})^T {\Sigma_x}^{-1} (\boldsymbol{v}_m - \bar{\boldsymbol{x}})} \leq \beta_{edge}^{-1} \\ \text{Failure value}, & \text{otherwise.} \end{cases} \tag{29}$$

The point rejection region for a 2-dimensional sample distribution is illustrated in figure 6 using an edge threshold parameter $\beta_{edge} = 0.5$, rejecting any fits occurring at more than $2\sigma_x$ from the center of the sample distribution.

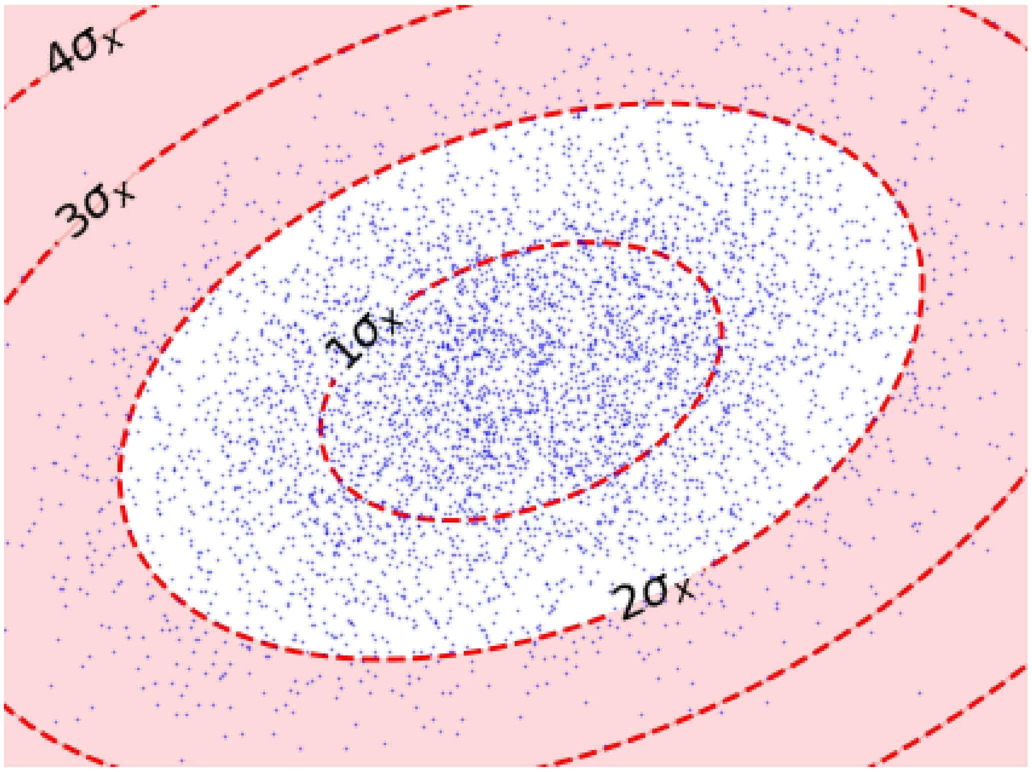


Figure 6. Edge rejection with $\beta_{edge} = 0.5$. No fits will be attempted more than $2\sigma_x$ from the center of the sample distribution (blue dots). The fit exclusion area is shaded red and lines of constant σ_x (Mahalanobis distance) are plotted as red dashed lines.

3.2.2. *Polynomial Order Rejection*

For the terms of a polynomial equation to be linearly independent over each dimension, we require that there be at least n_{order} samples to fit for a given order $\boldsymbol{o}$, where

$$n_{order} = \prod_{k=1}^{K} (o_k + 1) \tag{30}$$

For example, two samples can uniquely define a first order (linear) 1-dimensional polynomial so long as those samples occupy unique coordinates. However, those same two points can be modelled exactly by an infinite number of higher order polynomials. There is currently one mandatory and two optional polynomial order checks. In all cases it is mandatory that

$$|\Omega_m| \geq n_{order} \tag{31}$$

where $|\Omega_m|$ indicates the cardinality (size, or number of samples) of the set Ω_m for point m. If the above condition is not met, the fit is aborted, and a failure value is returned for the value at v_m. In general, we want a "best fit" solution (in a least-squares sense) where $|\Omega_m| > n_{order}$. However, equation 31 only requires that an exact fit is possible. If the user can guarantee that the samples are uniformly distributed, then the procedure may be stopped here to save processing time.

The second step checks the number of unique coordinates in each dimension k such that

$$|\{x_{i,k} \forall i \in \Omega_m\}| > o_k \tag{32}$$

where $|\{\cdots\}|$ gives the cardinality (number of unique values) of the sample coordinates for a given dimension. The above check ensures that we have enough unique terms to allow for matrix inversion in equations 11 and 12. Depending on how the user has defined β_{edge} (see 3.2.1), this may allow for extrapolation at any point within the window.

The final check ensures that there are enough samples to provide for a least-squares best fit solution, and that any fitting occurs within the bounds defined by the samples. For a fit to occur,

$$\begin{aligned} |\{x_{i,k} \forall i \in \Omega_m \,| x_{i,k} < v_{m,k}\}| \geq o_k \\ |\{x_{i,k} \forall i \in \Omega_m \,| x_{i,k} > v_{m,k}\}| \geq o_k \end{aligned} \tag{33}$$

ensuring that $|\Omega_m| \geq 2n_{order}$.

Examples of these three order checking algorithms are shown in figure 7. If the sample distribution does not meet the criteria defined in the order checking algorithm, the returned fit value will be set to a user defined failure value.

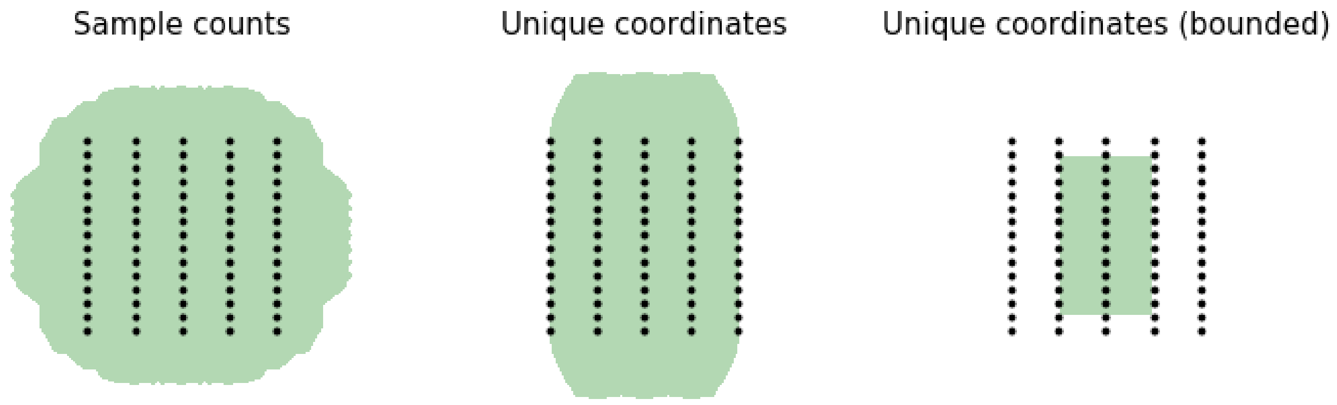


Figure 7. The three available edge checking algorithms shown in order of increasing robustness from left to right. Samples (black dots) are regularly spaced at $\Omega/2$ in the horizontal direction, and $\Omega/6$ in the vertical direction. Regions in which fits may occur are shaded green.

3.3. *Weighting*

Weighted polynomial regression is a method by which some samples in the local polynomial fit have a stronger influence on the solution of the best fit coefficients, as defined in equation 12. Weighting may be based on the measurement error associated with each sample, the position of a sample relative to each resampling point, or a combination of both error and position.

The weighting applied to point m is given as

$$diag(W_m) = w_{\sigma,i} w_\delta(x_{i,m}) \,|\, i \in \Omega_m \tag{34}$$

where Ω_m is the local subset of samples included in the fit for point m. If positional (distance) weighting is not applied, then $w_\delta = 1$, and error weighting is disabled when $w_\sigma = 1$. If no weighting occurs, then $W = I$, where I is an identity matrix of rank $|\Omega_m|$, and the notation $|.|$ indicates the cardinality (size or number of unique values) in the set.

3.3.1. *Error Weighting*

If the 1σ measurement error of sample i is known (σ_{y_i}), the associated weighting factor is:

$$w_{\sigma,i} = \frac{1}{\sigma_{y_i}^2} \tag{35}$$

The following figure (8) shows an example of fitting with and without error weighting.

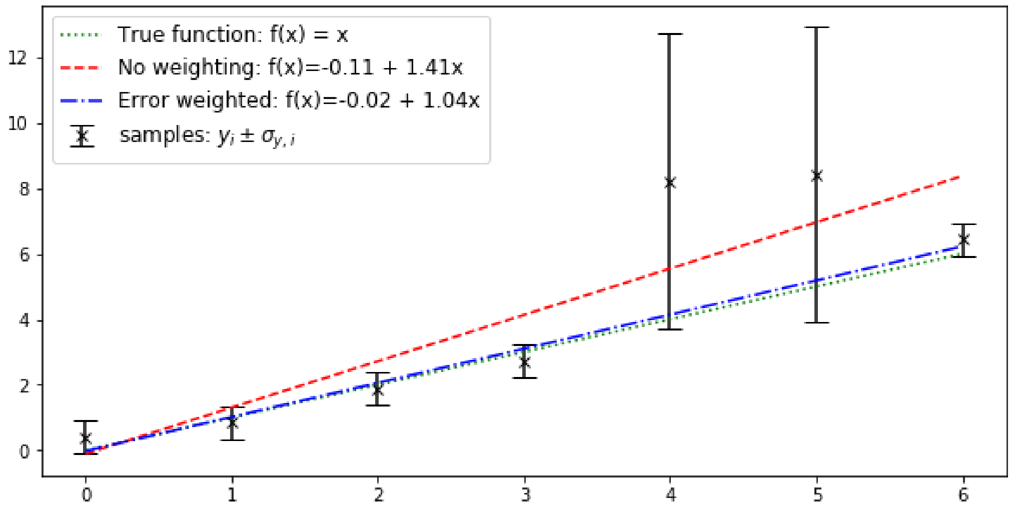


Figure 8. An example of the application of error weighting in 1-dimension. Two of the samples shown with large error bars will skew the fit away from the underlying function if errors are not accounted for.

3.3.2. *Distance Weighting*

Distance weighting applies higher significance to samples that are closer to the resampling point. For the solution to converge effectively, the weighting kernel should be a non-negative, bounded probability function (Gu et al. (2015)). Two popular functions are the Epanechnikov and Gaussian kernels, of which we have opted for the Gaussian function as it has some desirable qualities that are exploited during the adaptive weighting formulation (see section 3.4). The distance weighting applied to a single sample i at coordinate x_i for a local polynomial fit centered on resampling point m at coordinate v_m is given as

$$w_{(\delta,i,m)} = exp\left(-\sum_{k=1}^{K} \frac{(v_{m,k} - x_{i,k})^2}{\alpha_k}\right) \tag{36}$$

where α is a user supplied smoothing parameter which may vary for each dimension (k). As $\alpha \to 0$, closer samples gain increasing influence over the fit, whereas larger α values result in more equal weighting between samples, irrespective of distance. Note that in terms of the bandwidth given in equation 1, $h = \sqrt{\alpha}$. We also frequently use the equation defining a multivariate Gaussian where

$$w_\delta = exp\left(-(v-x)^T A^{-1}(v-x)\right) \tag{37}$$

where $A = diag(2\sigma_w^2) = diag(\alpha)$, and σ_w is the standard deviation of the Gaussian. An example of distance weighting, and the effect of modifying α can be seen in figure 9.

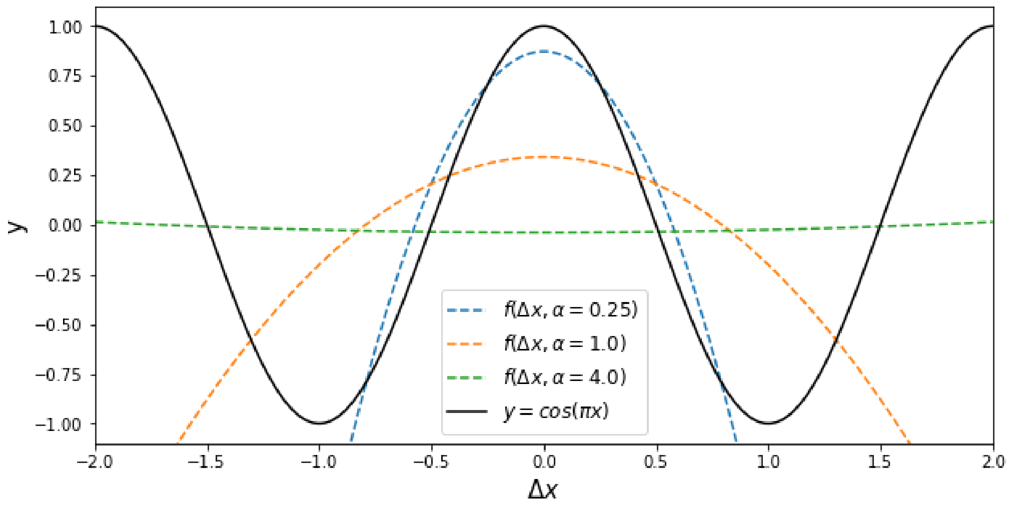


Figure 9. The effect of distance weighting on a second order polynomial fit to samples generated using $y = \cos(\pi x)$. The smoothing parameter (α) can be modified to increase the influence of samples in the fit based on distance from the resampling point (Δx).

3.4. *Adaptive Weighting*

If the data we wish to model can be easily represented by a smooth function, then standard distance weighting may produce an acceptable fit. However, when the underlying function contains a mix of structures of varying complexity, a single distance weighting parameter may not be adequate. This is alleviated somewhat by allowing the bandwidth parameter h in equation 1 to vary at each resampling point, and is extensively discussed by Fan et al. (1996) and Masry (1996) amongst others.

