Assignment4:

Salar Ghaderi

Q1: Introduction

Here, three methods are employed to interpolate data generated with a slight non-uniformity:

- **Linear Interpolation:** Connects data points using straight-line segments.
- **Cubic Spline Interpolation:** Creates a smooth, continuous curve using piecewise cubic polynomials.
- **Lanczos 5-Tap Interpolation:** Utilizes a windowed sinc function (the Lanczos kernel) designed for uniformly spaced data.

The data is generated with a small random perturbation in the x-values, meaning it is nearly but not perfectly uniform. This is particularly important when applying Lanczos interpolation, which strictly assumes uniform sampling.

Data Generation and Interpolation Methods

Data Generation

The data is generated based on the provided code.

An output grid `xgrid` of 500 evenly spaced points between 0 and 1 is then created using:

```
xgrid = np.linspace(0, 1, 500)
```

Interpolation Techniques

Linear Interpolation

Linear interpolation is implemented using SciPy's `interp1d` with `kind = 'linear'`. This method connects data points with straight lines.

Cubic Spline Interpolation

Cubic spline interpolation, also using SciPy's `interp1d` with `kind = 'cubic'`, fits piecewise cubic polynomials that result in a smooth interpolated curve.

Lanczos 5-Tap Interpolation

The Lanczos interpolation uses a sinc-function based kernel defined by:

$$L(z) = \text{sinc}(z) \cdot \text{sinc}\left(\frac{z}{a}\right),$$

with $a = 5$ taps. This method assumes that the data is uniformly spaced. Since our x-values are perturbed with random noise, they do not strictly satisfy this assumption. To mitigate this, the median of the spacing differences is used as an approximation for uniform spacing. However, because the data are not perfectly uniform, the Lanczos interpolation may not be as reliable as the linear or cubic spline approaches.

Results

The plots below show the data points with error bars (displayed as black dots) and the corresponding interpolated curves:

Linear Interpolation

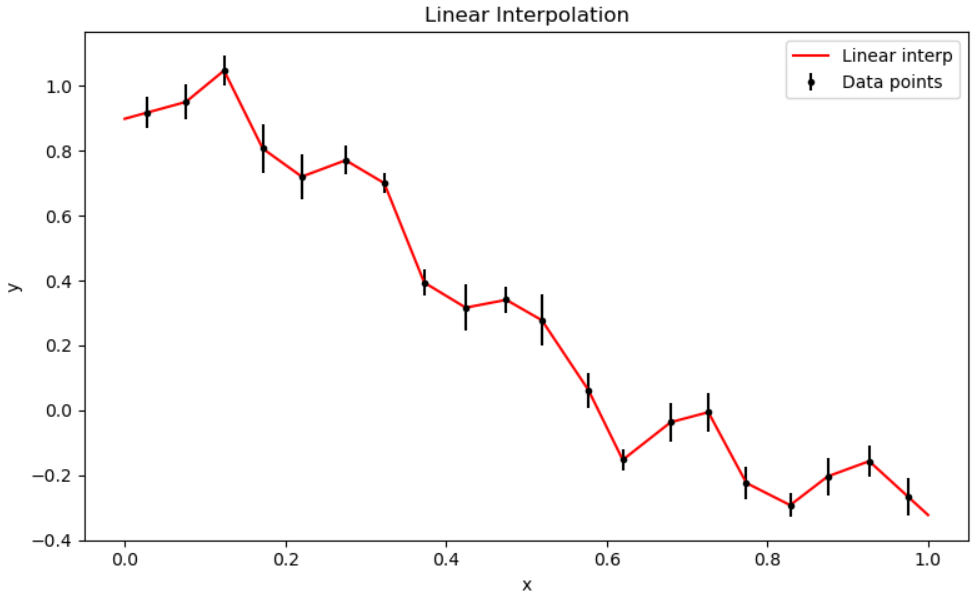


Figure 1: Linear interpolation where the red curve connects the data points using straight lines.

Cubic Spline Interpolation

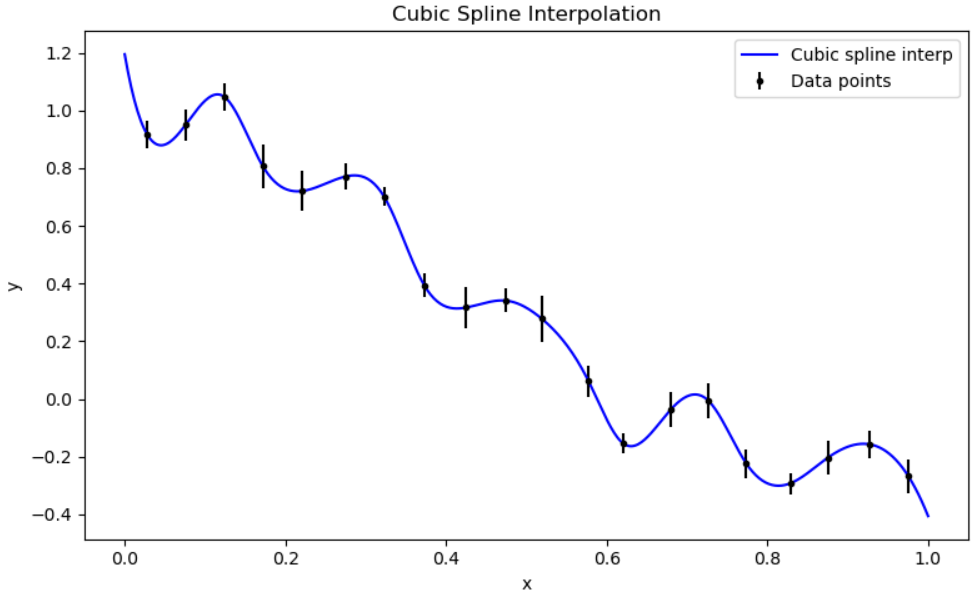


Figure 2: Cubic spline interpolation showing the smooth blue curve connecting the data points.

