Draft version October 18, 2025
Typeset using LaTeX **preprint** style in AASTeX631

Bayesian Inference and MCMC Methods in Astrophysics

Agastya Gaur
University of Illinois at Urbana-Champaign

ABSTRACT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Keywords: Astrophysics, Astrostatistics, Bayesian Statistics, Markov Chain Monte Carlo, Big Data

1. INTRODUCTION

1.1. *The History of Astrostatistics*

In the 4th century BC, Hipparchus, attempting to estimate the length of a year, found the middle of the range of a scattered set of Babylonian solstice measurements (Feigelson & Babu 2004). An achievement for the time, Hipparchus's measurement marked the beginning of what would become a long-standing marriage between astronomy and statistics. In the centuries to come, a number of breakthroughs in astrostatistics followed. Notably, Tycho Brahe in the late 1500s made repeated positional measurements of stars using naked-eye observations. The data were so precise that their modeling led to Kepler's new laws of planetary motion (Leavesley & Tárnok 2018). Furthermore, in the 1770s, Laplace rediscovered Bayesian statistics, and over the next decade he expanded upon the theory, using it to modernize Newton's theory of gravity (Stigler 1975).

The biggest advance in astrostatistics before the era of computing was in 1805 when Legendre published the method of least squares regression to model the orbit of comets (Feigelson & Babu 2004). He theorized that the model best fit to a set of data was one that minimized the sum of the squares of the errors. Though Legendre did not provide a formal proof of the method, regarding it only as a convenient trick, later works by Robert Adrain developed formal mathematical proofs of the method (Merriman 1877). In 1809, Gauss published his own work on least squares, using it to calculate the orbit of the dwarf planet Ceres. Controversially, he also insisted that he had discovered the method years before Legendre (Stigler 1981). Through its impact, least squares regression has cemented itself in history as one of the most important leaps in astrostatistics.

The recurring theme is clear: progress in astronomy often hinges on solving problems of statistical estimation. By the end of the century, astronomy had firmly established itself as a quantitative science, driven by the refinement of statistical methods to identify regularities in scattered measurements, fitting orbital models, and quantifying uncertainty in the presence of noise.

1.2. *The Impact of Astrophysics*

Later years brought two developments that reshaped the relationship between astronomy and statistics: the rise of physics as the foundation of astronomy and the advent of computing, which enabled unprecedented scales of data analysis. As astronomy grew increasingly intertwined with the theories of physics, the field transformed into what we now call astrophysics. Though a niche field called statistical astronomy persisted, the majority of astronomers made insufficient use of statistics in their work (Feigelson & Babu 2004). The focus shifted to deriving physical models from first principles, and statistical methods were often seen as secondary or even unnecessary. Hubble (1930) determined the fit for the light curve of elliptical galaxies by trial-and-error instead of regression. Zwicky (1937) first observed dark matter using a curve fitted only by eye.

The disdain for statistics stemmed from most astrophysicists' stubborn adherence to Newtonian determinism. Physics was regarded as the fundamental law of nature and an elegant basis for astronomy, while statistics was seen as rough, approximate, and imperfect. Statistics also flourished in the social sciences, further alienating it from astrophysics.

However, statistics would not be kept away from astronomy for long. As computing machines developed, astronomers increasingly adopted new tools for both calculation and simulation. In the 1920s, one of the earliest applications appeared in the production of lunar tables. Previously, astronomers calculated the position of the Moon using complex, error-prone methods that required extensive manual computation (Duncombe 1988). By the 1930s, however, Comrie (1932) demonstrated how punch card computing machines could automate the process, making lunar ephemerides faster and more reliable. From then on to the 1970s, Comrie's work was continued by Wallace Eckert, who improved the punch-card calculations using IBM computers (Olley 2018). Like lunar calculations, galactic simulations were also performed with computational devices as early as the 1940s. Holmberg (1940) modeled gravitational interactions using lightbulbs to represent galaxies, demonstrating how spiral structures could emerge. These analog demonstrations laid the groundwork for digital computer N-body simulations in the 1970s, such as Toomre & Toomre (1972), who explained tidal tails and bridges in interacting galaxies.

With these advances in computing came a natural resurgence of statistical methods in astronomy. Given the ability to automate calculations, handle larger datasets, and simulate complex systems, statistical analysis became indispensable. In the mid-20th century, the growth of galaxy surveys encouraged quantitative modeling of structure and dynamics. Early work such as Lynden-Bell (1967) applied statistical mechanics to stellar systems, laying the foundation for the study of galaxy formation and equilibrium. By the 1970s, statistics was increasingly recognized as a distinct methodological pillar of astronomy. Peebles (1973) systematically catlogued and analyzed extragalactic objects using power spectra and correlation functions, pioneering the statistical study of large-scale structure.

While the early history of the field was dominated by statistical reasoning and theory, the growth of physics and digital computation broadened this into what we now call quantitative analysis (QA). Quantitative analysis thus represents the merging of three traditions that once stood apart: the deductive rigor of physics, the inferential power of statistics, and the scalability of computation. In modern astronomy, progress often relies not on one of these strands in isolation but on their integration. QA therefore serves as both a methodological framework and a philosophy of practice, emphasizing reproducibility, uncertainty quantification, and the ability to extract physical meaning from complex data.

1.3. *The Data Deluge*

Today, astrophysics sits in a universe of complex statistical problems that demand new quantitative approaches and more computing power with each passing day. These are compounded by an unprecedented era of astronomical data generation in the 21st century. Sky surveys like Gaia DR3 alone provide astrometry and photometry for nearly two billion stars, plus more than ten million variable sources (Gaia Collaboration et al. 2023). The nineteenth data release of the Sloan Digital Sky Survey collected spectroscopic data from over 6 million objects (Collaboration et al. 2025). Advances in CCD detectors will see data from sky surveys increase in the next decade from gigabytes to terabytes today, and possibly to petabytes in the near future. The same trend can be seen in data from NASA's Solar Dynamics Observatory, which now generates over a terabyte of data per day, and the Rubin LSST, generating close to 30 terabytes per day (Borne 2009). Compared to the Henry Draper Catalogue (Cannon & Pickering 1918)—a century-old counterpart that cataloged roughly 200,000 stars—the explosion in data is striking.

The leap from hundreds of thousands of stars in the Henry Draper Catalogue to billions in Gaia represents more than a change in scale: it is a qualitative transformation in what science becomes possible. With the Draper Catalogue, astronomers could classify stellar spectra and trace broad patterns in stellar populations. With Gaia, it is now possible to reconstruct the full three-dimensional structure and kinematics of the Milky Way, identify hypervelocity stars, and test theories of Galactic evolution with unprecedented detail. Where older surveys allowed the identification of a few rare stellar types, modern surveys allow systematic searches for extreme outliers across billions of objects, transforming the statistical character of astronomy.

This data deluge makes QA indispensable. Large-scale surveys now span the entire electromagnetic spectrum, from radio (e.g., LOFAR, ALMA) to X-ray and gamma-ray observatories such as Chandra and Fermi. Multi-messenger astronomy adds yet another layer, with gravitational waves detected by LIGO/Virgo and high-energy neutrinos from IceCube (Abbasi et al. 2023). Time-domain surveys such as ZTF and the upcoming LSST produce streams of transient and variable sources, producing large data volumes and data rates. Each dataset has distinct noise properties, resolutions, and systematic biases, making multi-wavelength integration a formidable statistical task. The ability to extract meaningful insights from these massive datasets is crucial for advancing our understanding of the universe.

1.4. *Statistical Challenges in Modern Astrophysics*

Two recurring types of statistical challenges emerge across astrophysics. The first challenge is that noisy, incomplete, and often degenerate data have uncertain theoretical statistical distributions. In addition, the number, complexity, and degeneracy of physical parameters poses major challenges (Schafer 2015). For example, in exoplanet studies, radial velocity measurements can only determine a planet's minimum mass because they do not contain orbital inclination (Lovis & Fischer 2010). In cosmology, measurements of the cosmic microwave background couple the effects of the Hubble constant, the amount of ordinary matter, and dark energy, so isolating any one factor requires prior assumptions about the others (Christensen & Meyer 2022). Even within galaxies, rotation curve studies must weight the balance between visible stars and invisible dark matter. Each of these cases requires careful statistical treatment to avoid misleading conclusions.