Most methods seek to find the optimal smoothing parameter by minimizing the mean squared error given in equation 19, balancing the leading bias and variance. In the case where we have supplied measurement errors, this does not always result in what one may consider an "optimal fit". This is especially true in the case of FIFI-LS where minimizing the overall MSE results in highly spurious fits. An alternative presented here, defines the optimal fit as one in which the reduced chi-squared statistic (χ_r^2), given in equation 22, is equal to one. When the measurement error is known, $\chi_r^2 > 1$ indicates a poor fit which does not adequately represent the data. Conversely, a fit where $\chi_r^2 < 1$ is over-fitting the data, and can be thought of as fitting to the noise. Therefore, our goal in determining an optimal smoothing parameter is to get $\chi_r^2 \to 1$, rather than minimizing the MSE, which could likely result in $\chi_r^2 \ll 1$.

We accomplish this by performing an initial test reduction using both distance and error weighting where

$$W = W_\delta W_\sigma = diag(w_{\delta,i} w_{\sigma,i}) = diag(\frac{w_{\delta,i}}{\sigma_{y,i}^2}) \tag{38}$$

and propagate the theoretical and stochastic variance as defined in equations 20 and 23. We assume that the two measurements are related by

$$\chi_r^2 = \frac{\mathrm{Var}\{f(\boldsymbol{v} \,|\, \hat{c}_v)\}}{\mathrm{Var}\{f(\boldsymbol{v} \,|\, r)\}} \tag{39}$$

Now, for a set of n error-normalized observations the unbiased sample variance is given by

$$S_{n-1}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (Z_i - \bar{Z})^2 \tag{40}$$

where we set

$$Z = \frac{y - \hat{y}}{\sigma_y} \tag{41}$$

From Mood (1974) we know that the mean squared error is

$$\begin{aligned} \mathrm{MSE}(S_{n-1}^2) &= \frac{1}{n} \left(\mu_4 - \frac{n-3}{n-1} \sigma_f^4 \right) \\ &= \frac{1}{n} \left(\beta_2 + \frac{2n}{n-1} \right) \sigma_f^4 \end{aligned} \tag{42}$$

where μ_4 is the fourth moment of the distribution, and β_2 is the excess kurtosis, also given by

$$\beta_2 = \frac{\mu_4}{\sigma_f^4} - 3 \tag{43}$$

Clearly, in this case $\mathrm{MSE}(S_{n-1}^2) \approx \chi_r^2$ when $n \gg 1$, and if we make the strong assumption that excess kurtosis is independent of σ_f, then from equation 42 we can write

$$\chi_r^2 = \beta_s^2 \sigma_f^4 \tag{44}$$

where β_s is a constant. Since σ_y (the error in the measured sample values) was supplied by the user and is assumed to be correct, we allow σ_f to represent an error factor introduced by the weighting function. A representative value for σ_f can be derived by integrating equation 37 as

$$\sigma_f = \int_{R^K} w_\delta = \left(\frac{\pi}{2} \right)^{K/2} \det(A)^{1/2} \tag{45}$$

Therefore,

$$\chi_r = \beta_s \frac{\pi}{2} \det(A) \tag{46}$$

with $\chi_r = \sqrt{\chi_r^2}$. β_s can then be determined from a suitable test reduction, and a new weighting function can be defined such that $\chi_r \to 1$. To accomplish this, and using our assumption that β_s is constant, we set

$$\det(A_{\text{new}}) = \det(A)/\chi_r \tag{47}$$

3.4.1. *Test Reduction*

One fundamental assumption made during the adaptive weighting algorithm is that the MSE is dependent on the excess kurtosis introduced by the weighting function (see equation 42). This approximation can only be relied upon when $n \gg 1$, and we therefore need to select an initial kernel that can fulfill this requirement. However, if the kernel is too large such that a low order polynomial could not possibly represent the expected structure in the data, a misleadingly high excess kurtosis would be reported due to an excess of outliers.

In astronomical observations (for which this algorithm was devised), the Full-Width-Half-Maximum (FWHM) for the point spread function of the observing system is usually known, at least approximately. If so, we know the uncertainty in the positional measurements is given as

$$\sigma_x = \frac{\text{FWHM}_x}{2\sqrt{2\ln 2}} \tag{48}$$

To effectively nyquist sample the expected structures present in data retrieved from an instrument with a Gaussian response function we set

$$\begin{aligned}\sigma_{\text{test}} &= \frac{\pi}{\sqrt{2\ln 2}}\sigma_x \\ &= \frac{\pi}{4\ln 2}\text{FWHM}_x\end{aligned} \tag{49}$$

We can then define the test distance weighting kernel as

$$A_{\text{test}} = diag(2\sigma_{\text{test}}^2) = diag(\alpha_{\text{test}}) \tag{50}$$

There is one other important difference to the standard reduction: the user supplied resampling points (coordinates at which fits are required) are completely ignored, and instead replaced with the sample coordinates themselves. In other words, a fit is performed at each sample coordinate generated from the values of all samples within its window region (see figure 10).

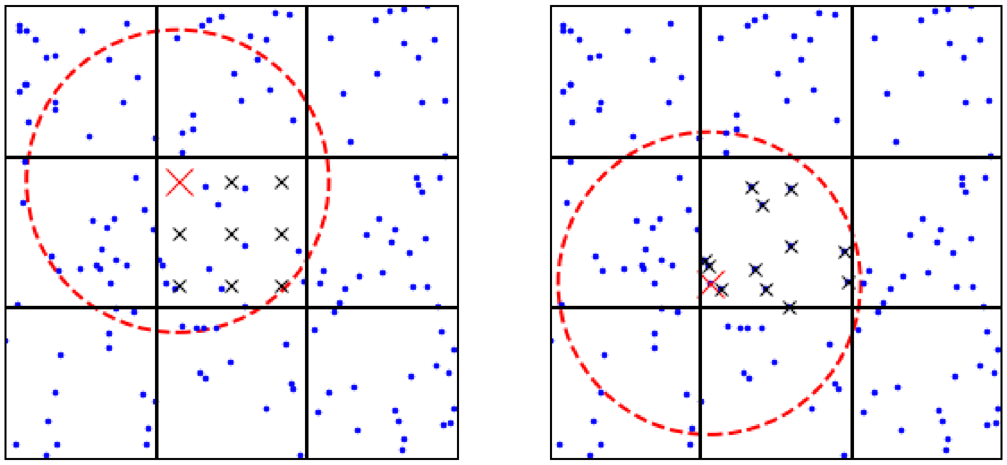


Figure 10. Left: A standard reduction on a single block. Samples are shown as blue dots, and resampling points are shown as red crosses with the window region of a single point marked as a dashed red line. Right: A test reduction for determining the adaptive weighting kernels where resampling points are equal to the sample coordinates.

3.4.2. *Adaptive Weighting Kernel*

A counter-intuitive consequence is that positional weighting kernels are derived for each sample rather than at each resampling point. This is extremely beneficial when repeatedly resampling the same sample set onto different resampling points as the weighting kernels only need to be calculated once. Once generated, these kernels allow for almost real-time interactive analysis of a large data set over a modestly sized region of interest. Kernels are also self-consistent, dependent only on the samples themselves rather than the resampling points defined by the user.

The final distance weighting applied to sample i for a fit at point m is

$$w_(\delta, i, m) = exp\left(-(v_m - x_i)^T A^{-1}_{\text{test},i}(v_m - x_i)\right) \tag{51}$$

where A_i is the adaptive weighting kernel (matrix of rank K), derived for each sample i, $i = 1, \cdots, N$. There are currently two adaptive weighting algorithms: "scaled" and "shaped". The scaled algorithm applies a scaling factor to α_{test}, expanding or contracting the width of the kernel by an equal factor across all dimensions. The second algorithm (shaped) is an extension of the first, available when $K > 1$, and allows the kernel to stretch and rotate about a given sample.

3.4.2.1. *Adaptive Kernel Scaling*—During the test reduction, a fit is performed at each sample coordinate x_i derived from all samples within the window region Ω_i. We use a test smoothing parameter α_{test}, defining the test weighting kernel as

$$A_{\text{test}} = diag(\alpha_{\text{test}}) \tag{52}$$

so that the scaled weighting kernel (A_{scaled}) will satisfy

$$det(A_{\text{scaled, i}}) = \frac{det(A_{\text{test}})}{\chi_{r,i}} \tag{53}$$

Since the scaled algorithm merely applies a global scaling factor to the test diagonal kernel, we can write

$$\begin{aligned} det(A_{\text{scaled}}) &= \frac{det(A_{\text{test}})}{\chi_r} \\ \prod_{k=1}^{K} \alpha_{\text{scaled}} &= \frac{1}{\chi_r} \prod_{k=1}^{K} \alpha_{\text{test}} \\ \alpha_{\text{scaled}} &= \chi_r^{-\frac{1}{K}} \alpha_{\text{test}} \end{aligned} \tag{54}$$

3.4.3. *Adaptive Kernel Shaping*

The shaped adaptive weighting algorithm allows the overall kernel scale to vary according to the scaled method described above, and allows the principle axes of the kernel to stretch and rotate about each sample. Note that this only has meaning in two dimensions or more ($K > 1$). When attempted for $K = 1$, the shaped kernel is equivalent to the scaled kernel.

To first order, variance on the fit may be propagated as

$$\begin{aligned} Var(y) &= J^T Var(x) J \\ &= V_x J^T I J \\ &= V_x J^T J \end{aligned} \tag{55}$$

where $Var(x)$ is the variance-covariance matrix of the sample measurements. We assume that $Var(x)$ is dimensionally independent, and may be represented by the scalar V_x as a function of measurement error only. Here, J is the Jacobian matrix of the fitting function given as

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_K} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_K} \end{bmatrix} \tag{56}$$

where $\frac{\partial f_i}{\partial x_k}$ is the partial derivative of the function for sample i with respect to dimension k. Derivatives of the polynomial function are easily determined using a subset of the Φ terms (see equation 7). The set of polynomial terms defined in equation 5 guarantees that all necessary derivative terms are present so long as the polynomial order is ≥ 1. For example, consider a second order polynomial fit in two dimensions

$$\begin{aligned} f(\boldsymbol{x}) &= c_1 + c_2 x_0 + c_3 x_0^2 + c_4 x_1 + c_5 x_0 x_1 + c_6 x_1^2 \\ \frac{\partial f}{\partial x_0} &= c_2 + 2c_3 x_0 + c_5 x_1 \\ \frac{\partial f}{\partial x_1} &= c_4 + c_5 x_0 + 2c_6 x_1 \end{aligned}$$

In terms of Φ, these become

$$\begin{aligned} f(\Phi) &= c_1\Phi_1 + c_2\Phi_2 + c_3\Phi_3 + c_4\Phi_4 + c_5\Phi_5 + c_6\Phi_6 \\ \frac{\partial f}{\partial x_0} &= c_2\Phi_1 + 2c_3\Phi_2 + c_5\Phi_4 \\ \frac{\partial f}{\partial x_1} &= c_4\Phi_1 + c_5\Phi_2 + 2c_6\Phi_4 \end{aligned}$$

Since all Φ terms are precalculated, calculating derivatives is a fast and computationally cheap operation. Once $f'(x_j)\,\forall\, j \in \Omega_i$ has been evaluated, we create the $(K, |\Omega_i|)$ matrix of weighted partial derivatives for J^TJ from all samples in Ω_i by defining

$$g_i = \frac{\partial f_j}{\partial x_k}\,,\ \forall\, k = \{1, 2, \cdots, K\}\,,\ \forall\, j \in \Omega_i \tag{57}$$

and define the derivative mean-squared-cross-products (MSCP) as

$$J^TJ = \frac{1}{tr(WW^T)} g^T W W^T g \tag{58}$$

where W is the weight matrix defined in equation 34. This results in the (K, K) matrix

$$J^TJ = \begin{bmatrix} \frac{\partial \bar{f}}{\partial x_0}\frac{\partial \bar{f}}{\partial x_0} & \frac{\partial \bar{f}}{\partial x_0}\frac{\partial \bar{f}}{\partial x_1} & \cdots & \frac{\partial \bar{f}}{\partial x_0}\frac{\partial \bar{f}}{\partial x_K} \\ \frac{\partial \bar{f}}{\partial x_1}\frac{\partial \bar{f}}{\partial x_0} & \frac{\partial \bar{f}}{\partial x_1}\frac{\partial \bar{f}}{\partial x_1} & \cdots & \frac{\partial \bar{f}}{\partial x_1}\frac{\partial \bar{f}}{\partial x_K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \bar{f}}{\partial x_K}\frac{\partial \bar{f}}{\partial x_0} & \frac{\partial \bar{f}}{\partial x_K}\frac{\partial \bar{f}}{\partial x_1} & \cdots & \frac{\partial \bar{f}}{\partial x_K}\frac{\partial \bar{f}}{\partial x_K} \end{bmatrix} \tag{59}$$

At this point we could define the normalized shape matrix as

$$g = \frac{1}{\det(J^TJ)^{1/K}} (J^TJ)^{-1} \tag{60}$$

where $\det(g) = 1$. This normalization reflects that we are only interested in the shape of the kernel, and that the overall scaling will still be handled according to section 3.4.2.1. However, highly elongated kernels could result in poor fits for certain sample distributions. Additionally, if $\chi_r^2 \approx 1$ for the test fit, then we really do not want to shape the kernel when it is already optimal. Therefore, we provide a correction to the stretch of the kernel based on number of factors. Singular value decomposition (SVD) is used to factorize the normalized g matrix into

$$USV^T = \mathrm{SVD}(g) \tag{61}$$

Note that this will only be attempted if g has full rank, and if $\det(g) \neq 0$. If not, determination of the kernel shape will be aborted, but the kernel will still be scaled accordingly with $g = I$.

The singular values S represent the relative magnitudes of the partial derivatives in each dimension where $\prod_{k=1}^{K} S_k = 1$, and the unitary $(X^TX = I)$ U and V matrices apply rotation and/or reflection.