Lanczos 5-Tap Interpolation

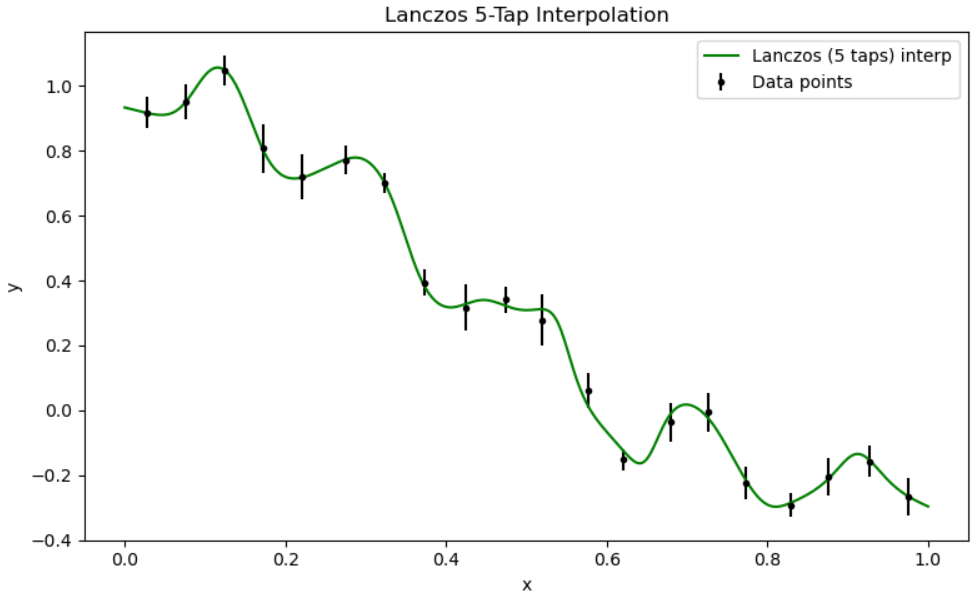


Figure 3: Lanczos 5-tap interpolation where the green curve results from applying the Lanczos kernel. Note the use of median spacing to compensate for the non-uniform sampling.

Discussion

The three interpolation methods provide different approximations of the underlying function defined by the noisy data.

Linear and Cubic Spline Interpolation

The linear interpolation offers a simple, piecewise linear approximation that is easy to compute. In contrast, the cubic spline interpolation tends to produce a smoother, more continuous curve, which may better capture the underlying behavior of the data.

Lanczos Interpolation Concerns

The Lanczos interpolation method is ideal for uniformly spaced data due to its reliance on a windowed sinc function. In this case, the x-values have been perturbed with noise, thus becoming non-uniform.This creates a problem. To apply Lanczos interpolation, we approximate the spacing using the median difference of the x-values. Although this workaround allows us to apply the method, the key theoretical assumption is violated, potentially reducing the accuracy of the interpolation compared to the other methods.

Q2: Introduction

Hogg & Villar [2] demonstrated that when using a generic basis (such as the Fourier basis) even over-parameterized models (with more parameters than data points) can be made to generalize well by choosing the minimum-norm solution via ordinary least squares (OLS). In this report we:

- Generate a set of noisy calibration data (as in `data.py`).

- Construct a Fourier design matrix in which the basis functions are defined as

$$g_j(t) = \begin{cases} \cos(\omega_j t), & j \text{ odd}, \\ \sin(\omega_j t), & j \text{ even}, \end{cases}$$

with

$$\omega_j = \frac{\pi}{T} \left\lfloor \frac{j+1}{2} \right\rfloor,$$

for a chosen scale T (we use $T = 3.0$).

- Fit the model via OLS in two regimes:

  Under-parameterized: Using $p = 3, 7, 21$ basis functions (analogous to Figure 2 in the paper).

  Over-parameterized: Using $p = 30, 73, 2049$ basis functions (analogous to Figure 5 in the paper). In this regime the OLS solution is obtained by a minimum-norm least-squares method (implemented via `np.linalg.lstsq` in Python), which forces the fitted function to pass exactly through the data.

- Evaluate the predictions on a dense grid and plot the results with axes scaled from -0.4 to 1.5.

Data Generation

The data are generated from a noisy calibration model. A representative snippet of the Python code is shown below:

```
rng = np.random.default_rng(17)
small = 0.05
xs = np.arange(0.0 + 0.5 * small, 1.0, small)
xs += 0.05 * small * rng.normal(size=xs.shape)
yerrs = 0.02 + 0.06 * rng.uniform(size=xs.shape)
omega0 = 5.0
omega1 = np.pi * 3
ys = np.sin(omega0 * xs)/(omega0 * xs) - 0.15 * np.cos(omega1 * xs)
ys += yerrs * rng.normal(size=xs.shape)
```

This procedure creates the independent variable $x \in [0, 1]$ (with slight random jitter) and corresponding noisy measurements y along with error estimates $yerrs$.