The second challenge is that the large volume of data and model complexity creates equally daunting problems of computing time and power. As noted above, the Rubin Observatory LSST will generate tens of terabytes of imaging data per night (Borne 2009), while Gaia has already released petabyte-scale catalogs (Gaia Collaboration et al. 2023). Brute-force exploration of parameter spaces is simply impossible at these scales. Efficient algorithms and scalable statistical methods are required to render analysis computationally tractable (Huijse et al. 2014). Together, these issues create a need for QA frameworks that can handle degeneracy in complex parameter spaces, model complexities, and scale efficiently with massive datasets.

1.5. *Bayesian Inference and MCMC in Context*

In this domain, Bayesian inference via Markov Chain Monte Carlo (MCMC) methods naturally emerges as a potential solution. Bayesian inference offers a principled framework for parameter estimation in complex systems. MCMC methods provide an effective way to explore parameter spaces by sampling from posterior distributions. This has become one of the most widely used and versatile approaches (Von Toussaint 2011). Computational models are becoming far more complex, and the data being analyzed is often noisy and incomplete. Bayesian methods, with their ability to incorporate prior parameter constraints and handle uncertainty, are particularly well-suited to these challenges. MCMC methods, in particular, provide a practical way to sample from complex posterior distributions that arise in Bayesian analysis. This makes them invaluable for parameter estimation, model comparison, and uncertainty quantification.

Another reason for the appeal of Bayesian methods is their contrast with frequentist approaches. Frequentist methods, which were dominant through much of the 20th century, emphasize point parameter estimates and confidence intervals derived from repeated sampling arguments (Trotta 2008). Bayesian inference, in contrast, provides full posterior probability distributions for parameters, naturally incorporating prior information from physics or earlier observations. This framework is particularly powerful where data are sparse, noisy, and incomplete.

The wider adoption of Bayesian statistics in astronomy gained momentum in the late 20th century. Cosmologists have applied Bayesian methods to the cosmic microwave background to extract cosmological parameters from noisy sky maps (Tegmark et al. 1997). In exoplanet science, Bayesian inference became common in the 1990s and 2000s for modeling radial velocity curves and transit signals in the era of larger and more precise datasets (Gregory 2005). Pulsar timing, supernova cosmology, and gravitational lens modeling similarly saw wider use of Bayesian methods. The trend reflects a broader recognition that many of astronomy's hardest problems demand not just point estimates, but principled uncertainty quantification.

We review here Bayesian inference and MCMC within an astrophysical perspective because of their flexibility, principled uncertainty quantification, and growing ubiquity across astrophysics. In Section II, the methodology section develops a working foundation: here we review Bayesian statistics, priors, and likelihood construction in realistic astronomical settings. We cover toy examples before escalating to domain-relevant formulations. Then, we introduce Monte Carlo methods and build to Markov Chain Monte Carlo, deriving the Metropolis family and related samplers, providing step-by-step Python implementations on simple problems.

Section III presents three focused case studies that illustrate how Bayesian–MCMC pipelines advance frontiers in different subfields. For exoplanet direct detection, the section outlines the observational context and the central challenge of disentangling planetary signals from stellar activity and disk structure in direct imaging. It then outlines joint stellar–planet modeling with Gaussian processes and Bayesian classification frameworks for robust detections and non-detections, highlighting benefits and failure modes in low SNR regimes. For CMB parameter estimation, we consider geometric degeneracies in the power spectrum and shows how MCMC accelerators and parallelizable frameworks enable efficient exploration and evidence calculations, including tradeoffs between sampler sophistication and wall-clock efficiency. For gravitational-wave inference, we consider waveform fitting in high-dimensional parameter spaces, discuss the cost of likelihood evaluations, and motivate gradient-informed MCMC strategies that reduce computation without compromising accuracy.

Finally, in Section IV and V we compare cases where Bayesian–MCMC excels with those where complementary methods are more appropriate, identify methodological gaps revealed by the case studies, and outline opportunities for future work in astrophysics and related fields that face similar statistical and computational challenges.

2. METHODOLOGY

2.1. *Bayesian Preliminaries*

The aim of Bayesian statistics is simple: determine $P(H|D)$, or the probability of a hypotheses H being true given data D. A hypothesis is any statement that can be true or false, and data is any information that can be used to evaluate the hypothesis. For example, imagine rolling a die. A hypothesis H would that the die roll is a 3. The data D is the result of the die roll.

Hypotheses live in the *hypothesis space*, which is the set of all possible hypotheses of a system (Brewer 2018). Going back to the die example, the hypothesis space is $\{1, 2, 3, 4, 5, 6\}$. The hypothesis space will also have a probability distribution, or a *prior*, written as $P(H)$. The prior is the probability of each hypothesis being true before seeing any data. For a fair die, the prior is uniform. It's $\frac{1}{6}$ for all H in the hypothesis space. In other words, $P(H)$ is the probability of the hypothesis being true.

The data D also has a probability distribution called the *evidence*, $P(D)$ (Brewer 2018). The evidence is the probability of seeing the data before knowing anything about the hypothesis. In the die example, if you roll a 3, then $P(D)$ is $\frac{1}{6}$ because there is a $\frac{1}{6}$ chance of rolling a 3 on a fair die. The evidence lives in the *data space*, which is the set of all possible data of a system. In the die example, the data space is also $\{1, 2, 3, 4, 5, 6\}$.

A *likelihood*, $P(D|H)$, is the probability of seeing the data assuming that the hypothesis is true. In the die example, if H is that the die roll is a 3, then $P(D|H)$ is 1 if the die roll is a 3 and 0 otherwise. Using the prior, evidence, and likelihood to calculate $P(H|D)$, which is the probability of a hypothesis being true given the data. This is called the *posterior.* In the die example, if you roll a 3, then $P(H|D)$ is 1 if H is that the die roll is a 3 and 0 otherwise.

The framework for finding $P(H|D)$ is called Bayes' Theorem, and it forms the center of Bayesian inference. It can be derived using two rules (Cox 1946). Firstly, the probability that a hypothesis is true and the probability that it is not true add up to 1:

$$P(H) + P(\tilde{H}) = 1. \tag{2.1}$$

The second rule is the product rule, which states:

$$P(H)P(D|H) = P(D)P(H|D). \tag{2.2}$$

This can be trivially rearranged to give Bayes' Theorem:

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}. \tag{2.3}$$

In other words, given the prior, evidence, and the likelihood, you can calculate the posterior $P(H|D)$.

2.1.1. *Example: The Double-Headed Coin*

The previous example of a die roll was trivial. The data, the result of the die roll, completely determined the hypothesis. Bayesian statistics becomes more useful when the data is incomplete, as is often the case in Astrophysics.

To demonstrate this, consider the following setup. You have 5 coins, four of which are fair, and one of which is double-headed. You pick a coin at random and flip it, and it lands heads. *What is the probability that you picked the double-headed coin?*

The first step is to determine H and D from the hypotheses and data spaces. The hypothesis space is $\{\text{Picked Fair}, \text{Picked Double-Headed}\}$. For the sake of conciseness, it can be written as $\{\text{fair}, \text{double}\}$. For this problem, we hypothesize that the double-headed coin was picked. So, H is 'double'. The data space is $\{\text{heads}, \text{tails}\}$, and for this problem, D is 'heads'.

Next, we determine the prior and evidence. There are 4 fair coins and 1 double-headed coin, so it is easy to find for the prior that $P(\text{fair}) = \frac{4}{5} = 0.8$ and $P(\text{double}) = \frac{1}{5} = 0.2$. The evidence is more complex to find in this case. Since the chance of flipping heads or tails includes the case that you picked up the double headed coin, the evidence cannot simply be 50-50. We must calculate the probability of getting heads and tails across all the coins:

$$P(\text{heads}) = \frac{4(0.5) + 1(1)}{5} = 0.6$$
$$P(\text{tails}) = \frac{4(0.5) + 1(0)}{5} = 0.4$$

Note the implicit rule for any space:

$$\sum_n P(x_n) = 1, \tag{2.4}$$

where x_n is a value in a space.