Therefore, the SVD of the gradient matrix can be thought of as an initial rotation (V), followed by a stretch (S) and a final rotation (U). To correct for stretch, we modify g as follows:

$$\bar{g} = US^{\gamma}V^{T} \tag{62}$$

where γ is a real-valued ($-1 \leq \gamma \leq 1$) stretch correction factor. Note that for any value of γ, $\det(\bar{g}) = 1$. i.e., $\bar{g}$ will always be normalized and not introduce any global scaling factors into the kernel. If $\gamma = 0$ the stretch along each dimensional axis will be equal, and consequently, rotation is irrelevant so that the solution is equivalent to the scaled solution. If $\gamma = 1$ the shape remains unchanged such that $\bar{g} = g$. Finally, if $\gamma = -1$, the kernel is effectively rotated so that it is perpendicular to maximum gradient direction. As $\gamma \rightarrow 0$ from either direction, the kernel becomes increasingly symmetrical. The effect of γ on the shape kernel is shown in figure 11.

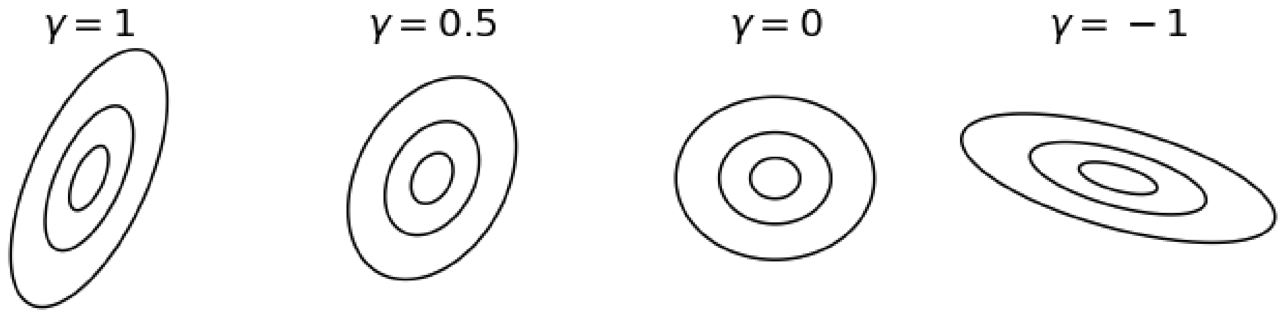


Figure 11. The effect of γ on the shape kernel. The unaltered shape is returned when $\gamma = 1$, symmetrical when $\gamma = 0$, and stretch is reversed when $\gamma = -1$.

Now all that remains is the task of determining γ, which is dependent on three factors: χ_r^2, the density profile of samples in the window region (ρ), and the deviation of the center of the window from the mean sample distribution ($\sigma_{\bar{x}}$). The function $\gamma = f(\chi_r^2, \rho, \sigma_{\bar{x}})$ must satisfy

$$\begin{aligned} \lim_{\chi_r^2 \to \infty} \gamma &= 1 \\ \lim_{\chi_r^2 \to 0} \gamma &= -1 \\ \lim_{\rho \to 0} \gamma &= 0 \\ \lim_{\sigma_{\bar{x}} \to \infty} \gamma &= 0 \end{aligned} \tag{63}$$

One such function that can fulfill these requirements is the logistic function; specifically, a modified version of the Richard's curve.

$$\gamma = \frac{2}{\left(1 + (2^{exp(\sigma_{\bar{x}})} - 1)exp(\rho(1 - \chi_r^2))\right)^{1/exp(\sigma_{\bar{x}})}} - 1 \tag{64}$$

Some properties to note are that:

1. The sign of γ is dependent on χ_r^2 and is negative for $0 \leq \chi_r^2 < 1$, positive for $\chi_r^2 > 1$, and sets $\gamma = 0$ at $\chi_r^2 = 1$.
2. The growth rate increases with ρ.
3. The asymptotes are at $\gamma = \pm 1$, with the point of inflection (point of maximal growth) set by $\sigma_{\bar{x}}$. The point of inflection is set at $\chi_r^2 = 1$ when $\sigma_{\bar{x}} = 0$, and approaches the upper asymptote as $\sigma_{\bar{x}}$ increases.

Figure 12 shows how γ varies with χ_r^2, and the effects of ρ and $\sigma_{\bar{x}}$.

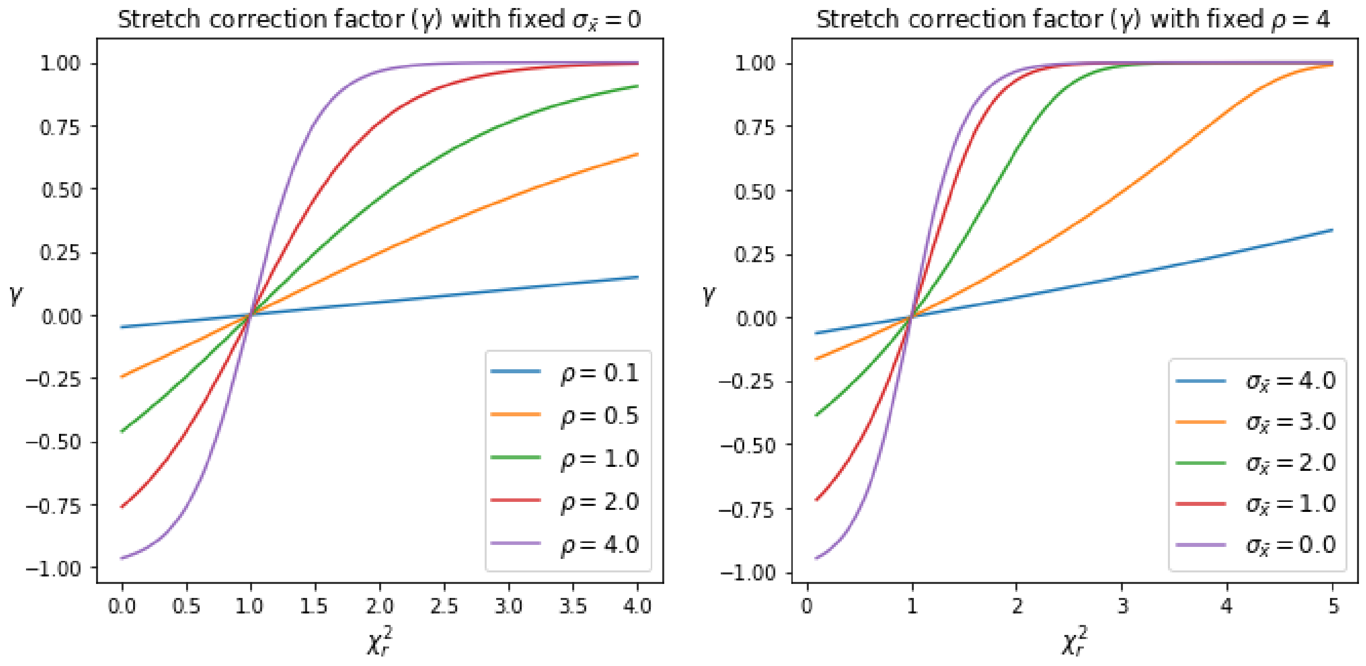


Figure 12. The stretch correction factor (γ) as a function of χ_r^2. Left: The effect of relative density (ρ) on the shape of the function with fixed $\sigma_{\bar{x}}$. Right: The effect of window deviation from the mean sample distribution ($\sigma_{\bar{x}}$) with fixed ρ.

3.4.3.1. *Relative Density*—The local relative density (ρ) of samples is required to calculate the factor γ in equation 64 when determining the shaped adaptive weighting kernel. This is a measure such that $\rho = 1$ when samples are uniformly distributed, $0 \leq \rho < 1$ when a fit occurs in a local depression of the sample density, and $\rho > 1$ when samples are clustered around the fit point. The local relative density is given as:

$$\rho = \frac{\rho(\text{measured})}{\rho(\text{uniform})} \tag{65}$$

The expected uniform density of samples inside a window region with principle axes given by ω_k in dimension k is

$$\rho(\text{uniform}) = N \frac{\Gamma\left(1 + \frac{K}{2}\right)}{\pi^{K/2} \prod_{k=1}^{K} \omega_k} \tag{66}$$

The measured density inside the window is

$$\rho(\text{measured}) = \frac{\sum_{i=1}^{N} w_{\delta,i}}{\int_{R\Omega} w_\delta(\mathbf{\Delta x})} \tag{67}$$

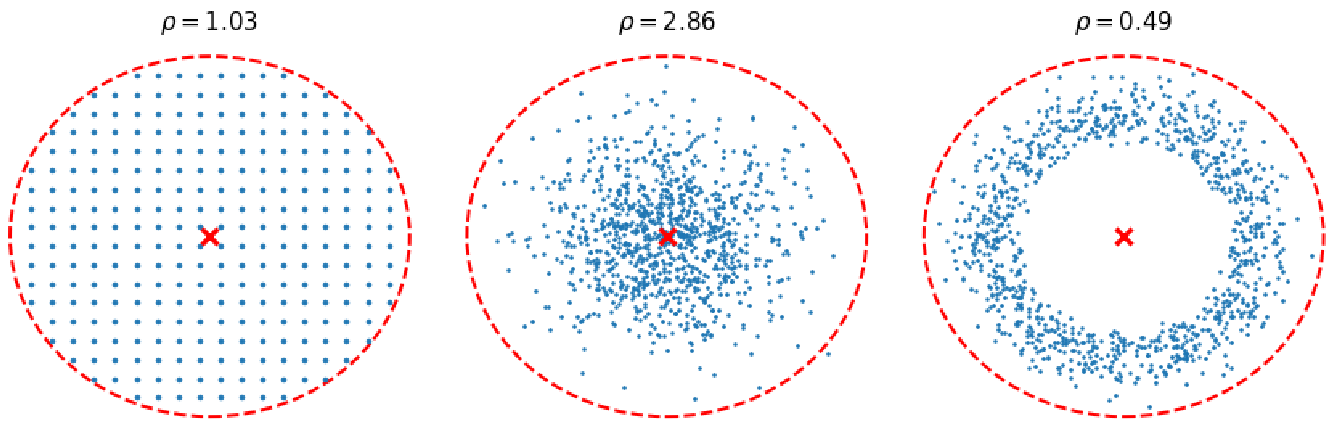


Figure 13. Relative density values for different types of sample distributions. The center of each window region (red dashed circle) is marked with a red cross, and samples are marked as blue dots. All plots were generated using $\omega = 1$ and $\alpha = 0.3$. Left: uniform sample distribution should give $\rho = 1$. In this case, the discrete nature of a small number of measurements leads to a slight deviation from the optimal value. Middle: samples clustered near the center of the window yield $\chi_r^2 > 1$. Right: $\chi_r^2 < 1$ when the window centeroccurs in a local minimum of the sample density.

where $\Delta x = \boldsymbol{v} - \boldsymbol{x}$ gives the relative position of samples to the center of the window, and $\int_{R\Omega} w_\delta(\mathbf{\Delta x})$ is the integral of the distance weighting function (given in equations 36 and 51) over the window region in K dimensions. This integral cannot be solved exactly, and thus, we resort to computationally expensive numerical methods (although "computationally expensive" is relative in this sense, and the operation takes only a small fraction of a second on most laptops). Fortunately, this only needs to be solved once as the test reduction uses a single weighting function due to constant ω for all sample fits. However, if one were to take an iterative approach to determining the optimal kernel shape, this integral would need to be evaluated for each sample, as each weighting function could be unique following an initial test reduction. If this approach was taken, then it would be recommended to either disregard density by setting $\rho =$ constant in equation 64, or using the result obtained from the initial test reduction. Examples of local density values for several types of sample distribution can be seen in figure 13.

3.4.3.2. *Window Offset Deviation*—The window offset deviation ($\sigma_{\bar{x}}$) that is applied in equation 64 is given as

$$\sigma_{\bar{x}}^2 = (\boldsymbol{v} - \bar{\boldsymbol{x}})^T \Sigma_x^{-1} (\boldsymbol{v} - \bar{\boldsymbol{x}}) \tag{68}$$

for a distribution of N samples with coordinates $\boldsymbol{x}$ with mean position $\bar{\boldsymbol{x}}$. The window region is centered on $\boldsymbol{v}$ where the covariance of the sample distribution between dimensions i and j is given as

$$\Sigma_{x_{i,j}} = \frac{1}{N-M} \sum_{n=1}^{N} (\bar{\boldsymbol{x}}_{n,i} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{n,j} - \bar{\boldsymbol{x}}_j) \tag{69}$$

Here, M are the number of degrees of freedom lost when determining the center of the sample distribution $\bar{\boldsymbol{x}}$. M is typically set to 1 when a simple mean operation is performed. Figure 6, in addition to displaying edge rejection thresholds, also shows lines of constant $\sigma_{\bar{x}}$ for a normally distributed set of samples.

3.4.3.3. *Shaped Adaptive Kernel*—Now all parameters required for the derivation of the shaped adaptive weight matrix have been derived, all that remains is to define the kernel itself.

$$A_{\text{shaped}} = \left(\frac{\det(A_{\text{test}})}{\chi_r} \right)^{1/K} \hat{g} \tag{70}$$

Note that once again, $\det(A_{\text{test}}) / \det(A_{\text{shaped}}) = \chi_r$. Unlike the scaled solution, the shape of the kernel depends only on $\hat{g}$ and does not propagate the shape of A_{test}. Therefore, since all information on the shape of A_{test} is lost, it is recommended that the user sets the ratio of ω/σ_x to be equal for all dimensions in which adaptive weighting is enabled (see section 3.4.3.4), or to otherwise use the scaled weighting kernel.

As a brief graphical example, the scaled and shaped kernels are generated from an image (samples are pixels) and then plotted for comparison in figure 14. Note that the kernels are large in smooth regions, and small in regions containing complex structure and edges. The area of a shaped and scaled kernel on a single sample will always be equal. However, the shaped kernels are truncated in the direction of the gradient, and will preserve edges to a greater degree.