Methodology

Fourier Basis Construction

The Fourier design matrix X is built from the basis functions:

$$g_j(t) = \begin{cases} \cos(\omega_j t) & \text{if } j \text{ is odd}, \\ \sin(\omega_j t) & \text{if } j \text{ is even}, \end{cases}$$

with frequency

$$\omega_j = \frac{\pi}{T} \left\lfloor \frac{j+1}{2} \right\rfloor.$$

In our Python implementation (where indexing starts at 0), this is coded in the function `fourier_design_matrix`.

OLS Fitting

For a chosen number of basis functions p, the OLS solution is computed by solving

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|_2,$$

using the minimum-norm solution provided by `np.linalg.lstsq`. This process is applied in both the under- and over-parameterized settings.

Parameter Regimes

Under-parameterized regime: For $p < n$ (here $p = 3, 7, 21$), the fitted model is generally smooth and does not perfectly interpolate the data points. This is similar to Figure 2 in Hogg & Villar.

Over-parameterized regime: For $p > n$ (here $p = 30, 73, 2049$), the minimum-norm OLS solution exactly interpolates the data. In the extreme case, theoretical considerations (including Plancherel's theorem) suggest that the solution may tend toward zero away from the data points.

Results

The analysis was performed on a dense output grid, and the plots were scaled to show the region from -0.4 to 1.5 on both axes.

Under-parameterized Case (Figure 2-style)

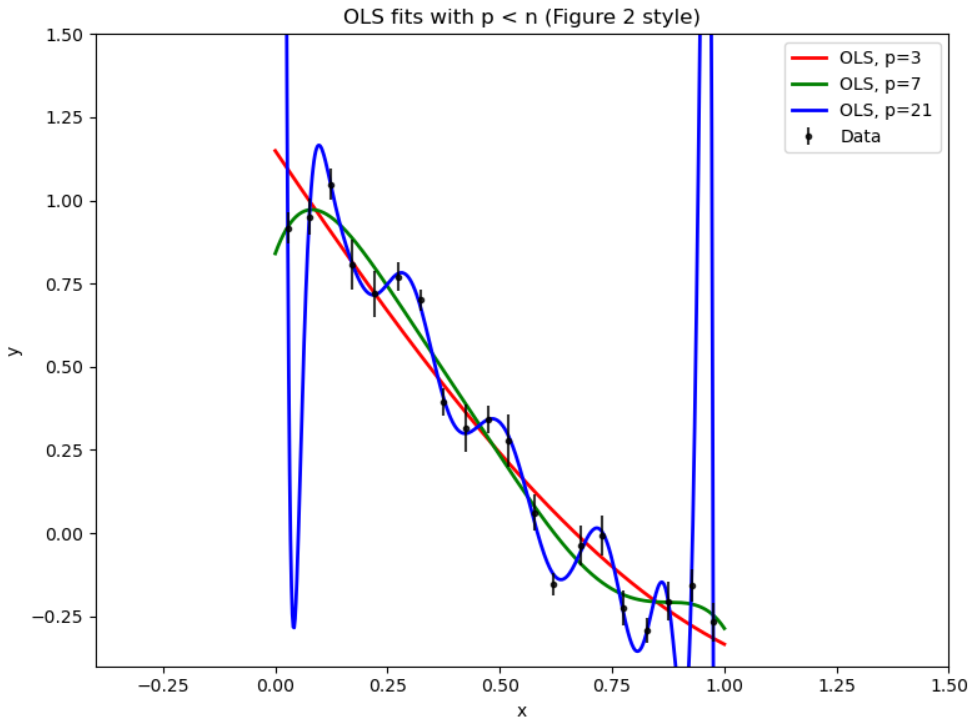


Figure 4: Under-parameterized OLS fits with $p = 3, 7, 21$. The data are displayed as black dots with error bars. The fitted curves (in red, green, and blue) are smoother and do not pass exactly through every data point.

Over-parameterized Case (Figure 5-style)