The final step before solving is to find the likelihood, $P(\text{heads}|\text{double})$. This is trivially 1, as you can only get heads from the double-headed coin. Now, we can solve for the posterior:

$$P(\text{double}|\text{heads}) = \frac{P(\text{double})P(\text{heads}|\text{double})}{P(\text{heads})} = \frac{0.2}{0.6} = \boxed{0.\bar{3}}$$

Before incorporating the data, the probability of picking the double headed coin was 0.2, but by using Bayesian statistics, we were able to "learn" from the data and increase the probability to $0.\bar{3}$. This is a simple example, but it demonstrates the power of Bayesian statistics in the face of incomplete data.

2.2. *Bayesian Statistics in Astrophysics*

In an astrophysical context, Bayes' Theorem is used to calculate the probability of *parameters*, θ, of a model rather than a hypothesis (Brewer 2018).

$$P(\theta|D) = \frac{P(\theta)P(D|\theta)}{P(D)}. \tag{2.5}$$

Take, for example, measurements of the radial velocity of a star over time. When a start hosts an orbiting planet, both bodies orbit a common center of mass. This induces a 'wobble' in the star's motion which is detected as periodic Doppler shifts from the star's spectrum, also known as the *radial velocity.* The radial velocity $v_r(t)$ of a star with a single planet in a strictly circular orbit can be modeled by the following equation:

$$v_r(t) = K \sin\left(\frac{2\pi t}{T} + \phi\right), \tag{2.6}$$

where K is the velocity semi-amplitude, T is the orbital period, and ϕ is a phase offset. Thus, the parameters of this model are $\theta = \{K, P, \phi\}$. This is a simplified model derived from Lovis & Fischer (2010). Some parameters, such as eccentricity, system velocity, and argument of periapsis, have been omitted for simplicity.

One could measure the radial velocity of a star over a certain time period and obtain the data shown in Figure 1.

Bayes' Theorem can be used to determine the posterior distribution of the parameters using the noicy, incomplete data. In this case, Equation 2.3 becomes:

$$P(K, T, \phi|D) = \frac{P(K, T, \phi)P(D|K, T, \phi)}{P(D)}. \tag{2.7}$$

To start, we must determine the prior, $P(K, T, \phi)$. For simplicity, we can assume that the parameters are independent, so the prior can be written as:

$$P(K, T, \phi) = P(K)P(T)P(\phi). \tag{2.8}$$

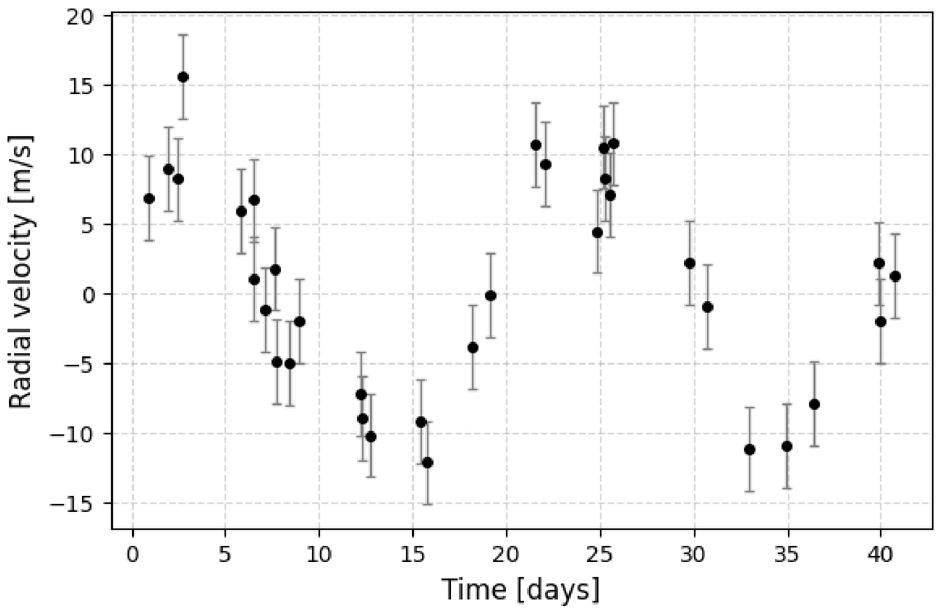


Figure 1. Simulated radial velocity data of a star with a single orbiting planet. Each point has an error of three standard deviations. The data is noisy and incomplete, making it difficult to determine the parameters of the model.

Since we know nothing about the priors, it is reasonable to assume that they are uniformly distributed in a certain range. Looking at the data, we can see that the velocity semi-amplitude K is between 5 and 15 m/s, the period T is between 20 and 30 days, and the phase offset ϕ is between 0 and 2π. Thus, the priors are:

$$P(K) = \begin{cases} \frac{1}{10} & 5 < K < 15 \\ 0 & \text{otherwise} \end{cases} \qquad P(T) = \begin{cases} \frac{1}{10} & 20 < T < 30 \\ 0 & \text{otherwise} \end{cases} \qquad P(\phi) = \begin{cases} \frac{1}{2\pi} & 0 < \phi < 2\pi \\ 0 & \text{otherwise} \end{cases}$$

and the combined prior is then:

$$P(K, T, \phi) = \begin{cases} \frac{1}{200\pi} & 5 < K < 15, 20 < T < 30, 0 < \phi < 2\pi \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

Next, we must determine the likelihood, $P(D|K, T, \phi)$, or the probability of seeing the data given the parameters. To do this, we calculate the probability of seeing each data point given what the model would predict for that data point. We can assume the model prediction is the mean of a normal distribution of possible points, and the error of each data point is the standard deviation. Using this, we can calculate the probability of obtaining a data point d_i at time t_i. Multiplying the probabilities of all the data points gives the probability of seeing the full dataset, which is the likelihood. Mathematically, this is:

$$P(D|K, T, \phi) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(D_i - v_r(t_i; K, T, \phi))^2}{2\sigma_i^2}\right), \tag{2.10}$$

Substituting values for each parameter gives the likelihood of seeing the data given the parameters.

Finally, we must determine the evidence, $P(D)$. It's obvious that when integrating the posterior over the entire parameter space, it must equal one, as the parameters are certain to take on some value within the space. Thus, the evidence can be found by rearranging the equation:

$$\begin{aligned} 1 &= \iiint P(K,T,\phi|D)dKdTd\phi \\ 1 &= \iiint \frac{P(K,T,\phi)P(D|K,T,\phi)}{P(D)}dKdTd\phi \\ \therefore P(D) &= \iiint P(K,T,\phi)P(D|K,T,\phi)dKdTd\phi \end{aligned} \tag{2.11}$$

Note that this integral is analytically intractable, so it must be calculated numerically.

Using the data in Figure 1, we can calculate the posterior distribution of the parameters using the code given in Appendix A. Note that the code uses the log-prior and log-likelihood. These are used to prevent any underflow errors in the program. These can be calculated by taking the natural logarithm of the prior and likelihood, respectively. The posterior can then be calculated by adding the log-prior and log-likelihood, and then exponentiating the result, then normalizing with the evidence.

$$P(K,T,\phi|D) = \frac{e^{\log(P(K,T,\phi))+\log(P(D|K,T,\phi))}}{P(D)} \tag{2.12}$$