3.4.3.4. *Partial Adaptive Weighting for Select Dimensions.*—The size and/or shape of the kernel may be fixed in certain dimensions and allowed to vary in others. To accomplish this, the number of fixed and adaptive enabled dimensions as set to K_{fix} and K_{adapt} where $K_{\text{fix}} + K_{\text{adapt}} = K$, and the sets of dimensions allocated to each are defined as k_{fix} and k_{adapt}. For the test reduction, the smoothing parameter is set to

$$\alpha_{\text{test, k}} = \begin{cases} \alpha_k, & \text{if } k \in k_{\text{fix}} \\ \alpha_{\text{test},k}, & \text{if } k \in k_{\text{adapt}} \end{cases} \tag{71}$$

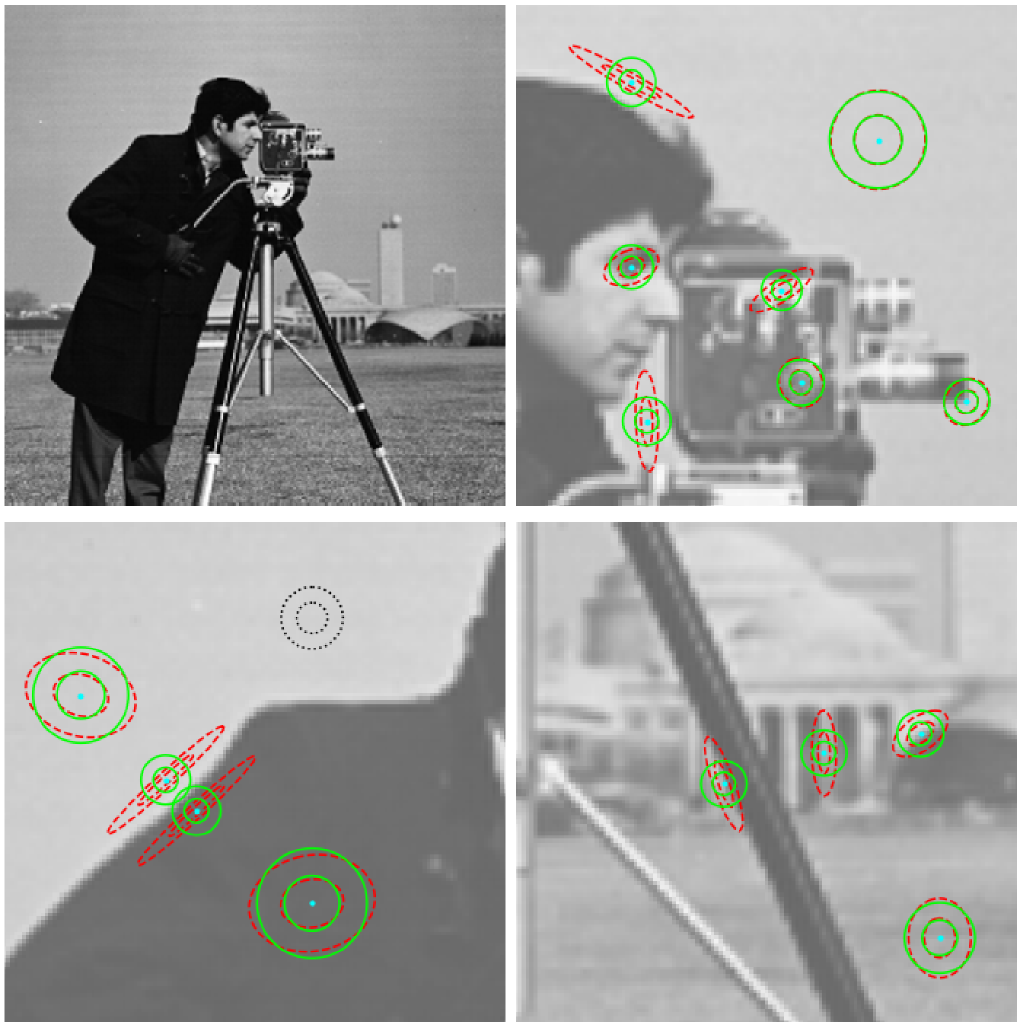


Figure 14. Comparison of the scaled (solid green) and shaped (dashed red) kernels. The upper-left image was used as an input for the test reduction where each pixel used as a sample. The bottom-left image contains a black-dotted kernel, representing the test kernel with smoothing parameter α_{test}. Kernels are relatively large in smooth regions and smaller in regions containing structure. Note that the shaped kernels are stretched perpendicular to the principle gradient direction, and will preserve edges well in subsequent reductions. The data for this image was taken from skimage.data.camera (see scikitimage (2014)).

All calculations are performed as described in sections 3.4.2.1 and 3.4.3.3 with a few important changes:

The final scaled adaptive kernel is given as

$$A_{\text{scaled},(i,j)} = \begin{cases} \alpha_i, & \text{if } i = j \text{ and } i \in k_{\text{fix}} \\ 0, & \text{if } i \neq j \\ {\chi_r}^{-1/K_{\text{adapt}}} \alpha_{\text{test}}, & \text{if } i = j \text{ and } i \in k_{\text{adapt}} \end{cases} \tag{72}$$

For the shaped kernel, a modification is made to $\hat{g}^2$ (following the stretch correction) so that

$$\hat{g}_{i,j} = \begin{cases} 1, & \text{if } i = j \text{ and } i \in k_{\text{fix}} \\ 0, & \text{if } i \neq j \text{ and } i \vee j \in k_{\text{fix}} \end{cases} \tag{73}$$

and the final shaped adaptive kernel is given as

$$A_{\text{shaped},(i,j)} = \begin{cases} \alpha_i, & \text{if } i = j \text{ and } i \in k_{\text{fix}} \\ 0, & \text{if } i \neq j \text{ and } i \vee j \in k_{\text{fix}} \\ \left(\frac{\prod_{k \in k_{\text{adapt}}} \alpha_{\text{test},k}}{\det(\hat{g}) \chi_r} \right)^{\frac{1}{K_{\text{adapt}}}} \hat{g}_{i,j}, & \text{if } i \wedge j \in k_{\text{adapt}} \end{cases} \tag{74}$$

Consideration must be given to the fact that for high K reductions where $K_{\text{adapt} \to 1}$, the change in size of a kernel along an adaptive-enabled dimension becomes increasingly severe as it must accommodate the necessary change in overall kernel volume based on χ_r^2. The effect of fixing certain dimensions in the adaptive weighting calculation can be seen in figure 15.

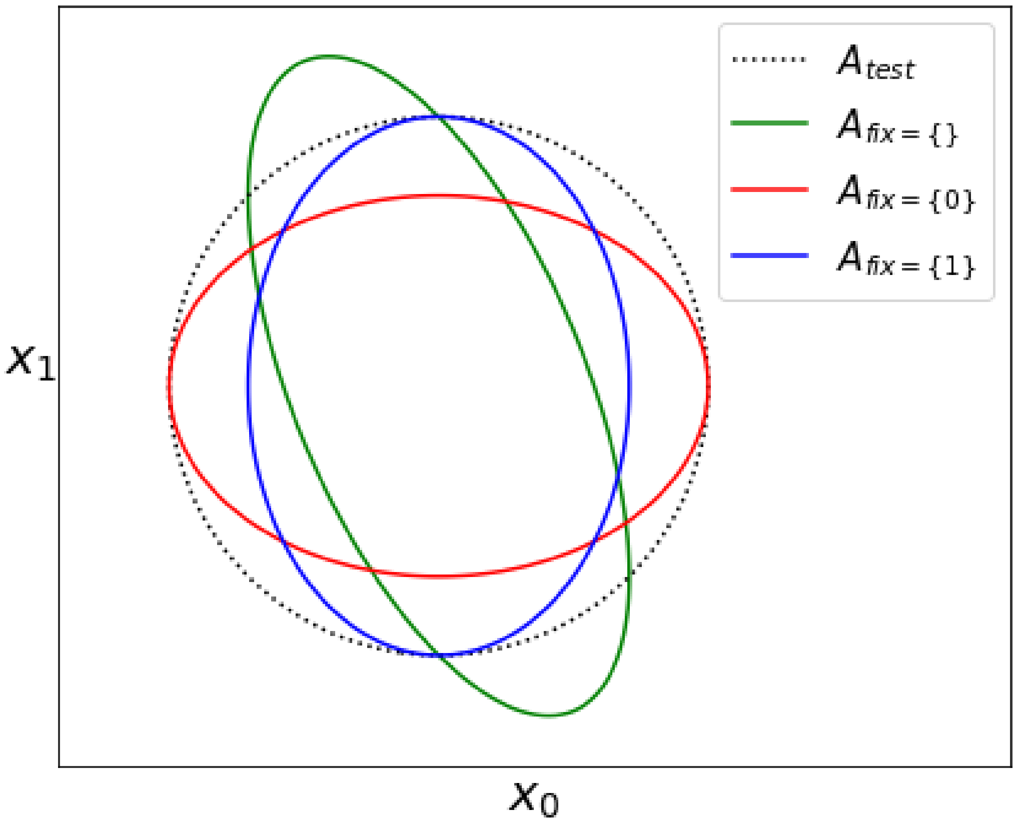


Figure 15. Fixing the smoothing parameter in a single dimension of the shaped adaptive kernel. In this example, $\chi_r^2 = 4$ and an arbitrary derivative MSCP of $[[3, 1], [1, 1]]$ was supplied with minimal stretch correction ($\rho \gg 1$). A single line of constant weighting is drawn for each kernel, with A_{test} shown as a dotted black circle with $\alpha_{\text{test},0} = \alpha_{\text{test},1}$ (circular). In all cases, the ratio of kernel determinants is equal to $(\det(A_{\text{test}}) / \det(A_{\text{fix}}) = \sqrt{\chi_r^2})$.

4. CODE ARCHITECTURE

4.1. *Language and packages*

The resampling code is written in Python [2] (see Rossum (1995)) version 3. Python is a dynamic open source programming language supported by a vast ecosystem of specialized packages of which we rely on only a few: Data such as the sample values, errors, Φ, etc., are stored as Numpy arrays (Oliphant (2006)) and processed using JIT (Just-In-Time) compiled Numba functions (Lam and Pitrou (2015)). Numba was instrumental in performing multidimensional resampling in a reasonable time frame since Python is notoriously slow when compared to other languages, and allows for operations to be performed on a time scale comparable with the C language. The balltree algorithm (Omohundro (1989)) from the Scikit-learn package (Pedregosa et al. (2011)) is used to quickly determine which samples are inside the window region of a resampling point (see figure 4). Finally, parallel processing is handled by the Joblib [3]. The resampling code exists as a submodule of the toolkit module in the SOFIA data reduction pipeline package (REDUX) that is not yet publicly available. However, efforts are being made to release the REDUX package within the year.

4.2. *Classes and Structure*

Figure 16 gives a high level representation of the various classes and process flow. The first step is to create an instance of the Resampler class, initialized with some parameters that must include the window defining the "local" region (ω), the polynomial fit order, and the data we wish to resample. The user may interact with the Resampler through some type of interface. Nearly all examples in this paper were generated from the Python command line and Jupyter notebooks, although some FIFI-LS images were created using the SOFIA data reduction pipeline (REDUX).

During the instantiation of the Resampler object, the sample coordinates (X) are used to initialize a Tree object that allocates samples into blocks and their associated neighborhoods as described in section 3.1 as a function of the window region ω. The $\Phi(x)$ terms (see section 2.1) are also created at this stage and stored as a Tree attribute.

Once the Resampler object has been created, a reduction may be started by supplying a set of reduction parameters (such as weighting schemes) and a set of coordinates (resampling points) containing the desired locations for the fitted output values. Another Tree object is created by the reduction manager from the resampling point coordinates (V), allocating blocks and creating a set of $\Phi(V)$ terms.

Each intersection of a reduction Tree block with a sample Tree neighborhood is than passed in parallel from the reduction manager to the main engine of the resampling algorithm (the box labelled "Fit" in figure 16). For a single block of resampling points, the sample sets Ω for each point are derived simultaneously from the sample neighborhood using the balltree algorithm. A fit at each point is processed in series before being passed back to the reduction manager where they are aggregated. Once all parallel reductions are complete, the results become available to the Resampler where they are either used to update the reduction parameters when determining an adaptive solution (resulting in a second reduction) or passed back to the user.