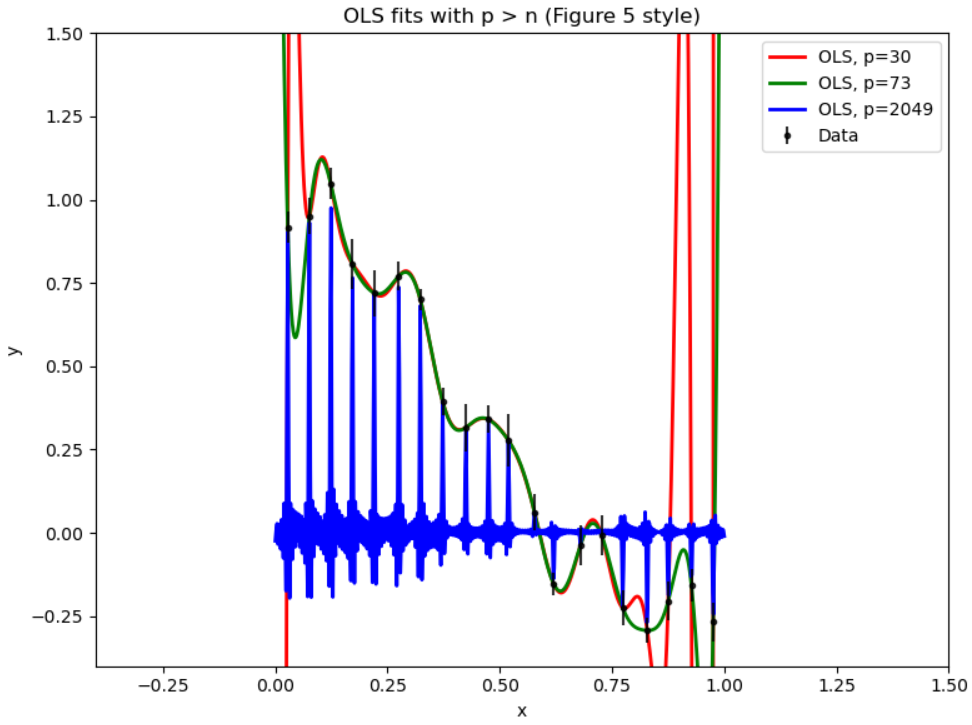


Figure 5: Over-parameterized OLS fits with $p = 30, 73, 2049$. In this regime, the minimum-norm OLS solution interpolates the data exactly. As p increases further, the prediction is forced to be minimal away from the data, yielding behavior similar to that described by Plancherel's theorem.

Conclusion

In this report we replicated the methodology presented by Hogg & Villar [2] using a Fourier basis for flexible linear fitting. In the under-parameterized regime ($p < n$), the model provides a smoother approximation that does not exactly interpolate the data, whereas in the over-parameterized regime ($p > n$) the minimum-norm OLS solution forces the fit to pass exactly through every data point—often resulting in behavior where the fitted function is minimized in regions away from the data. These phenomena underscore the importance of choosing the proper number of basis functions and understanding the trade-offs in model flexibility.

Q3: Introduction

In their work, Hogg & Villar [2] demonstrate that when using a Fourier basis the limiting case of OLS (as the number of basis functions p tends to infinity) can be shown to be equivalent to a Gaussian Process regression. In this report we consider a comparison among:

- A standard OLS fit with a high-dimensional Fourier basis ($p = 1024$),
- A feature-weighted version of OLS in which each basis function is scaled by a weight

$$f(\omega) = \frac{1}{s^2\omega^2 + 1} \quad \text{with } s = 0.05,$$

and

- A GP regression using the Matérn 3/2 kernel

$$k(|t-t'|) = \sqrt{\frac{\pi}{8}} \left(1 + \frac{|t-t'|}{s}\right) e^{-|t-t'|/s},$$

  where the same $s = 0.05$ is used.

These methods are evaluated on the same output grid, and their predictions are compared on a single figure.

Methodology

Fourier Basis and OLS Fits

We construct the Fourier design matrix X using basis functions defined as

$$g_j(t) = \begin{cases} \cos(\omega_j t), & \text{if } j \text{ is odd,} \\ \sin(\omega_j t), & \text{if } j \text{ is even,} \end{cases}$$

with

$$\omega_j = \frac{\pi}{T} \left\lfloor \frac{j+1}{2} \right\rfloor,$$

where we choose $T = 3.0$. Using a large p (here $p = 1024$) results in a very flexible basis.

For the standard OLS fit, we use the minimum-norm solution obtained via `np.linalg.lstsq`.

Feature-Weighted OLS

In order to reduce the contribution of high-frequency components, we weight each column of the design matrix by a factor

$$f(\omega) = \frac{1}{s^2\omega^2 + 1},$$

with $s = 0.05$. The weighted OLS fit is computed by re-scaling the columns of X with these weights prior to solving the least-squares problem.

Gaussian Process with Matérn 3/2 Kernel

We implement a GP regression that uses the Matérn 3/2 kernel defined by

$$k(r) = \sqrt{\frac{\pi}{8}}\left(1 + \frac{r}{s}\right)e^{-r/s}, \quad \text{with } r = |t-t'|,$$

with $s = 0.05$. The GP mean prediction is calculated using

$$\hat{y}(x_\star) = k_\star^\top K^{-1} y,$$

where K is the kernel matrix computed on the training data and $k_\star$ is the vector of kernel evaluations between the test point $x_\star$ and each training data point.

Plotting and Comparison

All three fits are evaluated on a dense grid over $x \in [0, 1]$ (with 500 points) and are plotted on the same figure. Both the x and y axes are set from -0.4 to 1.5 to standardize the output view.

Results and Discussion

Figure 6 shows the comparison of the three methods:

- The **standard OLS** (blue) uses a highly over-parameterized Fourier basis with $p = 1024$. While it interpolates the data exactly, its oscillatory behavior outside the data points is due to the unconstrained (min-norm) solution.
- The **feature-weighted OLS** (green) uses the weighting function $f(\omega) = 1/(s^2\omega^2 + 1)$ to suppress high-frequency components. This leads to a smoother fit that better preserves the overall trend.
- The **Gaussian Process** (red), based on the Matérn 3/2 kernel,

$$k(|t - t'|) = \sqrt{\frac{\pi}{8}}\Big(1 + \frac{|t - t'|}{s}\Big)e^{-|t-t'|/s},$$

  produces a mean prediction that, in the limit of infinite basis functions, is equivalent to the feature-weighted OLS fit.