The program evaluates the posterior on a grid of 100 points for each parameter, giving a total of $100^3 = 1,000,000$ points. The posterior is also cumulatively summed at every evaluated point, giving a numerical estimate of the evidence for normalization. Finally, the marginal posteriors of each parameter are calculated from the total posterior by integrating over the other two parameters:

$$P(K|D) = \iint P(K,T,\phi|D)dTd\phi \tag{2.13}$$

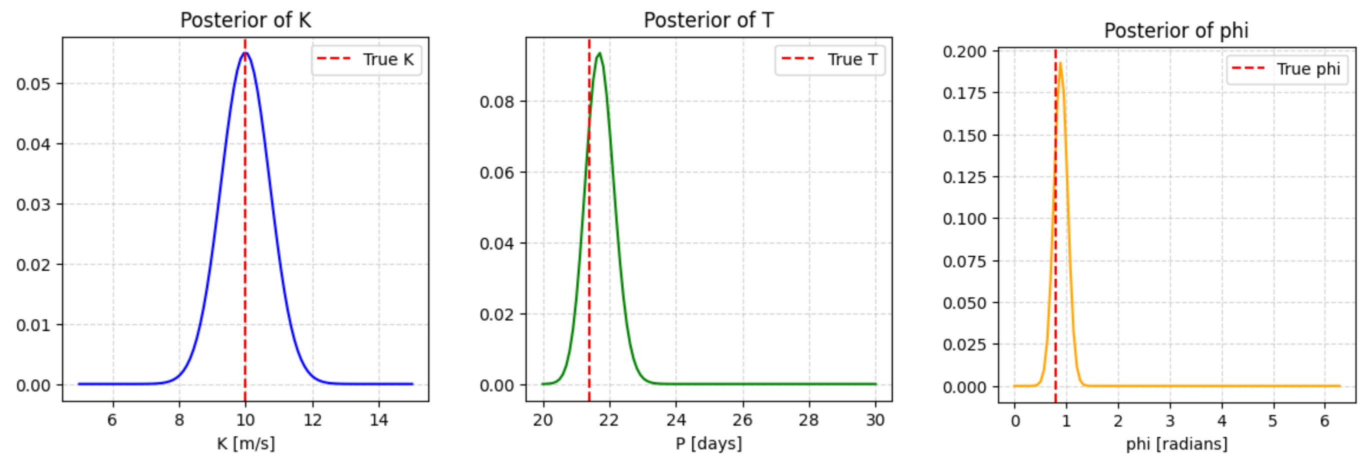
$$P(T|D) = \iint P(K,T,\phi|D)dKd\phi \tag{2.14}$$

$$P(\phi|D) = \iint P(K,T,\phi|D)dKdT \tag{2.15}$$

The results of the code are shown in Figure 2.

In reality, we will never know the exact parameters of a system, but posterior distributions give us a close estimate. To quantify how close the estimate is, we will compare the posterior distributions to the true parameters, which are $K = 10$ m/s, $T = 25$ days, and $\phi = \frac{\pi}{4}$. It is clear to see that the true values are within one standard deviation of the mean of each posterior distribution, indicating that the Bayesian method was successful in estimating the parameters.

Figure 3 shows the true model along with the mean prediction from the posterior distributions. Thus, we can apply Bayes' Theorem to predict the parameters for astrophysical models. However, this method is computationally expensive. Computing this numerically, we would have to iterate



(a) $\mu = 9.99$ m/s $\quad \sigma = 0.73$ m/s (b) $\mu = 21.72$ days $\quad \sigma = 0.44$ days (c) $\mu = 0.89$ rad $\quad \sigma = 0.13$ rad

Figure 2. The marginal posterior distributions of the parameters K, T, and ϕ given the data in Figure 1. These posteriors are normalized with a numerically calculated evidence.

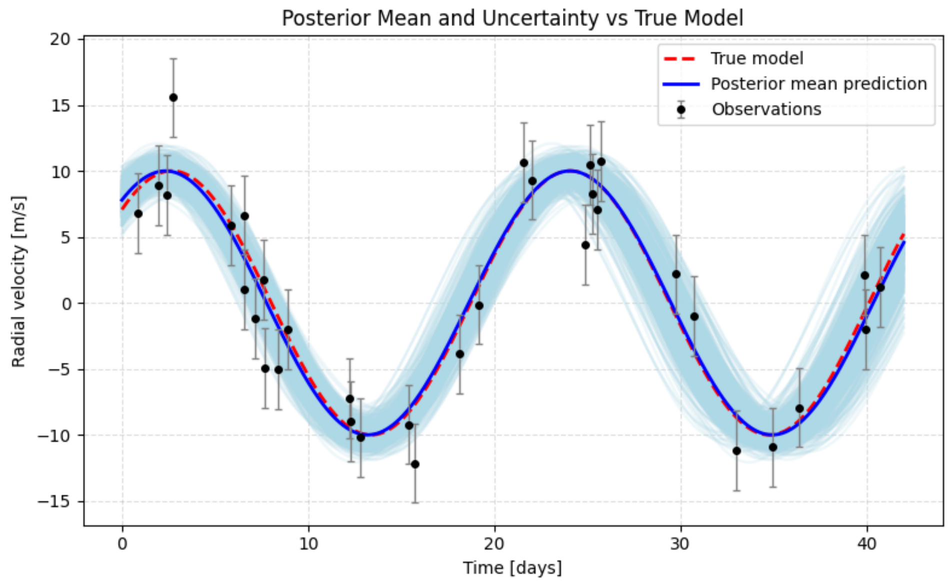


Figure 3. The radial velocity data from Figure 1 along with the mean prediction from the posterior distributions in Figure 2. The shaded region represents one standard deviation from the mean prediction.

through all combinations of all values of each parameter. With n parameters and m possible values for each parameter, this algorithm would have a time complexity of $O(m^n)$, which is infeasible for even small n and m. To solve this problem, we can use *Monte Carlo methods*.

2.3. *Monte Carlo Sampling*

The primary computational challenge in brute force Bayesian inference arises from evaluating the posterior probability at every point in the parameter space. One way to reduce this cost is to sample only a subset of points, distributed according to the posterior itself. These points are called *draws*. If, by some method, a set of draws were obtained, it would be possible to calculate expectation values

x	Description
θ_i	Expectation value of parameter θ_i
θ_i^2	Variance of parameter θ_i
$f(\theta_1, \theta_2, \ldots, \theta_n)$	Expectation value of any function of the parameters
$\mathbb{1}(\theta_i = a)$	Probability that $\theta_i = a$
$\mathbb{1}(a \leq \theta_i \leq b)$	Probability that θ_i is in the range $[a, b]$
$\mathbb{1}(\text{condition})$	Probability that the condition is true

Table 1. Forms of x to calculate expectation values of different quantities.

from the explicit posterior using only the sampled points (Von Toussaint 2011). This is the idea behind *Monte Carlo Sampling*, which can be used to approximate integrals and expectations that would otherwise require explicit evaluation over a continuous space.

If we know a probability density explicitly, the expectation value of some variable x can be calculated as:

$$\langle x \rangle = \int x P(\theta) dx. \tag{2.16}$$

Using a set of N values of x drawn using Monte Carlo, this can be approximated as:

$$\langle x \rangle \approx \frac{1}{N} \sum_{i=1}^{N} x_i, \tag{2.17}$$

which generalizes naturally to higher-dimensional parameter spaces, though convergence can become slower as dimensionality increases.

x is a function of the parameters, and can be defined arbitrarily to calculate the expectation value of any value of interest. Given a probability density $P(\theta_1, \theta_2, \ldots, \theta_n)$, some forms of x are listed in Table 1.

The usefulness of Monte Carlo sampling becomes apparent when we revisit the computation of marginal posteriors. In the previous example, we used the posterior to calculate the marginal posteriors of each parameter. In the brute force method, this was found by integrating the posterior over all other parameters. Monte Carlo makes this simpler. If we have a set of N draws from the posterior of a parameter space with parameters, $\{\theta_1, \theta_2, \ldots, \theta_n\}$, we can obtain draws for a certain parameter θ_i by simply ignoring all other parameters. A simple histogram of the draws of θ_i then gives the marginal posterior of θ_i. This is much simpler than integrating over all other parameters, especially in high-dimensional parameter spaces.

2.4. *Markov Chain Monte Carlo*

Before we can use Monte Carlo sampling, we must first obtain draws from the posterior. *Markov Chain Monte Carlo* (MCMC) is one of the many methods used to obtain these draws, and in modern astrophysics, has become a standard tool for Bayesian inference. Using MCMC, the posterior of even a high-dimensional parameter space can be numerically mapped without using unfeasible amounts of computing power (Trotta 2008). MCMC does this by creating a sequence of draws in the form of a *Markov Chain*.

2.4.1. *Markov Chains*

A Markov Chain is a sequence of random variables, $x_0, x_1, \ldots, x_n$, where the probability of each variable only depends on the previous variable. This is known as the *Markov Property*, and can be written mathematically as:

$$P(x_{n+1}|x_n, x_{n-1}, \ldots, x_0) = P(x_{n+1}|x_n). \tag{2.18}$$

To create a Markov chain, we must first decide an initial value x_0 using an initial probability distribution $P_0(x_0)$. After this, the probability distribution of the next value is evolved from the previous probability distribution using a *transition probability* $T_n(x_n, x_{n+1})$. The next value is then drawn using the new probability density.

$$P(x_{n+1}) = P(x_n)T_n(x_n, x_{n+1}) \tag{2.19}$$

This process can be repeated as many times as necessary.