[2] Available from the Python Software Foundation

[3] Available from https://joblib.readthedocs.io/en/latest/

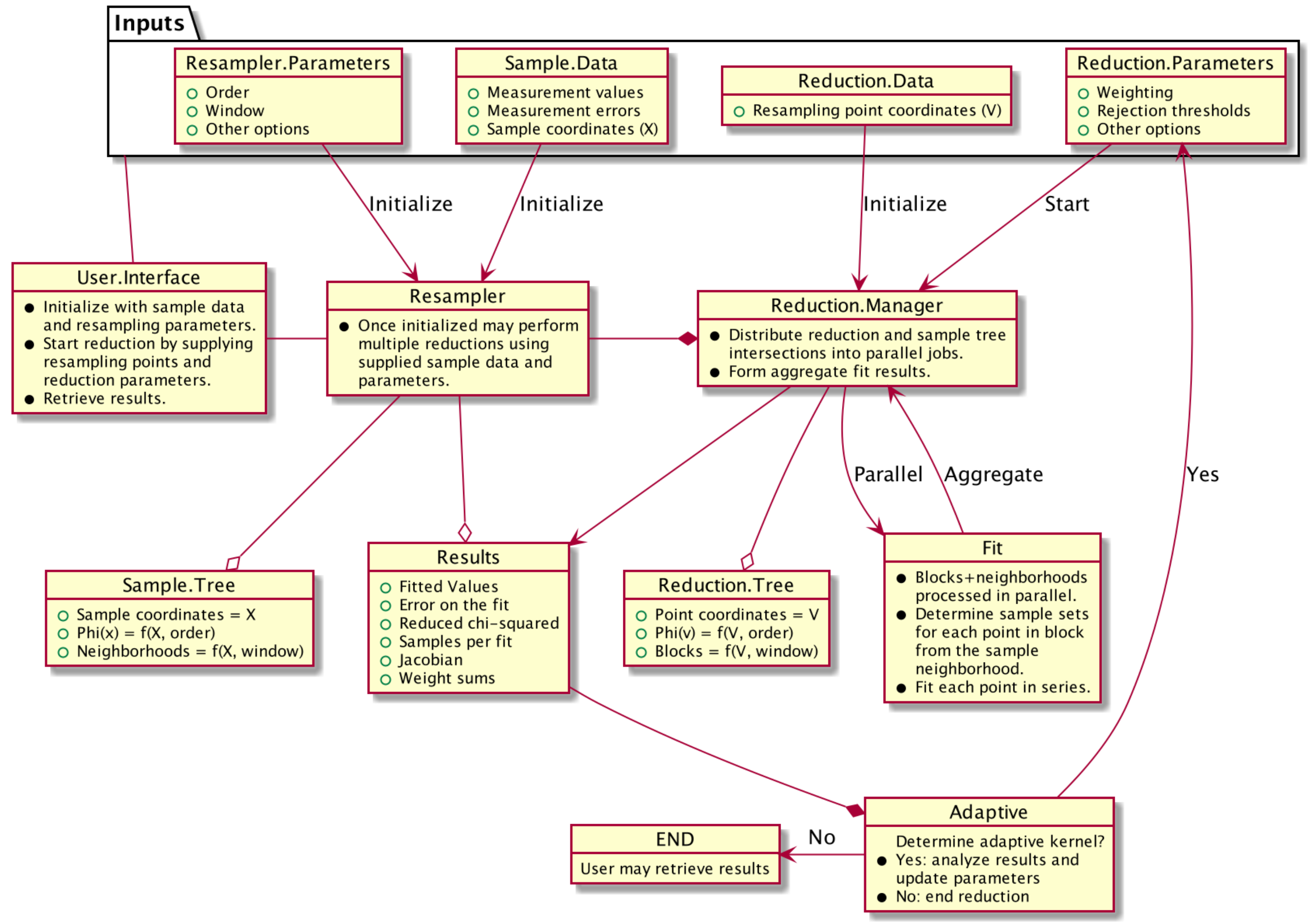

Inputs
Resampler.Parameters
Order
Window
Other options
Sample.Data
Measurement values
Measurement errors
Sample coordinates (X)
Reduction.Data
Resampling point coordinates (V)
Reduction.Parameters
Weighting
Rejection thresholds
Other options
Initialize
Initialize
Initialize
Start
User.Interface
Initialize with sample data and resampling parameters.
Start reduction by supplying resampling points and reduction parameters.
Retrieve results.
Resampler
Once initialized may perform multiple reductions using supplied sample data and parameters.
Reduction.Manager
Distribute reduction and sample tree intersections into parallel jobs.
Form aggregate fit results.
Parallel
Aggregate
Yes
Sample.Tree
Sample coordinates = X
Phi(x) = f(X, order)
Neighborhoods = f(X, window)
Results
Fitted Values
Error on the fit
Reduced chi-squared
Samples per fit
Jacobian
Weight sums
Reduction.Tree
Point coordinates = V
Phi(v) = f(V, order)
Blocks = f(V, window)
Fit
Blocks+neighborhoods processed in parallel.
Determine sample sets for each point in block from the sample neighborhood.
Fit each point in series.
Adaptive
Determine adaptive kernel?
Yes: analyze results and update parameters
No: end reduction
No
END
User may retrieve results

Figure 16. Main software objects and process flow for the full resampling algorithm. Data and variables are denoted by a green open circle. Notes and actions are marked by a filled black circle. Aggregate objects (can exist independently of the parent object) are marked with open diamond heads, while composite objects (cannot exist without parent object) are marked with a filled diamond head.

5. APPLICATIONS

5.1. *2-Dimensional Resampling of Regular Data*

The following example uses a (128 × 128 pixels) subset of the "camera" test data set from the Python scikit-image package (scikitimage (2014)), already displayed in figure 14 to show the effects of the resampling algorithm applied to regularly spaced 2-D samples onto a finer grid where $\Delta x = 0.2$ pixels. The FWHM of the observing device was estimated to be 1 pixel (by eye), and the error was estimated from the standard deviation of the blank sky region on the upper-right corner of the image ($\approx 0.7\%$ of the maximum sample value). The window radius is set to 12 pixels resulting in a median of 452 samples per 3rd order polynomial fit.

Figure 17 shows the sample image, and fits on a finer grid using standard distance weighting with $\alpha = 2\left(\mathrm{FWHM}/2\sqrt{2\ln 2}\right)^2$ and the results of the scaled and shaped adaptive distance weighting algorithm. In addition, different edge and order rejection algorithms were applied, the effects of which can be seen on the borders of each fitted image. The standard distance weighting algorithm rejected fits using the bounded order requirement, while the adaptive fits allowed for extrapolation in cases where enough unique samples are present. However, the "scaled" solution applied an edge rejection threshold of $\beta_{\text{edge}} = 1$. In this specific case, edge and order rejection options are applied for the sake of example only. Good fits are possible due to regularly spaced sampling, well-behaved data, and that we are not attempting any fits outside of the bounds of the sample region.

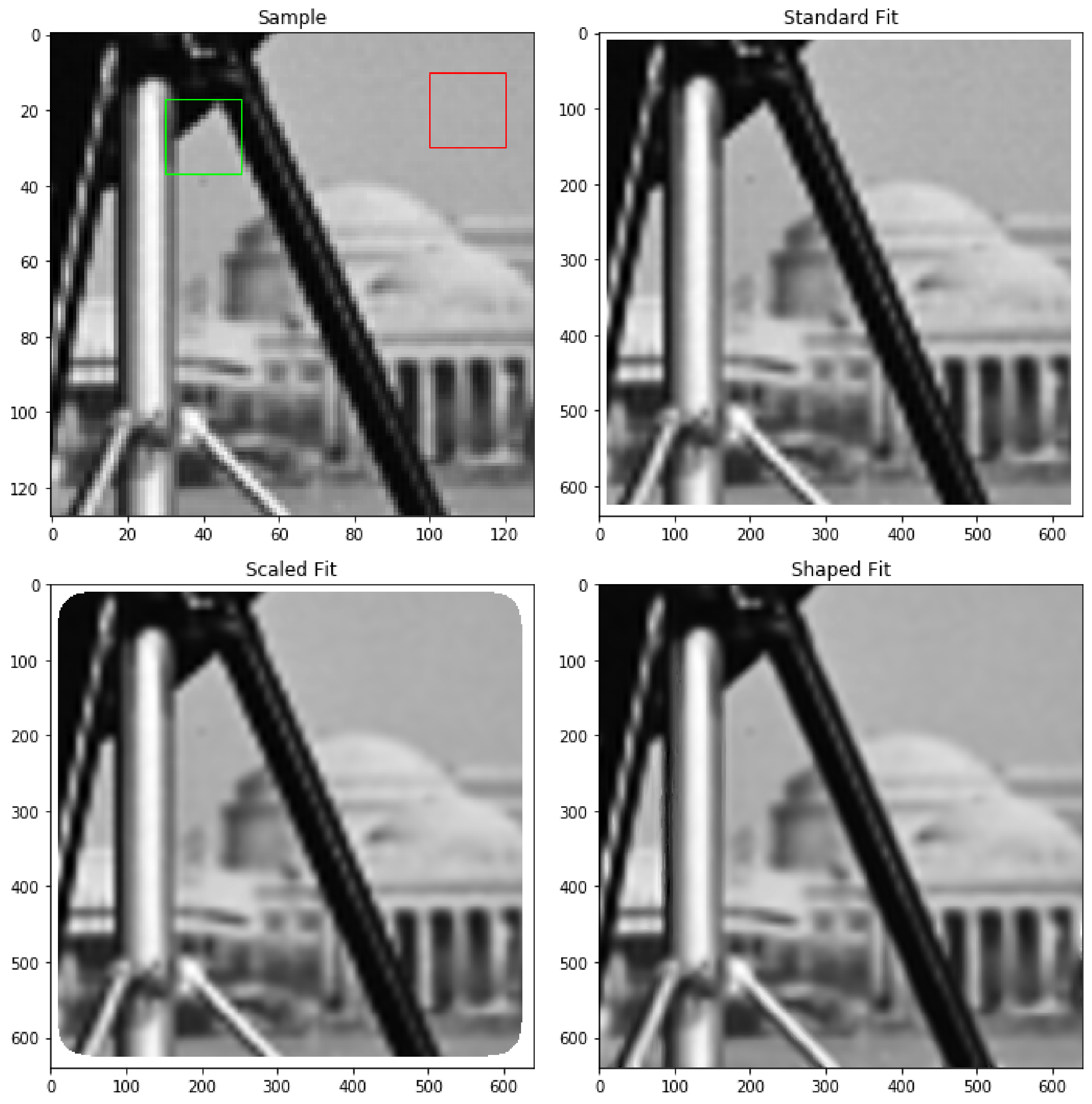


Figure 17. Resampling an image (upper-left) onto a finer grid using the standard distance weighting algorithm (upper-left), the "scaled" adaptive weighting algorithm (lower-left), and the "shaped" adaptive weighting algorithm (lower-right). The red and green squares on the upper-left image mark regions shown in figure 18.

At first glance, the standard and scaled solutions appear fairly similar to the original image, while a strong reduction in the pixelation effects can be seen in the shaped image. The similarity of the standard and scaled solutions is cosmetic only, and show notable differences when viewed on a smaller local scale. Figure 18 show enhanced sections of the fit on a smaller scale which are marked for reference on the upper left image of figure 17.

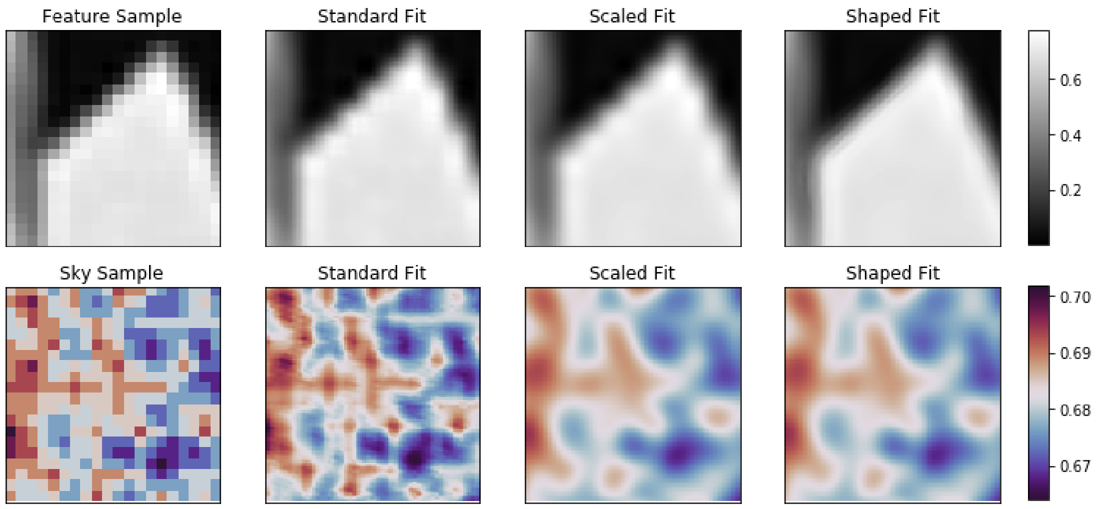


Figure 18. Subsections of the sample image and fits marked by the regions in figure 17. The upper row ("Feature"), contains an area with strong features (green box in figure 17), while the lower row ("Sky") shows a relatively smooth portion of the sample data (red box in figure 17). The columns from left to right display the sample image, the standard distance weighted fit, the scaled adaptive fit, and the shaped adaptive fit.

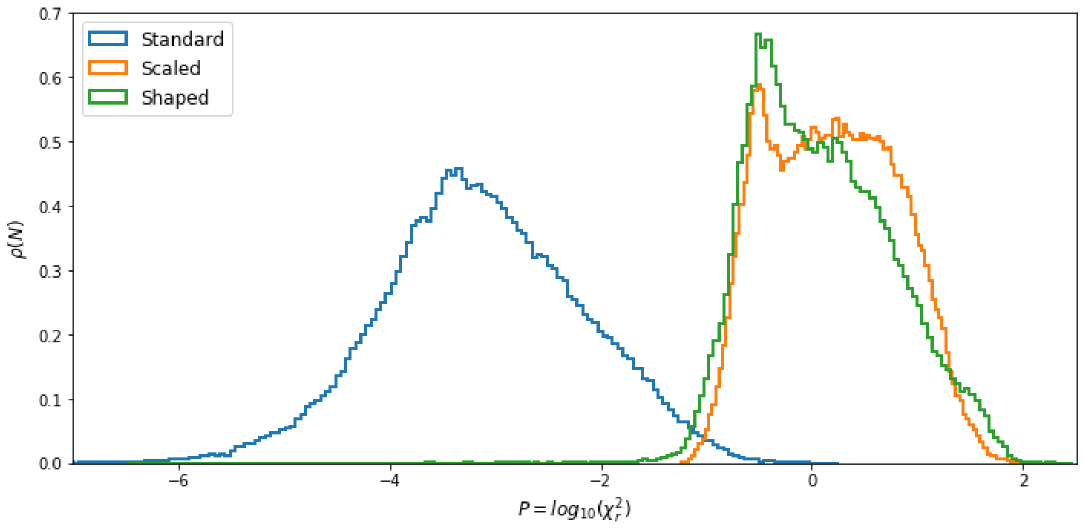


Figure 19. Histograms of $\log_{10}(\chi_r^2)$ for each of the distance weighting algorithms. The normalized count density is given as $\rho(N) = N_i / \left(\Delta P \sum_i N_i\right)$ where ΔP is the spacing between bins.

Finally, figure 19 displays histograms of the χ_r^2 distribution for all fits over the entire resampled images (409,600 pixels). On a $\log_{10}$ scale where $P = \log_{10}(\chi_r^2)$, the distribution can be approximated by a normal distribution $\mathcal{N}(\text{mean}(P), \text{variance}(P))$. For our example fits $\mathcal{N}_{\text{standard}} = (-3.126, 0.905)$, $\mathcal{N}_{\text{scaled}} = (0.198, 0.371)$, and $\mathcal{N}_{\text{shaped}} = (0.092, 0.442)$. In other words, the adaptive weighting algorithms produced fits where χ_r^2 is more closely centered around 1 with smaller variance.

5.2. *3-Dimensional Resampling of Irregular Data*

The following examples apply the new resampling algorithm to real data taken with the FIFI-LS instrument on board SOFIA (see section 1.1).

5.2.1. *M82: A comparison of weighting schemes on a high SNR source*

Observations of M82 were taken over multiple SOFIA flights using the long wavelength spectrometer. Data are irregularly spaced in cross-elevation, elevation, and wavelength (x, y, λ) centered on $\lambda = 157.694\mu m$, and J2000 coordinates (09h55m55.38s +69d40m53.4s) with x and y in units of arcseconds. The average spatial FWHM of the observations is $\text{fwhm}_x = \text{fwhm}_y = 15.8''$, and the average spectral FWHM is $\text{fwhm}_\lambda = 0.06689507\mu m$. Histograms of the sample measurement values and errors are shown in figure 20.