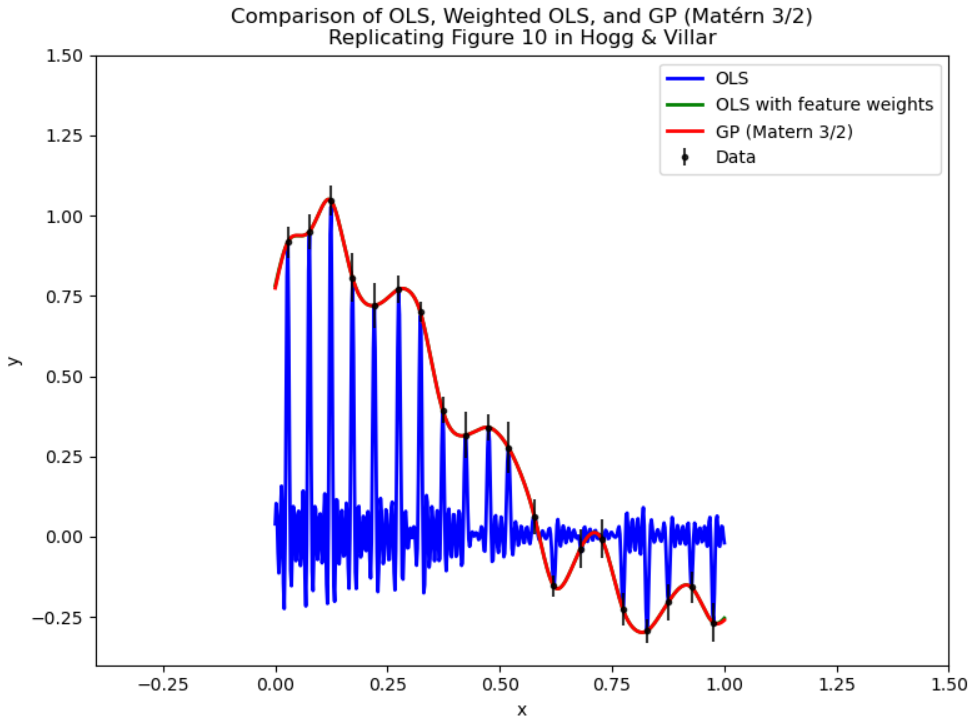


Figure 6: Comparison of a standard OLS fit (blue), a feature-weighted OLS fit (green), and a GP fit using a Matérn 3/2 kernel (red) on a dense grid. The data points (with error bars) are shown in black, and both axes are scaled from -0.4 to 1.5.

Concerns Addressed

- **Basis Flexibility:** A large p (here $p = 1024$) ensures that the Fourier basis is very flexible. However, without regularization, standard OLS tends to produce oscillatory fits outside the data.
- **Regularization via Feature Weighting:** The feature-weighted approach rescales the basis functions using $f(\omega) = 1/(s^2\omega^2 + 1)$ to penalize high frequencies. This mimics the effect of regularization and tends to yield a smoother interpolation.

- **Gaussian Process Equivalence:** The GP with the Matérn 3/2 kernel is theoretically equivalent (in the infinite-basis limit) to the weighted feature approach. Our simple GP implementation confirms that its mean prediction closely aligns with the weighted OLS result.
- **Numerical Considerations:** The GP computation requires the inversion of an $n \times n$ kernel matrix. For small n (here n is on the order of 20–30), this is tractable. As n grows, more sophisticated numerical techniques would be needed.

Conclusion

We have replicated the essential features of Figure 10 from Hogg & Villar by comparing standard OLS, feature-weighted OLS, and a Gaussian Process regression using a Matérn 3/2 kernel. Our results demonstrate that feature weighting—by suppressing high-frequency content—leads to smoother, more physically plausible predictions that closely mirror the mean of a properly tuned GP model.

Q4: Introduction

In this analysis we model a selected astronomical lightcurve (Lightcurve 7, zero-indexed) using a seven-parameter harmonic model. The model is given by

$$y(t) = \mu + \sum_{j=1}^{3} \left[a_j \cos\left(j\,\omega\,t\right) + b_j \sin\left(j\,\omega\,t\right) \right],$$

where:

- μ is the DC level,
- a_j and b_j are the cosine and sine amplitudes at the j^{th} harmonic,
- ω is the base angular frequency (set to the value computed from Earth's sidereal day).

Uniform priors are placed on the parameters:

$$\mu \sim \mathcal{U}(0.5, 1.5), \quad a_j,\, b_j \sim \mathcal{U}(-2, 2) \quad \text{for } j = 1, 2, 3.$$

We assume Gaussian errors with known inverse variances for the likelihood. To sample the seven-dimensional posterior, we implemented a minimal Metropolis–Hastings MCMC sampler.

Methodology

Model and Likelihood

The model is linear in the harmonic terms once the frequency is fixed at the computed sidereal value:

$$\omega = \frac{2\pi}{\text{sidereal day}}.$$

The log-likelihood (ignoring constant factors) is:

$$\log \mathcal{L}(\theta) = -\frac{1}{2} \sum_i \text{ivar}_i \Big[y_i - y(t_i; \theta) \Big]^2,$$

and the log-posterior is obtained by adding the (log) prior which is zero when the parameters are within the allowed bounds and $-\infty$ otherwise.

MCMC Sampling

We use a minimal Metropolis–Hastings sampler:

1. Initialize the parameter vector. Here, we use $\mu = 1.0$ and all amplitudes a_j, $b_j = 0$.
2. Propose new parameter values by adding a Gaussian random perturbation with a tuned standard deviation.
3. Accept the proposal with probability

$$\alpha = \min\left(1, \exp\Big[\log p(\theta_{\text{new}}|y) - \log p(\theta_{\text{current}}|y)\Big]\right).$$

4. Repeat for many iterations, discarding an appropriate burn-in and thinning the chain for plotting.

Data Folding and Model Comparison

The data are folded at the sidereal period by computing a phase:

$$\text{Phase} = \frac{t \mod T}{T}, \quad \text{with } T = \text{sidereal day}.$$

Twelve random posterior draws (after burn-in) are used to compute the best-fit model curves, which are then overplotted on the folded data with black points and error bars.