If the transition probability does not depend on the point in the chain, the chain is *homogenous*, or *stationary*. We can then write the transition probability as $T(x_n, x_{n+1})$ (Neal 1993).

Markov chains can be shown to converge to a *stationary distribution* $P^*(x)$, which is independent of the initial distribution (Trotta 2008). A stationary distribution, once reached, does not change as the chain evolves further.

Markov chains can also be *ergodic*, which means that a stationary distribution can be reached from any initial distribution. A chain being ergodic also implies its irreducibility and aperiodicity (Neal 1993). Irreducibility means that it is possible to reach any point in the space from any other point, and aperiodicity means that the chain does not get stuck in cycles (Von Toussaint 2011).

In the context of Bayesian inference, the goal of MCMC is to construct a Markov chain whose stationary distribution is the unknown posterior distribution. To ensure that the chain converges to the desired posterior, the choice of transition probability $T(x_n, x_{n+1})$ is crucial. The transition rule must satisfy the condition of *detailed balance*, which guarantees that, at equilibrium, the probability flow between any two states is symmetric:

$$P(x_n)T(x_n, x_{n+1}) = P(x_{n+1})T(x_{n+1}, x_n). \tag{2.20}$$

This condition ensures that the chain does not drift away from the target distribution once it has been reached (Von Toussaint 2011).

2.4.2. *The Metropolis-Hastings Algorithm*

One of the simplest and most widely used algorithms that satisfies detailed balance is the *Metropolis algorithm* (Metropolis et al. 1953). More specifically, its generalization, the *Metropolis–Hastings algorithm* (Hastings 1970). The algorithm constructs a Markov chain by iteratively proposing new states in the parameter space and deciding whether to accept or reject them based on the target distribution. Brewer (2018) outlines the steps of the Metropolis-Hastings algorithm as follows:

1. Initialize the chain with a starting point x_0.
2. Generate a candidate point x_{n+1} from a proposal distribution $q(x_{n+1}|x_n)$ based on the current state x_n.
3. Accept or reject the proposal
4. Repeat steps 2 and 3 for a large number of iterations to generate a sequence of samples.

The algorithm depends on two things: the proposal distribution $q(x_{n+1}|x_n)$ and the acceptance criterion. Commonly, a 'random walk' proposal is used to generate candidates by adding a small random perturbation to the current state. For an n-dimensional parameter space, the random walk proposal is a symmetric (usually Gaussian) distribution, also of n-dimensions, with the current state as its mean (Von Toussaint 2011). If the proposal distribution is indeed symmetric, then the following is implied:

$$q(x_{n+1}|x_n) = q(x_n|x_{n+1}). \tag{2.21}$$

In this case, the Metropolis-Hastings algorithm simplifies to the original Metropolis algorithm (Brewer 2018). Asymmetric proposal distributions are more general, and can also be used. However, they are less common in practice.

The acceptance criterion is based on the ratio of the posterior at the proposed and current states, adjusted by the proposal distribution. This is the probability of the proposal being accepted. Mathematically, this is:

$$\alpha = \min\left(1, \frac{P(x_{n+1}|D)q(x_n|x_{n+1})}{P(x_n|D)q(x_{n+1}|x_n)}\right) = \min\left(1, \frac{P(D|x_{n+1})P(x_{n+1})q(x_n|x_{n+1})}{P(D|x_n)P(x_n)q(x_{n+1}|x_n)}\right). \tag{2.22}$$

If the proposal distribution is symmetric, we can further apply Equation 2.21 to get:

$$\alpha = \min\left(1, \frac{P(D|x_{n+1})P(x_{n+1})}{P(D|x_n)P(x_n)}\right). \tag{2.23}$$

Working with log-priors and log-likelihoods makes this even simpler:

$$\log(\alpha) = \min\left(0, \log(P(D|x_{n+1})) + \log(P(x_{n+1})) - \log(P(D|x_n)) - \log(P(x_n))\right). \tag{2.24}$$

This criterion accepts any proposal that increases the posterior probability, while proposals that decrease it are accepted with a probability proportional to the decrease. If a proposal is rejected, the current state is counted again. This allows the chain sample high-probability regions more frequently while still having the ability to escape local maxima and explore the full parameter space.

The choice of proposal distribution can also significantly affect the efficiency of the algorithm. If the proposal steps are too small, the chain will explore the parameter space slowly, leading to high autocorrelation between samples. Conversely, if the steps are too large, many proposals will be rejected, also resulting in inefficient sampling. Tuning the proposal distribution to balance exploration and acceptance rates is often necessary for optimal performance (Von Toussaint 2011). A common heuristic is to adjust the proposal distribution to achieve an acceptance rate between 0.2 and 0.5, depending on the dimensionality of the parameter space (Gelman et al. 1997).

A consideration one may take when using the Metropolis-Hastings algorithm is the *burn-in* period. The initial samples of the chain may not be representative of the target distribution, especially if the starting point is far from high-probability regions. To mitigate this, it is common practice to discard a certain number of initial samples, known as the burn-in period, before using the remaining samples for inference (van Ravenzwaaij et al. 2018). This allows the algorithm to forget its initial state and generate samples that are more representative of the posterior distribution.

The Metropolis-Hastings algorithm outputs a Markov chain of samples that approximate the target posterior distributiion. Since the probability is only calculated for proposed points instead of the full parameter space, the algorithm is much more computationally efficient than brute force methods. The time complexity is reduced to $O(n * m)$, where n is the number of samples drawn and m is the number of walks, which, unlike brute-force Bayesian inferencing, is feasible for even high-dimensional parameter spaces.

2.5. *MCMC in Astrophysics*

Revisiting the radial velocity example, we can use MCMC to obtain draws from the posterior distribution of the parameters $\{K, T, \phi\}$ from Equation 2.6 given the data in Figure 1. Using the Metropolis-Hastings algorithm, 10 chains were run, each with 12,000 samples. The first 1,000 samples of each chain were discarded as burn-in. The code used to run the MCMC is given in Appendix B.

The trace plots of the chains are shown in Figure 4. These are visualizations of the Markov chains for each parameter. The chains appear to be well-mixed and stationary, indicating that they have converged to the target distribution. Plotting the marginal posterior distributions of each parameter gives Figure 5.

Comparing these posteriors to the ones obtained using the brute-force method, we can see that they are very similar. Table 2 summarizes the comparison between the true parameter values, the estimates from the brute-force method (Figure 2), and the estimates from the MCMC method (Figure 5). The two algorithms are compared using the mean and standard deviation of each posterior distribution, as well as the Z-score of the true value from the mean of each posterior distribution. The Z-score is a measure of how many standard deviations a value is from the mean, and is calculated as:

$$Z = \frac{X - \mu}{\sigma}, \tag{2.25}$$

where X is the value being compared, μ is the mean of the distribution, and σ is the standard deviation of the distribution. A Z-score of 0 indicates that the value is equal to the mean, while a Z-score of 1 indicates that the value is one standard deviation above the mean. A Z-score of -1 indicates that the value is one standard deviation below the mean. A Z-score within the range of -2 to 2 is generally considered to be acceptable, as it indicates that the value is within two standard deviations of the mean.

