Draft version November 13, 2017
Typeset using LaTeX default style in AASTeX61

GAUSSIAN PROCESS MODELING: A BAYESIAN APPROACH TO DETECTING QUASI-PERIODIC OSCILLATIONS IN SOLAR FLARE DATA

Christopher A. Ick,[1] David W. Hogg,[2] and Daniela Huppenkothen[3]

[1] *New York University*
726 Broadway, Ofc 912
New York, NY 10003, USA
[2] *New York University*
726 Broadway, Ofc 1005
New York, NY 10003, USA
[3] *University of Washington*
3910 15th Ave NE
Office B356B
Seattle, WA 98195, USA

ABSTRACT

Solar flare analysis for quasi-periodic oscillations (QPOs) was formally done using Fourier power spectra analysis, limiting computational efficiency to $\mathcal{O}(n^2)$. We demonstrate a more effective and efficient method of modeling solar flare data using Gaussian Processes, a continuous-domain model, which scales at $\mathcal{O}(n)$ in one-dimensional models, and provide demonstrations of these methods by applying it to both simulated and real astronomical datasets. Using the package Celerite for Gaussian processes, we provide demonstrations of effectively modeling QPOs using a self-correllation function (a kernel function) to model and simulate solar flare phenomena, and then to remodel and capture the original characteristics of the simulated flare data, using standard optimization methods as well as Markov-chain Monte Carlo sampling. After demonstrating the effectiveness of these fitting methods, we go on to apply these modeling/fitting methods to capture the characteristics of real solar flare data, and discuss the potential use of the methods for automated solar flare and QPO detection.

Keywords: keywords go here

Corresponding author: Is that me?
chris.ick@nyu.edu, david.hogg@nyu.edu, daniela.huppenkothen@nyu.edu

1. INTRODUCTION

Introduction goes here.

2. DATA

Data for this project was acquired from the Gamma-ray Burst Monitor (GBM) onboard the Fermi Gamma-ray Space Telescope, as well as the GOES-15 sattelite's x-ray imager. Solar flare data from GOES was provided by Andrew Inglis. The data is formatted in a .fits file, with thw first two columns corresponding to the flux of the 1-8Åand 0.5-4Åchannels respectively.

3. METHODS

Once our data is cleanly organized into a time series containing time coordinates of our measurements, x, intensity measurements, y, and error δ_y, in separate arrays, we can begin our modeling algorithm. We model the flare envelope using the flare model:

$$\mu_\theta(t) = A\lambda exp\left(\frac{-\tau_1}{t-t_s} - \frac{t-t_s}{\tau_2}\right) \tag{1}$$

where t is the time since trigger, t_s is the trigger time, A is the flare evelope amplitude, τ_1 and τ_2 are characteristics of the pulse rise and decay respectively, and $\lambda = exp(2\frac{\tau_1}{\tau_2})^{1/2}$ (CITATION) The free parameters for our fit are A, τ_1, and τ_2, which we will refer to as θ, a parameter vector. We generate our initial guesses for this parameter vector using a simple function that takes the maximum of the flare as A, τ_1 to be $\frac{1}{3}$ of the length of our flare, and τ_2 to similarly be $\frac{2}{3}$ the length of our flare. These initial guesses need only to be approximate, as we'll be optimizing these values after defining all free parameters of our model.

To model a QPO, we use a covariance function, or kernel function $k(\tau_{ij})$, where τ_{ij} is the matrix $\tau_{ij} = t_i - t_j$. To model a QPO using a kernel, we begin utilizing Celerite, a library for 1D Gaussian process regression. We can initialize our envelope model using the aptly-named abstract "Model" class. Similarly, using Celerite's built-in "terms" class, we define a kernel representing stochastically-driven, dampened harmonic oscillator, defined by the differential equation:[1]

$$\left(\frac{d^2}{dt^2} + \frac{\omega_0}{Q}\frac{d}{dt} + \omega_0^2\right) y(t) = \epsilon(t) \tag{2}$$

Where ω_0 corresponds to the natural frequency of the undamped oscillator, Q is the quality factor, and $\epsilon(t)$ is a stochastic driving force. There is an additional scaling term S_0 in the PSD of this process that corresponds to the power at natural frequency, $\omega = \omega_0$. Working under the assumption that all quasi-periodic oscillations are triggered by the same mechanism, we set our initial values of these parameters as those we captured from a real flare.

Finally, we similarly model a red noise process using another model in the "terms" class in Celerite defined by:

$$k(\tau) = a * exp(-c\tau) \tag{3}$$

Where a and c are similar terms for correlational strength and timescale, that are similarly free, and guessed based on known values from a real solar flare. One convenience of our modeling protocol is that we can combine multiple kernel functions into a composite kernel, k_α, which will autmatically concatenate the individual parameter vectors for each kernel into one large parameter vector, which we'll denote as α.

Since we are using a Bayesian probability measurement as our fitting function, we can encode our prior knowledge of our parameters into our models, as prior distributions. For the sake of convenience and simplicity, we applied tophat priors for each of these parameters within reasonable known values of these parameters. [2]

From here, we have all neccessary information to initialize our models: the envelope model and the kernel functions are initialized with their parameters, the initial guesses for each parameter, and the prior distribution for each parameter. From here, we can initialize a GP-class (Gaussian Process) object, utilizing the model as the mean function for the process, and the kernel function as a constructor for the kernel matrix. Once initialized, we have access to several of the GP computations encoded in the Celerite solver, such as computing the kernel matrix and factorizing.

[1] I can include the whole solution set, which depends on the size of the quality factor Q, is that useful?

[2] We may want to think about our priors a bit more before I write out a table for them...

With these computations ready, we feed the time coordinates (x) as well as intensity error (δ_y) into the GP.compute() function to compute the kernel matrix. With the kernel matrix computed, we can begin the optimization process. For a fit, we want to maximize the likelihood of our fit over parameterspace defined by θ and α. For a Bayesian model, we can write out the computation of the likelihood as a result of Bayes theorem [3]. It's more useful to write out the log-form:

$$\ln \mathcal{L}(\theta, \alpha) = \ln p(y|x, \theta, \alpha) = -\frac{1}{2} r_\theta^T K_\alpha^{-1} r_\theta - \ln det(K_\alpha)^{\frac{1}{2}} - \frac{N}{2} \ln(2\pi) \tag{4}$$

Where the vector $r_\theta \equiv y - \mu_\theta(x)$. With this function defined, we can define a function that computes the negative log-likelihood ($-\mathcal{L}$) as a function of the joint parameter vector (θ, α). With this, we can use any optimization scheme to find the optimal values for the parameter vectors; in our case, we called the minimize function from scipy's optimization submodule, utilizing Limited-memory BFGS optimization.

This is generally sufficient to compute the optimal parameters of the solar flare envelope, θ, but for the QPO parameters, α, we can better understand the parameter-space, and to maximize the probability of our fit by using Markov chain Monte Carlo sampling (MCMC). Using the emcee package, we initialize a set of walkers to explore the parameter-space of the model, distributed about the optimal parameters previously computed. Defining a log-probability method as the sum of our likelihood computation as well as our prior at the parameterspace θ acts as our probability distribution for our walkers.

4. DISCUSSION

Let's talk about what happened.

REFERENCES

Something 2017

[3] Do I need to know how to derive this?