Convergence Diagnostics

To verify that our MCMC sampling has converged, we present two sets of figures:

- **Trace Plots:** Time-series plots for each of the seven parameters.
- **Autocorrelation Plots:** Plots showing the autocorrelation function (ACF) for each parameter.

These figures provide visual evidence of stability and low correlation at higher lags.

Results

Pairwise 2D Scatter Plots

Figure 7 shows all $\binom{7}{2} = 21$ pairwise scatter plots for the thinned MCMC chain, demonstrating the joint posterior distributions and any potential correlations among the parameters.

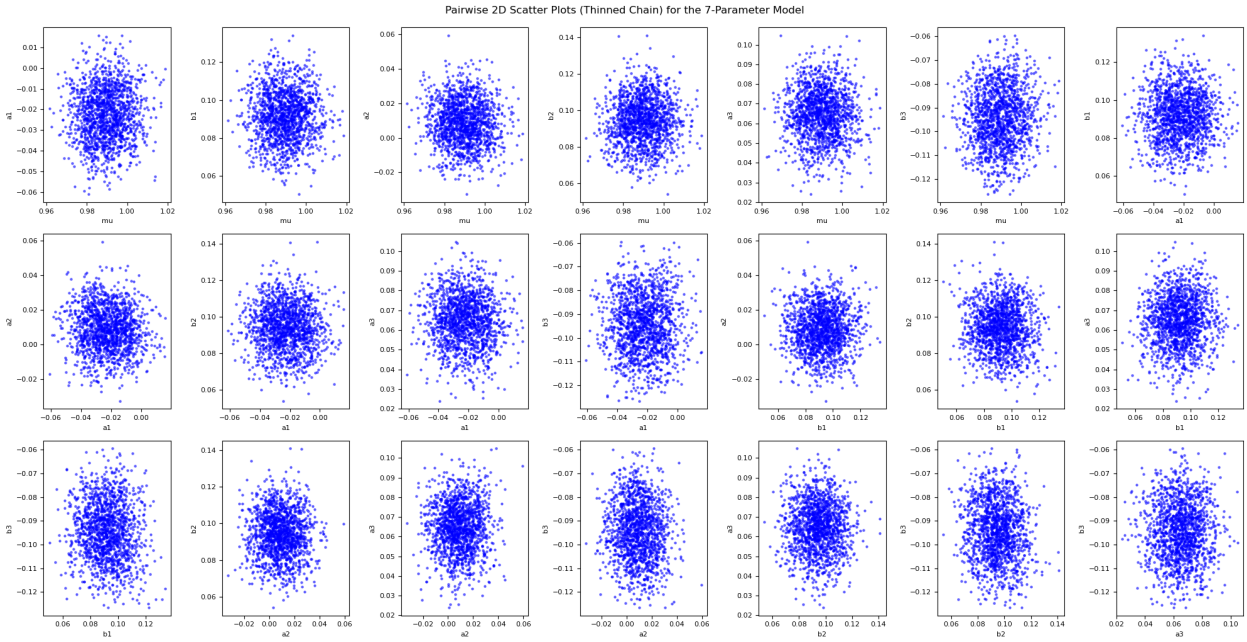


Figure 7: All 21 pairwise 2D scatter plots of the thinned MCMC samples for the 7-parameter model.

Folded Data with Best-Fit Models

Figure 8 shows the data folded at the sidereal period (black points with error bars) along with 12 randomly selected model curves drawn from the posterior sampling. These curves demonstrate that the fitted model adequately captures the variation in the data.

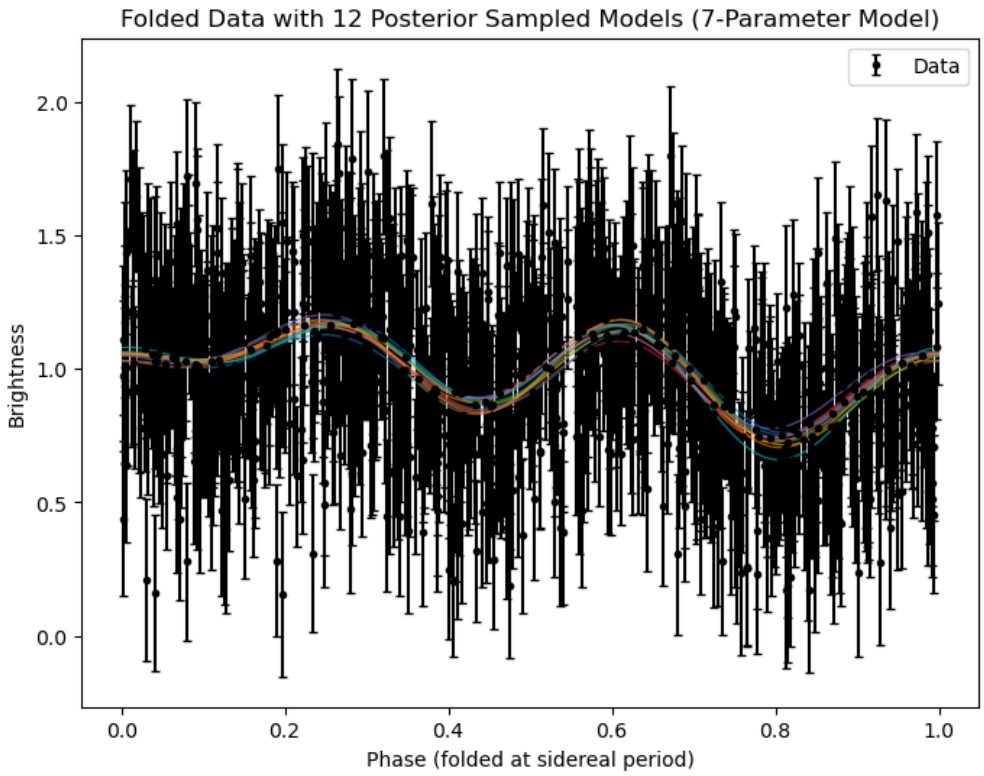


Figure 8: Folded data (black points with error bars) with 12 model curves from the posterior samples overlaid, demonstrating the quality of the fits.