From Table 2, we can see that both methods give very similar results. The Z-scores of the true values from the means of each posterior distribution are all within 1, indicating that both methods were successful in estimating the parameters. The MCMC method, however, was far more computationally efficient. As shown in the code in Appendix A, the brute-force method had to evaluate the posterior at $100^3 = 1,000,000$ points in the parameter space. The MCMC method, given in Appendix B, only

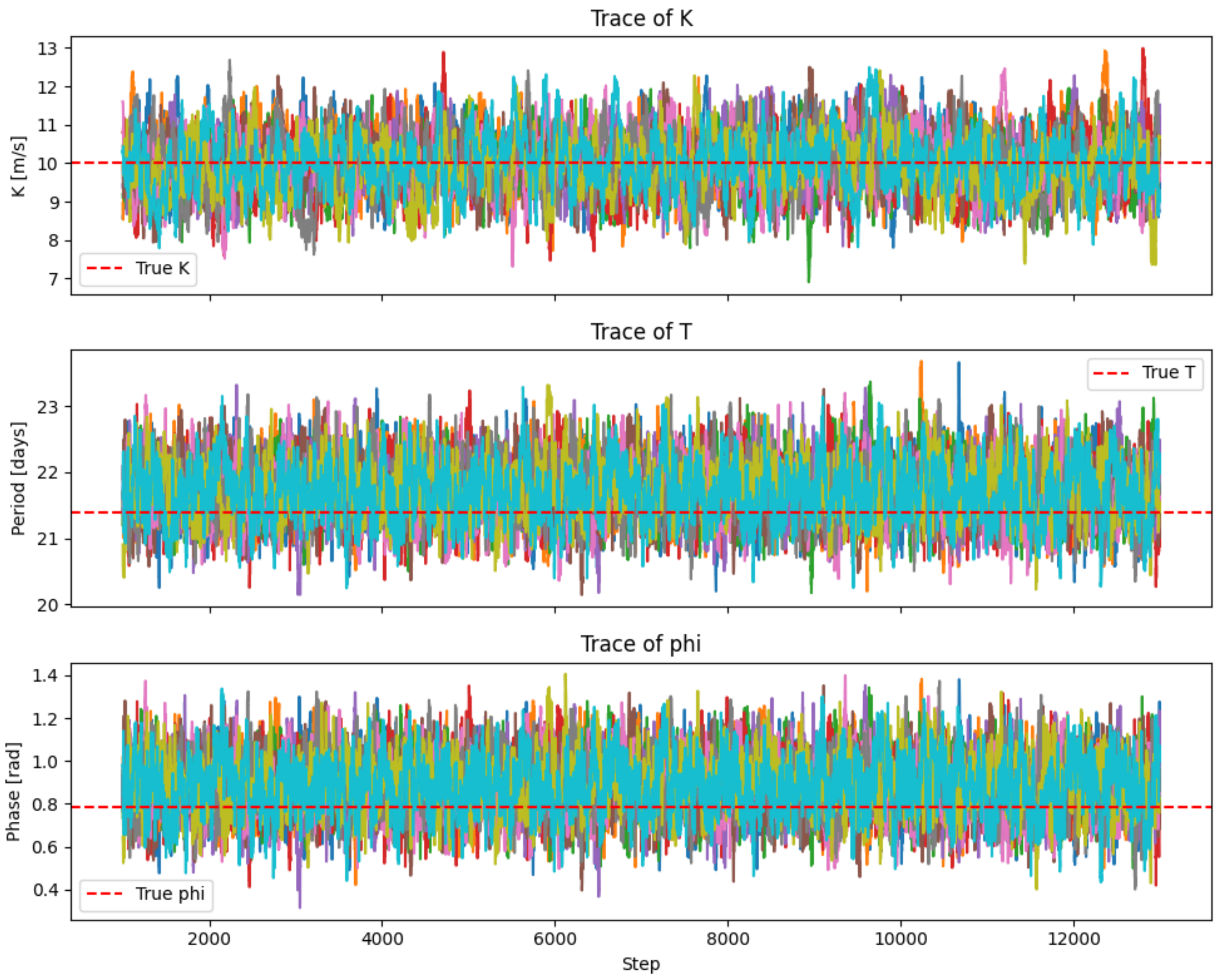
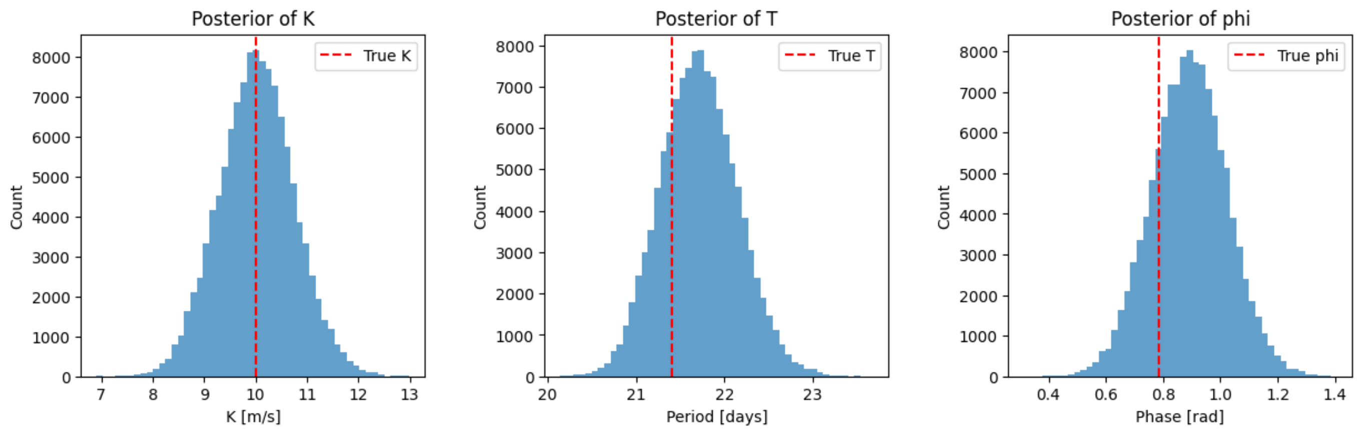


Figure 4. Trace plots of the Markov chains for each parameter. A total of 10 chains were run, each with 12,000 samples. The first 1,000 samples of each chain were discarded as burn-in.

Parameter	True	Brute-Force	Brute-Force Z-score	MCMC	MCMC Z-score
K (m/s)	10.00	9.99 ± 0.73	-0.01	10.02 ± 0.73	0.03
T (days)	21.4	21.72 ± 0.44	0.68	21.71 ± 0.43	0.66
ϕ (rad)	0.25π	0.89 ± 0.13	0.77	0.89 ± 0.13	0.77

Table 2. Comparison of the true parameter values, the estimates from the brute-force method, and the estimates from the MCMC method.

had to evaluate the posterior at $10 \times (12,000 - 1,000) = 110,000$ points, a reduction of nearly an order of magnitude, to provide virtually the same results.



(a) $\mu = 10.02$ m/s $\quad \sigma = 0.73$ m/s (b) $\mu = 21.71$ days $\quad \sigma = 0.43$ days (c) $\mu = 0.89$ rad $\quad \sigma = 0.13$ rad

Figure 5. The histograms of the draws from the posterior distributions of the parameters K, T, and ϕ given the data in Figure 1.

Generalizing this to higher-dimensional parameter spaces, the computational savings of MCMC become even more apparent. In a hypothetical 10-dimensional parameter space with 100 possible values for each parameter, the brute-force method would have to evaluate the posterior at $100^{10} = 10^{20}$ points, which is infeasible. The MCMC method, on the other hand, would still only have to evaluate the posterior at N points, where N is the number of samples drawn, which is feasible for even high-dimensional parameter spaces.

3. CASE STUDIES

Bayesian inference and MCMC has found its place in contemporary astrophysical challenges. The adaptability of the Bayesian–MCMC framework makes it useful in tackling problems of parameter estimation, model selection, and uncertainty quantification across various subfields. Specifically, we will explore the detection of exoplanets through direct imaging, the estimation of cosmological parameters from the cosmic microwave background (CMB), and the inference of gravitational wave signals.

3.1. *Direct Imaging of Exoplanets*

3.1.1. *Introduction*

Exoplanets are planets that orbit stars outside our solar system, with each being a unique laboratory for studying planetary formation and evolution (Kaushik et al. 2025).