The sample distribution is shown in figure 21. Note that the spatial (x, y) distribution consists of a sparsely sampled region near the center bounded by two more densely sampled areas above and below, as shown on the left image. It can be seen from the right-most image that the wavelength sampling interval For a single cluster of spatial coordinates shown on the left image as a single blue dot, the median λ sampling interval is $\Delta\lambda = 0.003482\mu m \approx 19$ samples/fwhm_λ.

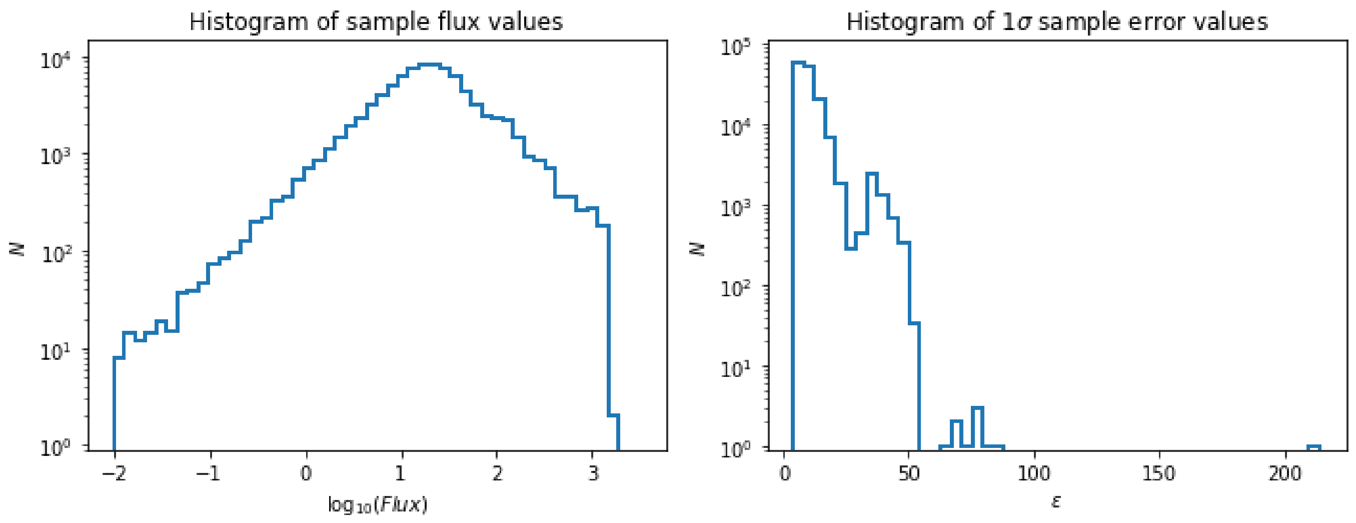


Figure 20. Histograms of the sample measurement values (left) and 1σ measurement errors (right).

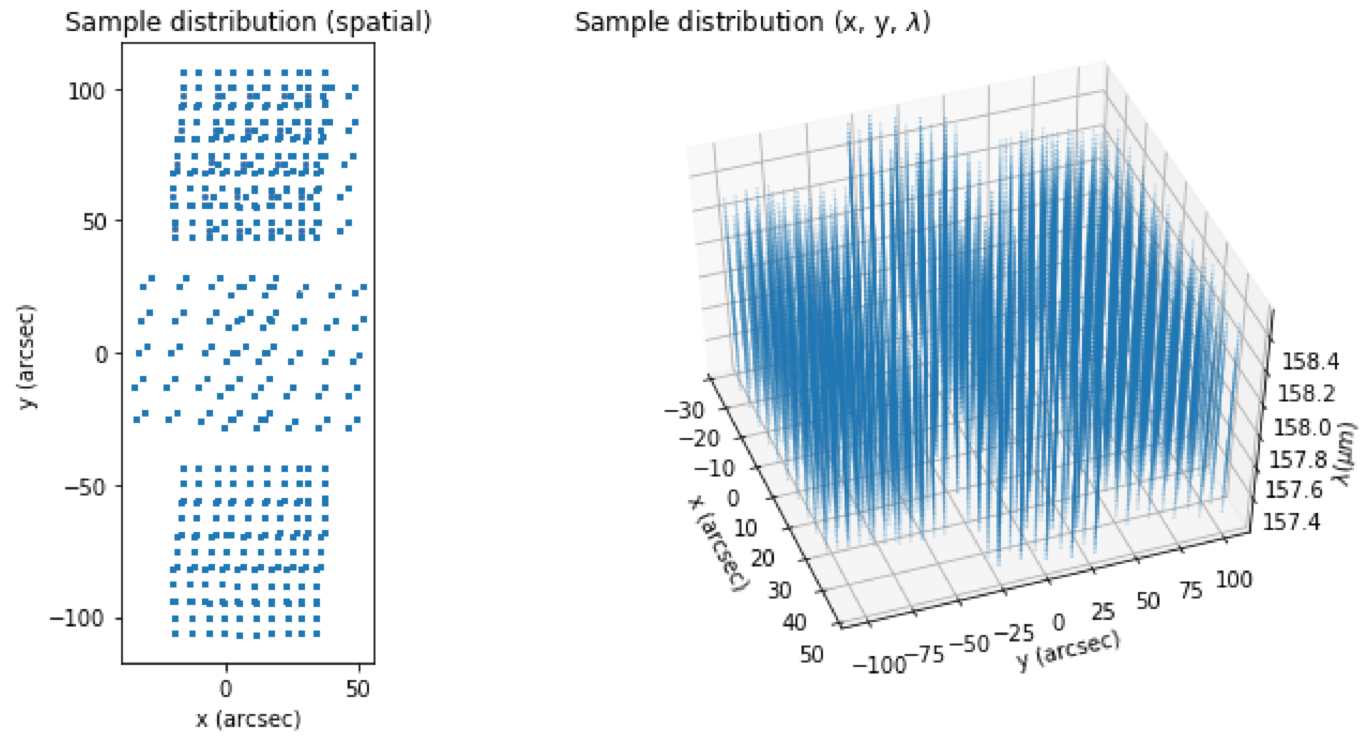


Figure 21. Left: Spatial sample distribution in x (cross-elevation) and y (elevation) coordinates. Right: Sample distribution displayed in x, y, and λ (wavelength).

Resampling was conducted to determine solutions for the standard, scaled, and shaped distance weighting algorithms on a regular spaced (x, y) (30×72) grid $(M = 2,160)$ with spacing $\Delta x = \Delta y = 3''$ at constant $\lambda = 157.83418\mu m$. The window region was set to $\omega_{x,y} = \text{fwhm}_{x,y} \times 3$, and $\omega_\lambda = \text{fwhm}_\lambda/2$ resulting in a median of 4,200 samples per second order polynomial fit in a sample space containing $N = 147,200$ samples. Fits were rejected according to the bounded order rejection algorithm, along with an edge rejection threshold of $\beta_{\text{edge}} = 0.7$ in the spatial dimensions, and 0.5 in λ. For the standard distance weighting reduction, the smoothing parameter was set to $\alpha_{x,y} = \text{fwhm}_{x,y}$, and $\alpha_\lambda = \text{fwhm}_\lambda/4$. Adaptive weighting (scaled and shaped) was only applied to the spatial dimensions, and fixed at α_λ in the wavelength dimension.

Figure 22 displays all three fits on the same scale, whereas figure 23 displays each fit normalized to within $10^{-3} \to 1$ on a log scale in order to better represent the dynamic range.

Clear discontinuities can be seen from a visual inspection of the standard distance weighting reduction in figure 22 in areas where spatial sampling density appears to transition from one value to another as shown in the left image of figure 21. However, both adaptive weighting algorithms appear smooth and continuous in these density transition regions.

Reduction times on a MacBook Pro with a 2.5 GHz Intel Core i7 processor using 16 GB of memory is 0.6 seconds using the standard weighting algorithm, and 22 and 20 seconds for the "shaped" and "scaled" weighting algorithms, respectively.

This factor in increased reduction time represents the fact that kernels are calculated for each sample (in the neighborhoods of all resampling points) rather than at each resampling point, and that $M \ll N$. Generally, when $M \approx N$, the reduction time for an adaptive weighting algorithm is roughly 2 to 3 times that of the standard weighting algorithm.

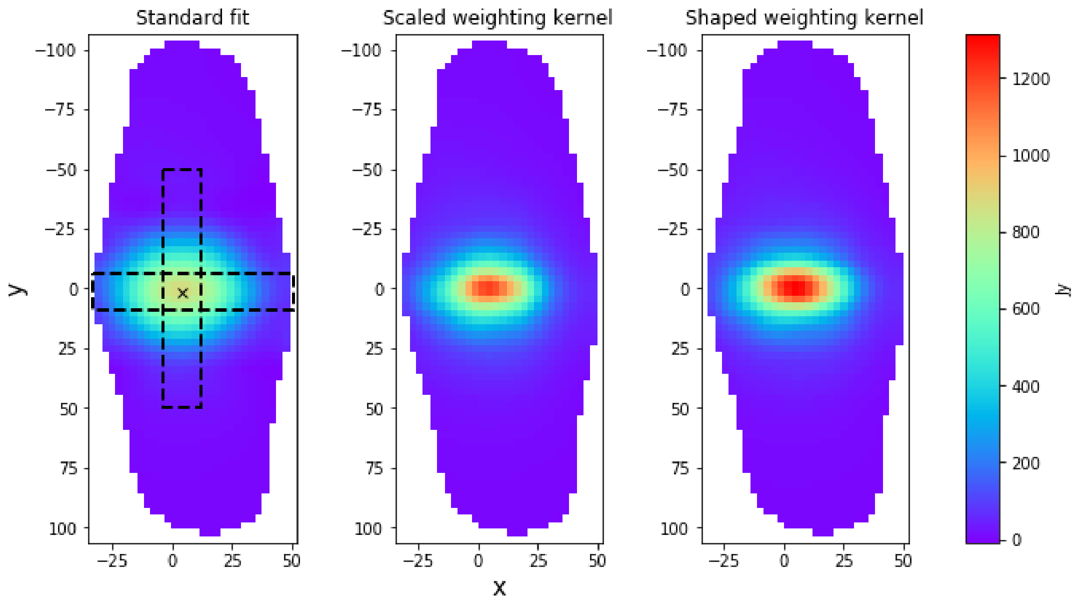


Figure 22. Images generated for $\lambda = 157.83418\mu m$ with each available weighting algorithm. Black dotted lines indicate regions centered on constant $x = 4.06''$ and $y = 1.43''$ of width fwhm$_{x,y}$. The samples plotted in figure 24 reside in these regions, with the cross marking a line of constant $\lambda = 157.83418\mu m$ through the image.

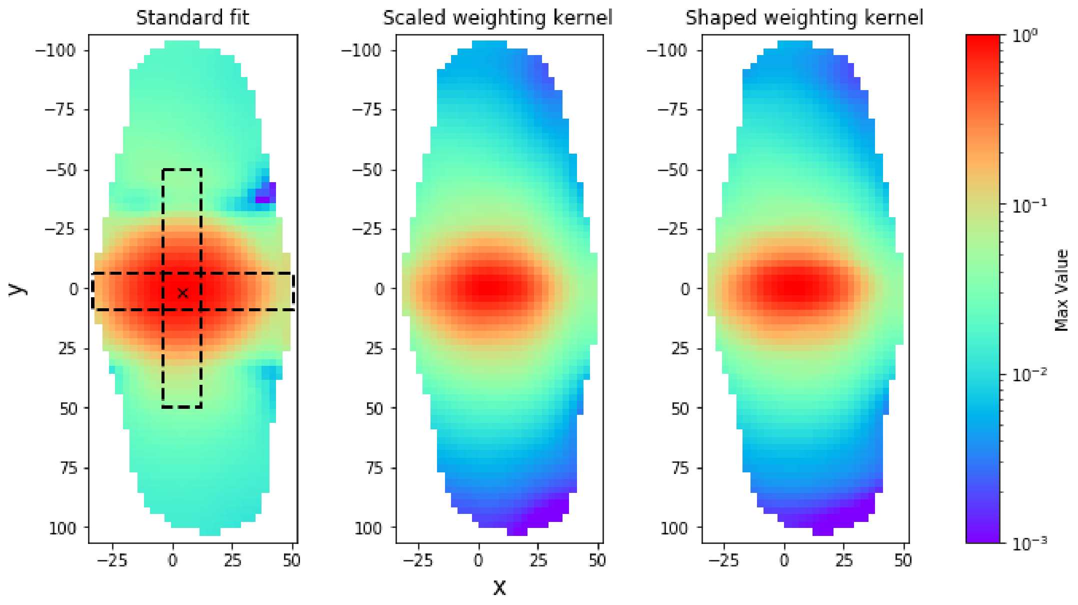


Figure 23. Images from figure 22 displayed on a log scale.