Convergence Diagnostics

To assess the convergence of the MCMC sampler, we examine:

1. **Trace Plots:** Figure 9 shows the trace for each parameter across the MCMC iterations.
2. **Autocorrelation Plots:** Figure 10 shows the autocorrelation functions for each parameter.

These figures help to visually verify that the chains have reached stationarity and that the samples are nearly uncorrelated at high lags.

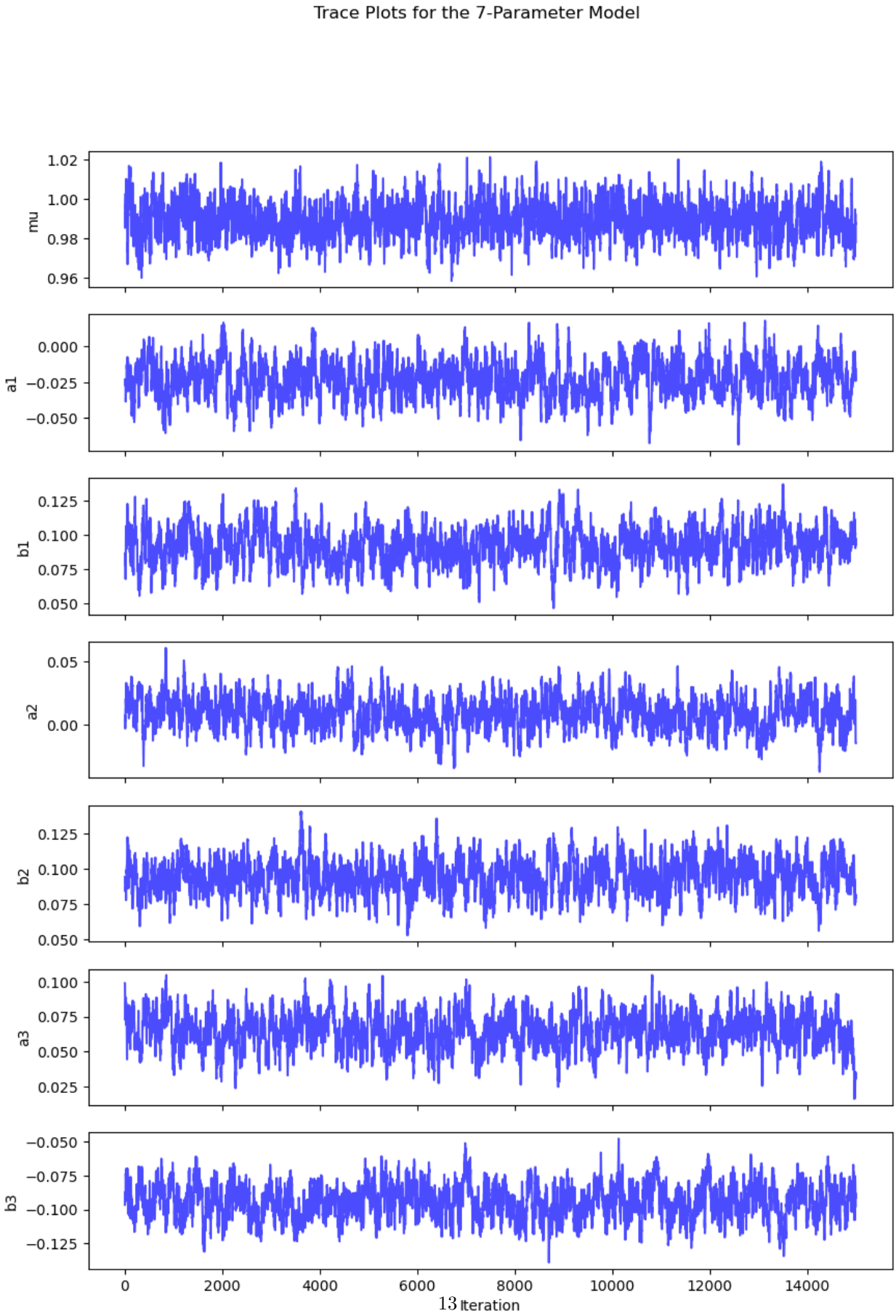
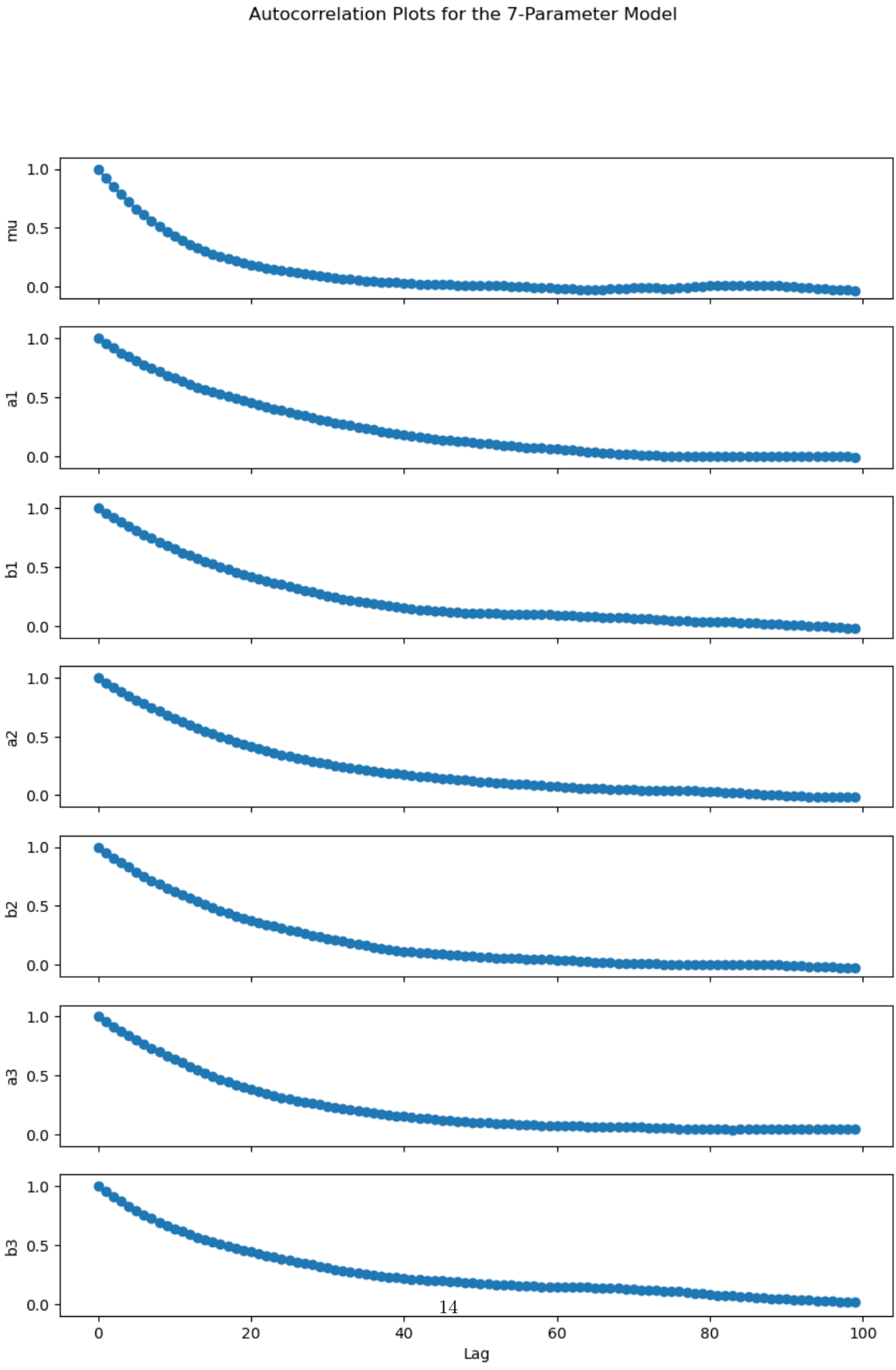