APPENDIX

A. RADIAL VELOCITY BRUTE-FORCE APPROACH

```
#%%
# --- Radial Velocity Example ---
# This script demonstrates simple Bayesian parameter estimation for a
# sinusoidal radial velocity model of the form:
# v(t) = K * sin(2 * pi * t / P + phi)

import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)  # The meaning of life for reproducibility

#%%
# --- True Parameter Values ---
K = 10.0             # Velocity semi-amplitude [m/s]
P = 21.4             # Period of the signal [days]
phi = 0.25 * np.pi   # Phase offset [radians]
sig_ob = 3.0         # Observational uncertainty [m/s]

#%%
# --- Generate Synthetic Data ---
# Simulate 34 observations across 42 days
t = np.sort(42 * np.random.rand(34))

# Observed velocities = true model + Gaussian noise
v_ob = K * np.sin((2 * np.pi * t / P) + phi) + np.random.normal(0, sig_ob,
    size=len(t))

#%%
# --- Plot Observations Only ---
plt.figure(figsize=(6, 4))
plt.errorbar(
    t, v_ob, yerr=sig_ob, fmt='o', color='black',
    ecolor='gray', elinewidth=1, capsize=2,
    label='Observations', markersize=4
)
plt.xlabel("Time [days]", fontsize=12)
plt.ylabel("Radial velocity [m/s]", fontsize=12)
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

#%%
# --- Plot Observations and True Model ---
t_dense = np.linspace(0, 42, 1000)
plt.figure(figsize=(6, 4))
plt.errorbar(
    t, v_ob, yerr=sig_ob, fmt='o', color='black',
    ecolor='gray', elinewidth=1, capsize=2,
    label='Observations', markersize=4
)
plt.plot(t_dense, K * np.sin(2 * np.pi * (t_dense / P) + phi),
        'r--', linewidth=2, label='True model')
plt.xlabel("Time [days]", fontsize=12)
plt.ylabel("Radial velocity [m/s]", fontsize=12)
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

#%%
# --- Brute-Force Bayesian Parameter Estimation ---
# Define prior, likelihood, and posterior functions

# Uniform prior over plausible parameter ranges
def log_prior(K, P, phi):
    if 5 < K < 15 and 20 < P < 30 and 0 < phi < 2 * np.pi:
        return -np.log(10 * 10 * 2 * np.pi)
    else:
        return -np.inf

# Gaussian likelihood assuming constant sigma
def log_likelihood(K, P, phi):
    v_model = K * np.sin((2 * np.pi * t / P) + phi)
    return -0.5 * np.sum(
      ((v_ob - v_model)/sig_ob)**2 + np.log(2*np.pi*sig_ob**2)
    )

#%%
# --- Evaluate Posterior on a Grid ---
K_vals = np.linspace(5, 15, 100)
P_vals = np.linspace(20, 30, 100)
phi_vals = np.linspace(0, 2 * np.pi, 100)
posterior_vals = np.zeros((len(K_vals), len(P_vals), len(phi_vals)))

norm = 0
for i in range(len(K_vals)):
    for j in range(len(P_vals)):
        for k in range(len(phi_vals)):
            posterior_vals[i, j, k] = np.exp(log_prior(K_vals[i], P_vals[j
               ], phi_vals[k]) + log_likelihood(K_vals[i], P_vals[j],
               phi_vals[k]))
            norm += posterior_vals[i, j, k]
posterior_vals /= norm

#%%
# --- Compute Marginal Posteriors and Statistics ---
posterior_K = np.sum(np.sum(posterior_vals, axis=2), axis=1)
posterior_P = np.sum(np.sum(posterior_vals, axis=0), axis=1)
posterior_phi = np.sum(np.sum(posterior_vals, axis=0), axis=0)

```

```
mean_K = np.sum(K_vals * posterior_K) / np.sum(posterior_K)
std_K  = np.sqrt(np.sum((K_vals - mean_K) ** 2 * posterior_K) / np.sum(
    posterior_K))
mean_P = np.sum(P_vals * posterior_P) / np.sum(posterior_P)
std_P  = np.sqrt(np.sum((P_vals - mean_P) ** 2 * posterior_P) / np.sum(
    posterior_P))
mean_phi = np.sum(phi_vals * posterior_phi) / np.sum(posterior_phi)
std_phi  = np.sqrt(np.sum((phi_vals - mean_phi) ** 2 * posterior_phi) / np
    .sum(posterior_phi))

#%%
# --- Plot Marginal Posterior Distributions ---
plt.figure(figsize=(4, 4))
plt.plot(K_vals, posterior_K, color='blue')
plt.axvline(K, color='r', ls='--', label='True K')
plt.xlabel("K [m/s]")
plt.title("Posterior of K")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
print(f"mu={mean_K:.2f}, sigma={std_K:.2f}")

plt.figure(figsize=(4, 4))
plt.plot(P_vals, posterior_P, color='green')
plt.axvline(P, color='r', ls='--', label='True T')
plt.xlabel("P [days]")
plt.title("Posterior of T")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
print(f"mu={mean_P:.2f}, sigma={std_P:.2f}")

plt.figure(figsize=(4, 4))
plt.plot(phi_vals, posterior_phi, color='orange')
plt.axvline(phi, color='r', ls='--', label='True phi')
plt.xlabel("phi [radians]")
plt.title("Posterior of phi")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
print(f"mu={mean_phi:.2f}, sigma={std_phi:.2f}")

#%%
# --- Compare True Model vs Posterior Predictions ---
plt.figure(figsize=(8, 5))
N_samples = 500

for i in range(N_samples):
    K_draw = np.random.choice(K_vals, p=posterior_K)
    P_draw = np.random.choice(P_vals, p=posterior_P)
    phi_draw = np.random.choice(phi_vals, p=posterior_phi)
    plt.plot(t_dense, K_draw * np.sin(2 * np.pi * (t_dense / P_draw) +
       phi_draw),
             color='lightblue', alpha=0.2)

plt.plot(t_dense, K * np.sin(2 * np.pi * (t_dense / P) + phi),
         'r--', lw=2, label='True model')
plt.plot(t_dense, mean_K * np.sin(2 * np.pi * (t_dense / mean_P) +
   mean_phi),
         'b-', lw=2, label='Posterior mean prediction')

plt.errorbar(t, v_ob, yerr=sig_ob, fmt='o', color='black',
             ecolor='gray', elinewidth=1, capsize=2,
             label='Observations', markersize=4)
plt.xlabel("Time [days]")
plt.ylabel("Radial velocity [m/s]")
plt.title("Posterior Mean and Uncertainty vs True Model")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.4)
plt.tight_layout()
plt.show()
# %%
```

B. RADIAL VELOCITY MCMC APPROACH

```
#%%
# --- Radial Velocity Example ---
# This script demonstrates simple MCMC parameter estimation for a
# sinusoidal radial velocity model of the form:
# v(t) = K * sin(2 * pi * t / P + phi)

import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)  # The meaning of life for reproducibility
#%%
# --- True Parameter Values ---
K = 10.0            # Velocity semi-amplitude [m/s]
P = 21.4            # Period of the signal [days]
phi = 0.25 * np.pi  # Phase offset [radians]
sig_ob = 3.0        # Observational uncertainty (standard deviations)

#%%
# --- Generate Synthetic Data ---
# Simulate 34 observations across 42 days
t = np.sort(42 * np.random.rand(34))
```

```

# Observed velocities = true model + Gaussian noise
v_ob = K * np.sin((2 * np.pi * t / P) + phi) + np.random.normal(0, sig_ob,
    size=len(t))

#%%
# --- MCMC Parameter Estimation ---
# Define prior, likelihood, and posterior functions

# Uniform prior over plausible parameter ranges
def log_prior(K, P, phi):
    if 5 < K < 15 and 20 < P < 30 and 0 < phi < 2 * np.pi:
        return -np.log(10 * 10 * 2 * np.pi)
    else:
        return -np.inf

# Gaussian likelihood assuming constant sigma
def log_likelihood(K, P, phi):
    v_model = K * np.sin((2 * np.pi * t / P) + phi)
    return -0.5 * np.sum(
      ((v_ob - v_model)/sig_ob)**2 + np.log(2*np.pi*sig_ob**2)
    )

#%%
# Define proposal distribution
def newProposal(current):
  step_size = [0.25, 0.25, 0.1] # Standard deviation for new proposals
  return [
    np.random.normal(current[0], step_size[0]),
    np.random.normal(current[1], step_size[1]),
    np.random.normal(current[2], step_size[2])
  ]

#%%
# Metropolis Algorithm
walks = 10
burn_steps = 1000
steps = 12000
chain = np.zeros((walks, burn_steps + steps + 1,3))

for w in range(walks):
  curr_state = [14,25,np.pi] # Initial guess, should be a good guess
     within the parameter space

  curr_prior = log_prior(*curr_state)
  curr_likelihood = log_likelihood(*curr_state)
  chain[w,0] = curr_state

  accept_count = 0

  for i in range(burn_steps + steps):
    new_proposal = newProposal(curr_state)

    new_prior = log_prior(*new_proposal)
    new_likelihood = -np.inf
    # Only calculate likelihood if the new proposal is in the parameter space
    if (new_prior != -np.inf):
      new_likelihood = log_likelihood(*new_proposal)

    # Accept or reject the proposal?
    new_log_post = new_prior + new_likelihood
    curr_log_post = curr_prior + curr_likelihood
    alpha = np.exp(new_log_post - curr_log_post)
    if alpha > 1:
      alpha = 1

    if np.random.rand() < alpha:
      curr_state = new_proposal
      curr_prior = new_prior
      curr_likelihood = new_likelihood
      accept_count += 1

    chain[w,i+1] = curr_state
  print(accept_count/(burn_steps + steps))

# Remove burn steps
chain = chain[:,burn_steps+1:]

#%%
# Plot the results of the algorithm
K_chain = chain[:,:,0]
P_chain = chain[:,:,1]
phi_chain = chain[:,:,2]
chain_range = np.arange(burn_steps,burn_steps+steps)

fig, axes = plt.subplots(3, 1, figsize=(10, 8), sharex=True)

# K trace
for p in range(len(chain)):
  axes[0].plot(chain_range, K_chain[p])
axes[0].axhline(y=K, color='r', ls='--', label='True K')
axes[0].set_ylabel('K [m/s]')
axes[0].set_title('Trace of K')
axes[0].legend()