Figure 24 displays fits in which the coordinates for two dimensions are held fixed, and resampling occurs along the remaining (x, y, or λ) dimension as marked in the left image of figure 22 (over the peak). The scale of the Gaussian standard deviation of the weighting function is plotted for each dimension. In those dimensions in which adaptive weighting is enabled (x, y), the minimum and maximum standard deviations of the "scaled" weighting kernels are also drawn. For reference, all samples within ±fwhm/2 are plotted with color representing the Mahalanobis distance of a sample from the resampling point in the fixed dimensions. Color indicates the sample coordinate deviation of coordinates in the two fixed dimensions with respect to the resampling point coordinate. i.e., for the top plot in figure 24 along x, where $y = 1.43''$ and $\lambda = 157.83418\mu m$

$$\Delta\sigma(y, \lambda) = \sqrt{\left(\frac{y - 1.43}{\sigma_y}\right)^2 + \left(\frac{\lambda - 157.83418}{\sigma_\lambda}\right)^2}$$

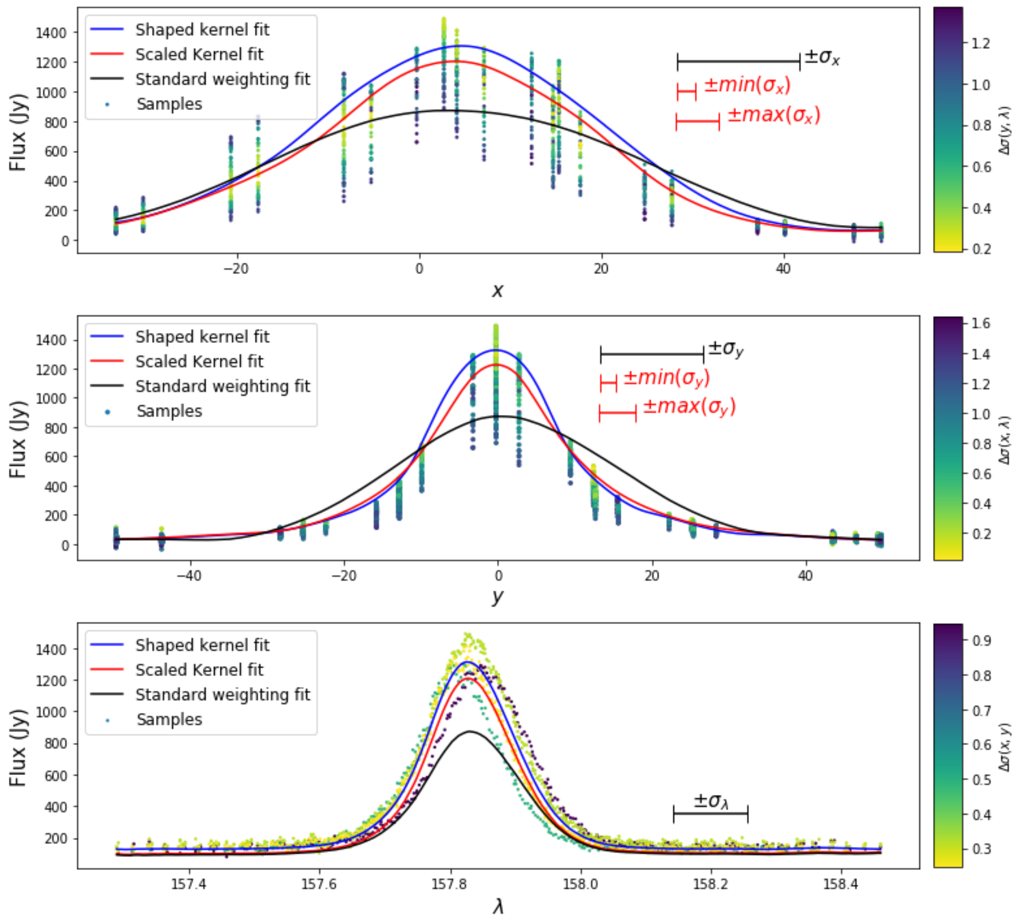


Figure 24. Cross sections of the fit along each dimension for each weighting scheme. Top: A fit along the x dimension, with y and λ held constant. Center: A fit along the y dimension, with x and λ held constant. Bottom: A fit along the λ dimension, with x and y held constant. The intersection of all fixed dimensions is marked by a cross in the left image of figure 22, and samples within the marked regions are plotted for reference. Samples are color coded according to deviation of each sample to the resampling point in dimensions that are held constant.

Figure 25 shows the resulting χ^2_r distribution for the fit along each dimension. In all cases, $\chi^2_r > 1$ indicating a poor fit. However, this is the result of attempting a fit on data samples that cannot be modelled within the expected 1σ noise limits supplied with the data. From figure 20, the typical supplied sample measurement errors are $\epsilon = 10Jy$. A quick look at figure 24 shows that the range of values at a single coordinate can vary by approximately $\pm 200Jy$ between samples that occupy close spatial/spectral coordinates.

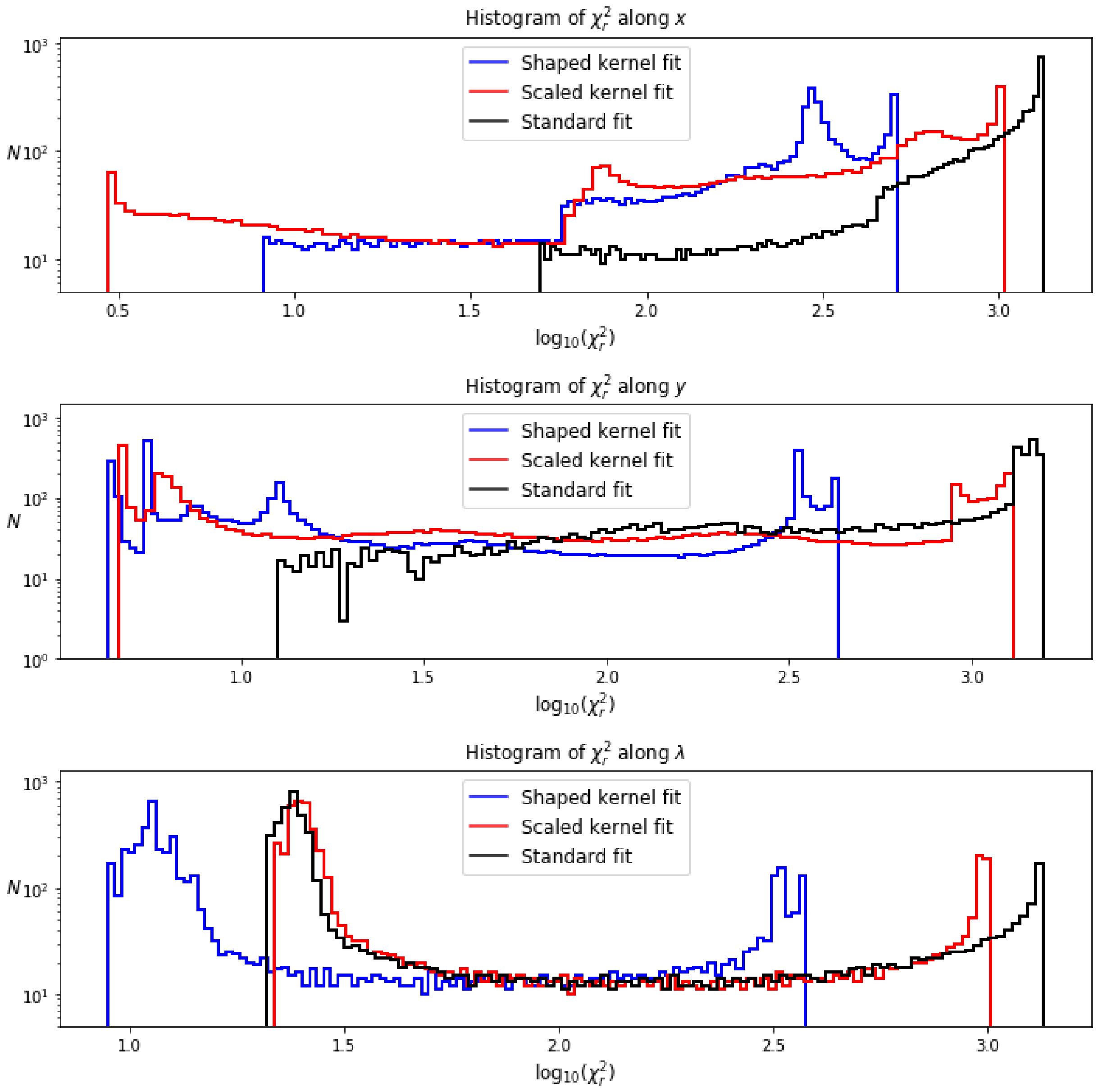


Figure 25. Histograms of the χ^2_r values for each cross-section in x, y, and λ at 5,000 regularly spaced resampling points along each dimension into 100 bins.

For reference, a fit generated using standard distance weighting with $\alpha = [\alpha_x, \alpha_y, \alpha_\lambda]/12$ is shown in figure 26 resulting in a χ_r^2 approximately centered around 1. The resulting fit is unusable with no discernible structure and must be placed on a log scale for visualization.

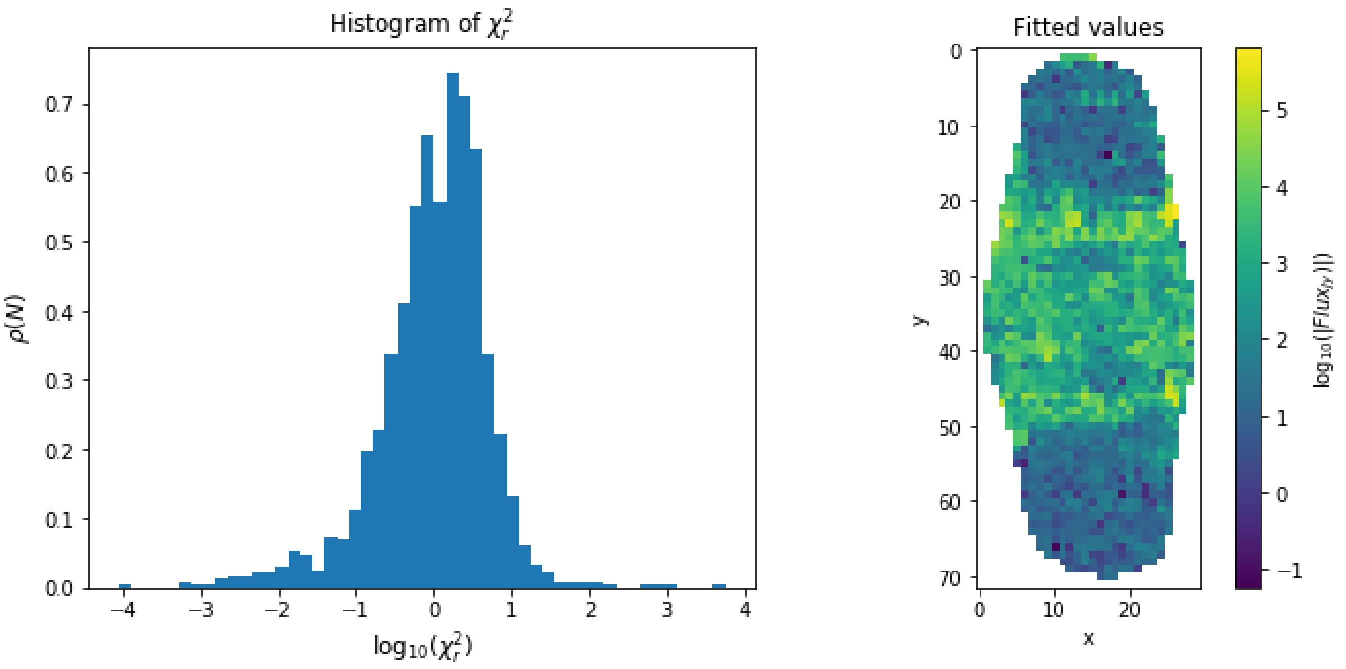


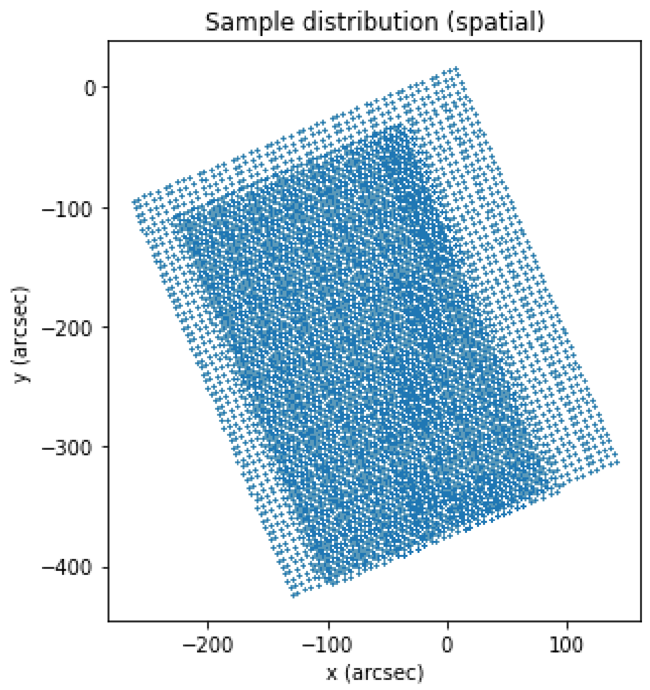
Figure 26. The result of fitting M82 sample data to $\approx \chi_r^2 = 1$. Left: Histogram of $\log_{10}(\chi_r^2)$ into 50 bins. Right: The image slice (also shown in figure 22) when fit approximating $\chi_r^2 = 1$. Colors represent flux (Jy) on a $\log_{10}$ scale of the absolute value.

5.2.2. *30 Doradus: A comparison of simultaneous vs. sequenced dimensional resampling*

The 30 Doradus data set is intended to highlight some differences between the old and new FIFI-LS data reduction pipelines, and the advantages of fitting multiple dimensions simultaneously. The initial pipeline sequentially resampled the data in wavelength (1 dimension) followed by spatial resampling in two dimensions onto a regular grid.

Our example data set contains multiple observations of 30 Doradus totalling 481,600 (x, y, λ) FIFI-LS sample measurements which are resampled onto a $(327 \times 356 \times 50)$ grid. Unlike the previous M82 example, the sampling density is relatively uniform near the area of interest, but the SNR ratio is much lower. The sample distribution is displayed in figure 27, and histograms of sample measurement values and errors are shown in figure 28.

The central wavelength of the observations was at $\lambda = 51.87619\mu m$ with $\text{FWHM}_\lambda = 0.056389\mu m$, and $\text{FWHM}_{x,y} = 6.2''$. For both resampling algorithms, the window radius in the spectral dimension is set to $\omega_\lambda = \text{FWHM}_\lambda/2$. When resampling in 3-dimensions simultaneously, the spatial window radius is set to $\omega_{x,y} = 3 \times \text{FWHM}_{x,y}$. Spectral resampling drastically reduces the number of samples available for spatial resampling. To allow for a smooth fit, the standard practice using the old pipeline was to increase the spatial window radius to $\omega_{x,y} = 6 \times \text{FWHM}_{x,y}$. In both cases, a second order polynomial fit is generated along the spectral and spatial dimensions using the standard distance weighting algorithm with a smoothing parameter of $\alpha_\lambda = \text{FWHM}_\lambda/4$ and $\alpha_{x,y} = 2 \times \text{FWHM}_{x,y}$.