Figure 9: Trace plots for the seven parameters of the 7-parameter model.


Autocorrelation Plots for the 7-Parameter Model
mu
a1
b1
a2
b2
a3
b3
1.0
0.5
0.0
0
20
40
60
80
100
Lag

Conclusions

The seven-parameter model for Lightcurve 7 provides a good representation of the data. The 21 pairwise scatter plots illustrate the structure of the posterior distribution and reveal any correlations among the parameters. The overplotted best-fit models show excellent agreement with the folded data, and the convergence diagnostics (trace and autocorrelation plots) visually confirm that the MCMC sampler has reached convergence.

References

[1] Data generation code from `data.py` provided in NYU 2025 Spring PHYS-GA-2059 Special Topics, David Hogg.

[2] Hogg, D. W. and Villar, S. (2021). *Fitting very flexible models: Linear regression with large numbers of parameters.* arXiv:2101.07256.

[3] Hogg, D. W., Bovy, J., and Lang, D. (2010). "Data analysis recipes: Fitting a model to data." arXiv:1008.4686.

[4] Hogg, D. W. (2025). "Lectures on Data Analysis."

[5] Wikipedia contributors. (n.d.). "Sidereal year." Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Sidereal$_{y}ear$.

[6] Wikipedia contributors. (n.d.). "Bootstrapping (statistics)." Wikipedia, The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Bootstrapping$_{(}statistics$).

[7] Wikipedia contributors. (n.d.). "Bayesian Information Criterion." Wikipedia, The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Bayesian$_{i}nformation_{c}riterion$.

[8] Wikipedia contributors. (n.d.). "Cross-validation (statistics)." Wikipedia, The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Cross-validation$_{(}statistics$).

[9] Wikipedia contributors. (n.d.). "Markov chain Monte Carlo." Wikipedia, The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Markov$_{c}hain_{M}onte_{C}arlo$.

[10] Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. (n.d.). "emcee: The MCMC Hammer." Available at: https://emcee.readthedocs.io/en/stable/.

[11] Wikipedia contributors. (n.d.). "Interpolation." Wikipedia, The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Interpolation.