```

```python
# P trace
for p in range(len(chain)):
  axes[1].plot(chain_range, P_chain[p])
axes[1].axhline(y=P, color='r', ls='--', label='True T')
axes[1].set_ylabel('Period [days]')
axes[1].set_title('Trace of T')
axes[1].legend()

# phi trace
for p in range(len(chain)):
  axes[2].plot(chain_range, phi_chain[p])
axes[2].axhline(y=phi, color='r', ls='--', label='True phi')
axes[2].set_ylabel('Phase [rad]')
axes[2].set_xlabel('Step')
axes[2].set_title('Trace of phi')
axes[2].legend()

plt.tight_layout()
plt.show()
# %%
# Plot histograms
plt.figure(figsize=(4, 4))
plt.hist(K_chain.flatten(), bins=50, color='tab:blue', alpha=0.7)
plt.axvline(K, color='r', ls='--', label='True K')
plt.ylabel('Count')
plt.xlabel('K [m/s]')
plt.title('Posterior of K')
plt.legend()
plt.show()

plt.figure(figsize=(4, 4))
plt.hist(P_chain.flatten(), bins=50, color='tab:blue', alpha=0.7)
plt.axvline(P, color='r', ls='--', label='True T')
plt.ylabel('Count')
plt.xlabel('Period [days]')
plt.title('Posterior of T')
plt.legend()
plt.show()

plt.figure(figsize=(4, 4))
plt.hist(phi_chain.flatten(), bins=50, color='tab:blue', alpha=0.7)
plt.axvline(phi, color='r', ls='--', label='True phi')
plt.ylabel('Count')
plt.xlabel('Phase [rad]')
plt.title('Posterior of phi')
plt.legend()
plt.show()
# %%
# Calculate expectation values and standard deviations
K_mean = np.mean(K_chain)
K_std = np.std(K_chain)

P_mean = np.mean(P_chain)
P_std = np.std(P_chain)

phi_mean = np.mean(phi_chain)
phi_std = np.std(phi_chain)

print(f"K = {K_mean:.2f} +/- {K_std:.2f}")
print(f"P = {P_mean:.2f} +/- {P_std:.2f}")
print(f"phi = {phi_mean:.2f} +/- {phi_std:.2f}")
# %%
```

REFERENCES

Abbasi, R., Ackermann, M., Adams, J., et al. 2023, The Astrophysical Journal, 959, 96, doi: 10.3847/1538-4357/aceefc

Borne, K. D. 2009, Astroinformatics: A 21st Century Approach to Astronomy, arXiv, doi: 10.48550/arXiv.0909.3892

Brewer, B. J. 2018, in Bayesian Astrophysics, ed. A. A. Ramos & I. Arregui (Cambridge: Cambridge university press)

Cannon, A. J., & Pickering, E. C. 1918, The Henry Draper Catalogue (The Observatory)

Christensen, N., & Meyer, R. 2022, Reviews of Modern Physics, 94, 025001, doi: 10.1103/RevModPhys.94.025001

Collaboration, S., Pallathadka, G. A., Aghakhanloo, M., et al. 2025, The Nineteenth Data Release of the Sloan Digital Sky Survey, arXiv, doi: 10.48550/arXiv.2507.07093

Comrie, L. J. 1932, Monthly Notices of the Royal Astronomical Society, 92, 694, doi: 10.1093/mnras/92.7.694

Cox, R. T. 1946, American Journal of Physics, 14, 1, doi: 10.1119/1.1990764

Duncombe, R. L. 1988, Celestial Mechanics, 45, 1, doi: 10.1007/BF01228969

Feigelson, E. D., & Babu, G. J. 2004, Statistical Challenges in Modern Astronomy, arXiv, doi: 10.48550/arXiv.astro-ph/0401404

Gaia Collaboration, Vallenari, A., Brown, A. G. A., et al. 2023, Astronomy & Astrophysics, 674, A1, doi: 10.1051/0004-6361/202243940

Gelman, A., Gilks, W. R., & Roberts, G. O. 1997, The Annals of Applied Probability, 7, 110, doi: 10.1214/aoap/1034625254

Gregory, P. C. 2005, The Astrophysical Journal, 631, 1198, doi: 10.1086/432594

Hastings, W. K. 1970, Biometrika, 57, 97, doi: 10.1093/biomet/57.1.97

Holmberg, E. 1940, The Astrophysical Journal, 92, 200, doi: 10.1086/144212

Hubble, E. P. 1930, The Astrophysical Journal, 71, 231, doi: 10.1086/143250

Huijse, P., Estevez, P. A., Protopapas, P., Principe, J. C., & Zegers, P. 2014, IEEE Computational Intelligence Magazine, 9, 27, doi: 10.1109/MCI.2014.2326100

Kaushik, M., Mattoo, A., Rastogi, R., & Khare, M. D. 2025, in Big Data Analytics in Astronomy, Science, and Engineering, ed. S. Sachdeva, Y. Watanobe, & S. Bhalla (Cham: Springer Nature Switzerland), 138–153, doi: 10.1007/978-3-031-86193-2_9

Leavesley, S., & Tárnok, A. 2018, Cytometry Part A, 93, 977, doi: 10.1002/cyto.a.23637

Lovis, C., & Fischer, D. 2010, in Exoplanets, 2–4

Lynden-Bell, D. 1967, Monthly Notices of the Royal Astronomical Society, 136, 101, doi: 10.1093/mnras/136.1.101

Merriman, M. 1877, The Analyst, 4, 33, doi: 10.2307/2635472

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. 1953, The Journal of Chemical Physics, 21, 1087, doi: 10.1063/1.1699114

Neal, R. 1993, Probabilistic Inference Using Markov Chain Monte Carlo Methods (Department of Computer Science, University of Toronto)

Olley, A. 2018, Revue de Synthèse, 139, 267, doi: 10.1163/19552343-13900014

Peebles, P. J. E. 1973, The Astrophysical Journal, 185, 413, doi: 10.1086/152431

Schafer, C. M. 2015, Annual Review of Statistics and Its Application, 2, 141, doi: 10.1146/annurev-statistics-022513-115538

Stigler, S. M. 1975, Biometrika, 62, 503, doi: 10.1093/biomet/62.2.503

—. 1981, The Annals of Statistics, 9, 465

Tegmark, M., Taylor, A., & Heavens, A. 1997, The Astrophysical Journal, 480, 22, doi: 10.1086/303939

Toomre, A., & Toomre, J. 1972, The Astrophysical Journal, 178, 623, doi: 10.1086/151823

Trotta, R. 2008, Contemporary Physics, 49, 71, doi: 10.1080/00107510802066753

van Ravenzwaaij, D., Cassey, P., & Brown, S. D. 2018, Psychonomic Bulletin & Review, 25, 143, doi: 10.3758/s13423-016-1015-8

Von Toussaint, U. 2011, Reviews of Modern Physics, 83, 943, doi: 10.1103/RevModPhys.83.943

Zwicky, F. 1937, The Astrophysical Journal, 86, 217, doi: 10.1086/143864