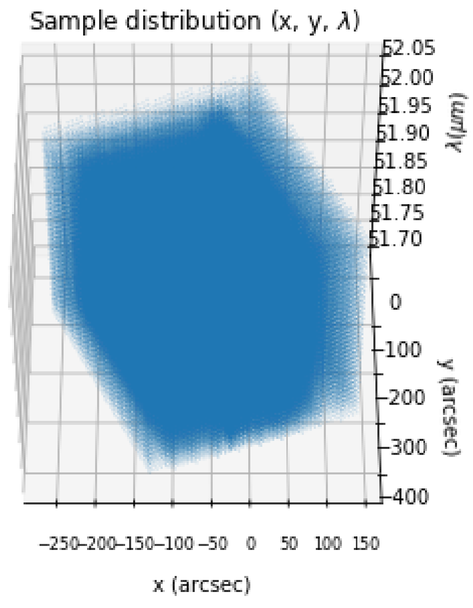


Figure 27. Left: The spatial distribution of samples in x (cross-elevation), and y (elevation). Right: The full 3D sampling distribution in x, y, and λ (wavelength).

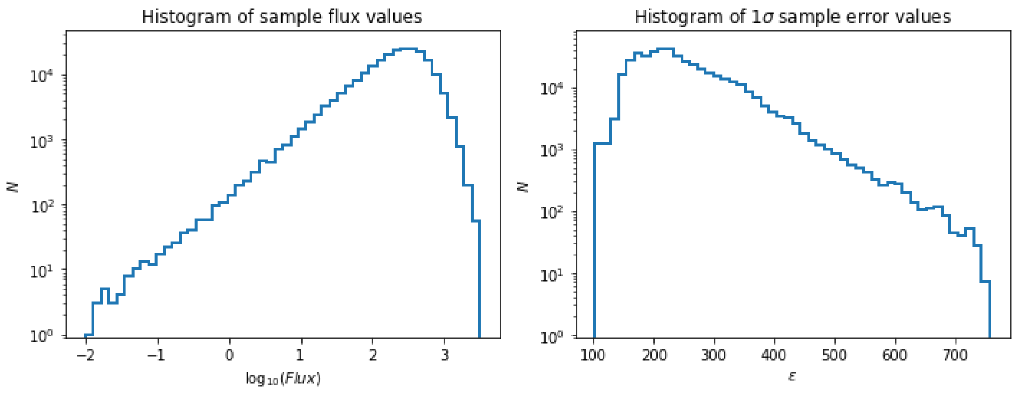


Figure 28. Left: Histogram of the sample measurement values. Right: Histogram of the associated 1σ measurement errors.

Many properties of the reduction are hard to compare due to major differences in the way data are handled (for example, number of samples per fit or χ_r^2). We do, however, present comparisons of both a spatial and spectral slice that intersect at the point of greatest emission.

Figure 29 displays a spatial slice at $\lambda = 51.94550\mu m$, and a spectral profile is shown in figure 30 at (RA, Dec) = (5h38m45.9046s, -69d05m06.2397s) which is also marked as a black cross on the left image of figure 29.

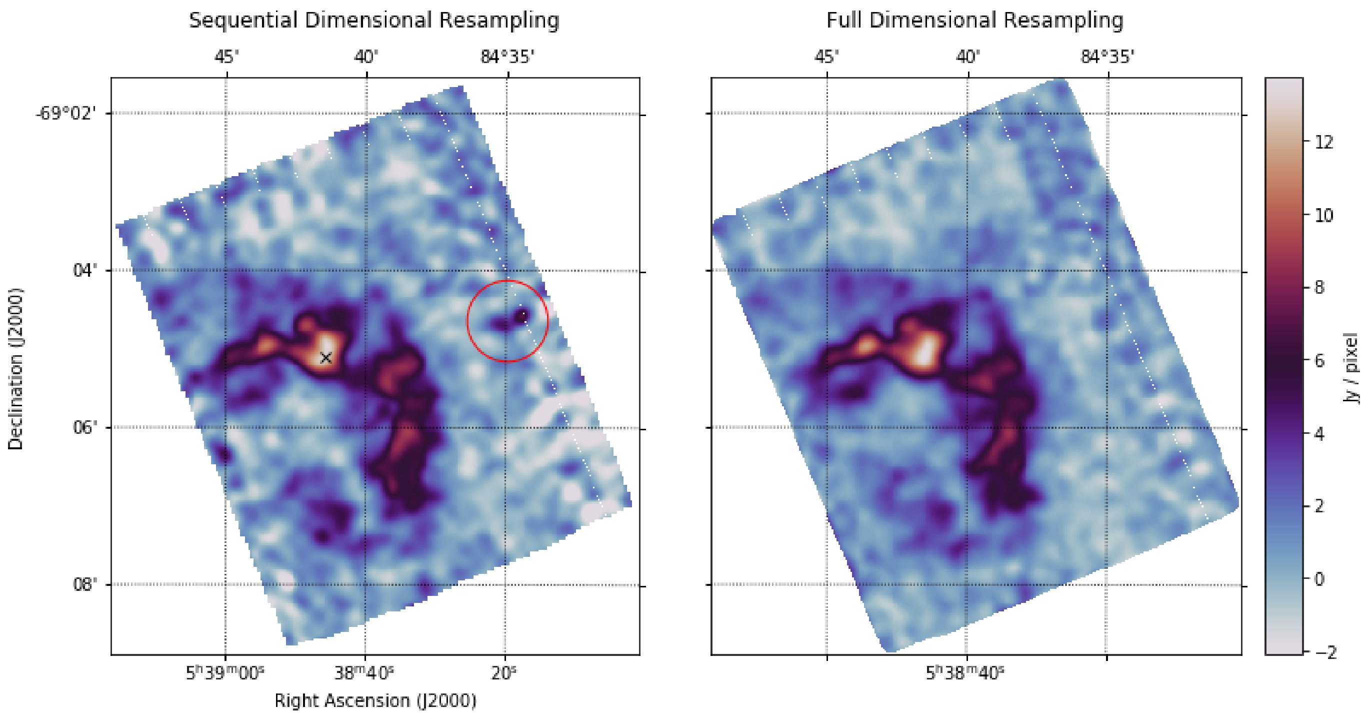

Sequential Dimensional Resampling
Full Dimensional Resampling
Declination (J2000)
Right Ascension (J2000)
Jy / pixel

Figure 29. Spatial slices at $\lambda = 51.94550\mu m$. Left: The result of first resampling in wavelength followed by the 2-dimensional spatial interpolation. Right: resampling in all dimensions simultaneously. The spatial position of the spectral profile shown in figure 30 is marked as a black cross on the left image. The red circle encloses a prominent artifact.

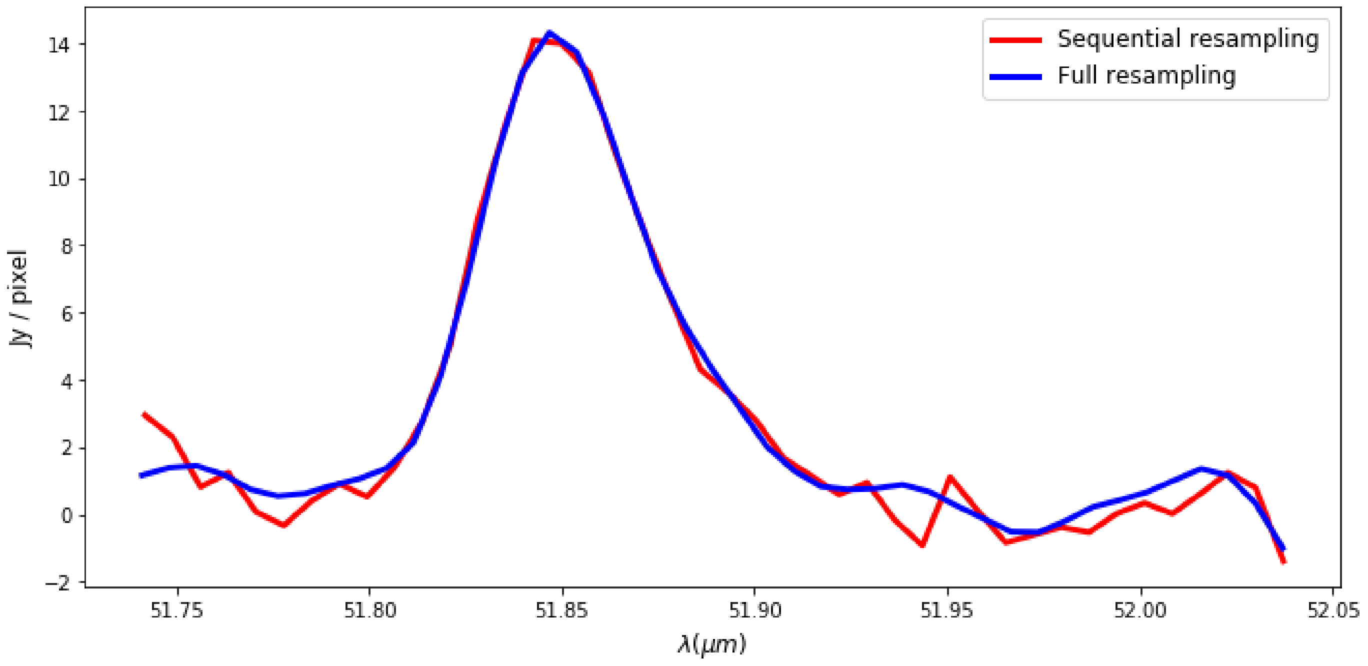


Figure 30. The spectral profile at RA = 5h38m45.9046s, Dec = -69d05m06.2397s. The red line shows the results of sequential resampling, while the blue line shows the results of directly resampling onto a 3-dimensional grid.

It can easily be seen that resampling in all dimensions simultaneously using the new FIFI-LS pipeline has produced a smoother image with fewer artifacts than the older version, although peak fluxes are relatively equal. Additionally, the spectral profile produced by the 3-dimensional fit is smooth and continuous, whereas discontinuities are clearly visible when resampling occurs sequentially over spectral and spatial dimensions. In fact, the strong artifact shown by the red circle in figure 29 is not consistent with slices at neighboring wavelengths, and is colinear with many other artifacts that are on the boundary of a change in sampling density as shown in figure 27, indicating that it is not a valid structure.

6. FUTURE WORK

This paper is primarily concerned with local polynomial regression, but could easily be extended to other families of functions or filters. For example, Gaussian, Lorentzian, Voigt, trigonometric, median etc. Modified versions of the algorithm already exist for the weighted median or mean of a local region, but do not yet implement adaptive weighting.

Another improvement that could be made would be to allow for variable window regions. Currently, the user must explicitly select a single set of window dimensions (ω) applied around each resampling point. In practice, this should be set to some minimum value to allow sufficient samples from which to derive a fit, but also be large enough to account for expansion of the adaptive weighting kernel. Since computational complexity is $\propto \mathcal{O}(|\Omega|)$, the window dimensions can have a significant effect on total reduction time. Given that selecting a small window region can invalidate certain fits, the user may be forced to choose a large window region that rectifies this situation effecting only a few points at the cost of increased processing time.

7. CONCLUSION

We have introduced a new procedure by which simultaneous multi-dimensional interpolation can be carried out. Consideration has been given to the distribution of irregularly sampled data, and with appropriate choices, spurious fits can be reduced or eliminated. An optional method is available to determine adaptive weighting factors in a single step, approximating $\chi^2_r \to 1$ on subsequent reductions. These methods have been applied to resample three-dimensional irregular data observed with the FIFI-LS instrument onto a regular data cube.

ACKNOWLEDGMENTS

This work was supported by funds granted to the Universities Space Research Association, Inc. (USRA) through the NASA/DLR Stratospheric Observatory for Infrared Astronomy (SOFIA) program. SOFIA is jointly operated by USRA under NASA contract NNA17BF53C and the Deutsches SOFIA Institut (DSI) under DLR contract 50 OK 0901 to the University of Stuttgart.

REFERENCES

Colditz S. et al., 2018, Spectral and Spatial Characterization and Calibration of FIFI-LS – The Field Imaging Spectrometer on SOFIA, JAI, Volume 7, 1840004-(12)556

Fan, J., & Gijbels, I. 1996, Local polynomial modelling and its applications, Monographs on Statistics and Applied Probability 66 (Vol. 66). CRC Press.

Fan, J., Gijbels, I., Hu, T. C., & Huang, L. S. 1996, A study of variable bandwidth selection for local polynomial regression. Statistica Sinica, 113–127.

Fischer, C. et al. 2018, FIFI-LS: The Field-Imaging Far-Infrared Line Spectrometer on SOFIA . JAI, 7, 1840003–556.

Gelman, A., & Imbens, G. 2019, "Why high-order polynomials should not be used in regression discontinuity designs," Journal of Business & Economic Statistics, 37(3), 447–456.

Gu, J., Li, Q., & Yang, J. C. 2015, "Multivariate local polynomial kernel estimators: Leading bias and asymptotic distribution," Econometric Reviews, 34(6–10), 979–1010.

Masry, E., 1996, "Multivariate local polynomial regression for time series: uniform strong consistency and rates," Journal of Time Series Analysis, 17(6), 571–599.

Mood, A., Graybill, F., & Boes, D., 1974 Introduction to the Theory of Statistics (3rd ed.). McGraw-Hill. p. 229.

Noferini, V., & Townsend, A. 2016, "Numerical instability of resultant methods for multidimensional rootfinding," SIAM Journal on Numerical Analysis, 54(2), 719–743.

Lam, S. K., Pitrou, A., & Seibert, S. (2015). "Numba: A llvm-based python jit compiler," In Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC (pp. 1–6).

Oliphant, T. E. (2006). A guide to NumPy (Vol. 1). Trelgol Publishing USA .

Omohundro, S. M., (1989), "Five balltree construction algorithms," International Computer Science Institute Technical Report

Pedregosa, F. et al., 2011. "Scikit-learn: Machine learning in Python," Journal of machine learning research, 12(Oct), pp.2825–2830.

G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

Stone, C.J. 1977, "Consistent Nonparametric Regression," Ann. Statist., vol. 5 (1977), 595–645

Vacca, W. D. et al. 2020, "The Data Reduction Pipeline for FIFI-LS, the MIR Integral Field Spectrograph for SOFIA," ADASS 2019

Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. 2014, ”scikit-image: Image processing in Python,” PeerJ 2:e453