Draft version January 22, 2018
Typeset using LaTeX **RNAAS** style in AASTeX62

# Scalable gradients of Gaussian Process models with celerite

Daniel Foreman-Mackey[1]

[1] *Center for Computational Astrophysics, Flatiron Institute, New York, NY*

*Keywords:* methods: data analysis — methods: statistical

## ABSTRACT

This research note presents a derivation and implementation of reverse-mode gradients of the *celerite* algorithm for scalable inference with Gaussian Process (GP) models. These gradients can be integrated into existing automatic differentiation frameworks to provide a scalable and computationally efficient method for evaluating the gradients of the GP likelihood with respect to the kernel hyperparameters. The algorithm derived in this note uses less memory and is more efficient than versions using automatic differentiation and the computational cost scales linearly with the number of data points.

## INTRODUCTION

Gaussian Processes (GPs Rasmussen & Williams 2006) are a class of models used extensively in the astrophysical literature to model stochastic processes. The applications are broad-ranging and some example include the time domain variability of astronomical sources (Brewer & Stello 2009; Kelly et al. 2014; Haywood et al. 2014; Rajpaul et al. 2015; Foreman-Mackey et al. 2017), data-driven models of light curves or stellar spectra (Wang et al. 2012; Luger et al. 2016; Czekala et al. 2017), and the cosmic microwave background (Bond & Efstathiou 1987; Wandelt & Hansen 2003). In all of these applications, the calculation and optimization of the GP marginalized likelihood function (here we follow the notation of Foreman-Mackey et al. 2017)

$$\log \mathcal{L}(\boldsymbol{\theta},\, \boldsymbol{\alpha}) = -\frac{1}{2}\, [\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{\theta}}]^{\mathrm{T}}\; {K_{\boldsymbol{\alpha}}}^{-1}\; [\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{\theta}}] - \frac{1}{2}\, \log\det K_{\boldsymbol{\alpha}} + \text{constant} \tag{1}$$

is of central importance and this can often be the computational bottleneck. The details of this model are omitted here (more details can be found in Rasmussen & Williams 2006 and Foreman-Mackey et al. 2017), but the key point is that, for a dataset with $N$ data points, every evaluation of a GP model requires computation of the inverse[1] and log-determinant of the $N \times N$ covariance matrix. The computational cost of these operations scales as $\mathcal{O}(N^3)$ in the general case, but the *celerite* method was recently introduced in the astronomical literature to compute the GP likelihood for a class of one-dimensional models with $\mathcal{O}(N)$ scaling (Foreman-Mackey et al. 2017). The details of these models can be found elsewhere (Foreman-Mackey et al. 2017), but the basic idea is that for some covariance functions, the matrix $K$ can be written as a rank-$J$ semi-separable matrix

$$K = \mathrm{diag}(\boldsymbol{a}) + \mathrm{tril}(U\, V^{\mathrm{T}}) + \mathrm{triu}(V\, U^{\mathrm{T}}) \tag{2}$$

where $\boldsymbol{a}$ is a $N$-vector, $U$ and $V$ are $N \times J$ matrices, the diag function creates a diagonal matrix from a vector, and the tril and triu extract the strict lower and upper triangular matrices from a dense matrix. For a matrix of this form, Foreman-Mackey et al. (2017) derived an algorithm with $\mathcal{O}(N\, J^2)$ scaling for computing the Cholesky factorization

$$K = L\, \mathrm{diag}(\boldsymbol{d})\, L^{\mathrm{T}} \tag{3}$$

where

$$L = I + \mathrm{tril}(U\, W^{\mathrm{T}}) \tag{4}$$

[1] Or, more specifically, a linear operator that can left multiply a vector or a matrix by the inverse of this matrix.

for some $N \times J$ matrix $W$. The algorithm from Foreman-Mackey et al. (2017) includes an $N-1 \times J$ "preconditioning" matrix[2] that we include with the notation $P$. The Cholesky factorization algorithm derived by Foreman-Mackey et al. (2017) is given is reproduced in the following algorithm:

**function** `celerite_factor`($U$, $P$, $\boldsymbol{d}$, $W$)
  # If at input $\boldsymbol{d} = \boldsymbol{a}$, $W = V$, and $K$ is given by Equation (2),
  # at output $K = L\,L^{\mathrm{T}}$ where $L$ is given by Equation (4).
  $S \leftarrow$ `zeros`($J$, $J$)
  $\boldsymbol{w}_1 \leftarrow \boldsymbol{w}_1 / d_1$
  **for** $n = 2, \ldots, N$:
    $S \leftarrow \mathtt{diag}(\boldsymbol{p}_{n-1})\,[S + d_{n-1}\,{\boldsymbol{w}_{n-1}}^{\mathrm{T}}\,\boldsymbol{w}_{n-1}]\,\mathtt{diag}(\boldsymbol{p}_{n-1})$
    $d_n \leftarrow d_n - \boldsymbol{u}_n\,S\,{\boldsymbol{u}_n}^{\mathrm{T}}$
    $\boldsymbol{w}_n \leftarrow [\boldsymbol{w}_n - \boldsymbol{u}_n\,S]\,/d_n$
  **return** $\boldsymbol{d}$, $W$, $S$

In this algorithm, the `zeros`($J$, $K$) function creates a $J \times K$ matrix of zeros, the `diag` function creates a diagonal matrix from a vector, and $\boldsymbol{x}_n$ indicates a row vector made from the $n$-th row of the matrix $X$. Using this factorization, the log-determinant of $K$ is

$$\log\det K = \sum_{n=1}^{N} \log d_n \quad . \tag{5}$$

Similarly, Foreman-Mackey et al. (2017) derived a $\mathcal{O}(N\,J)$ algorithm to apply the inverse of $K$ as shown in the following algorithm:

**function** `celerite_solve`($U$, $P$, $\boldsymbol{d}$, $W$, $Z$)
  # If at input $Z = Y$, at output $Z = K^{-1}\,Y$.
  $F \leftarrow$ `zeros`($J$, $N_{\mathrm{rhs}}$)
  $G \leftarrow$ `zeros`($J$, $N_{\mathrm{rhs}}$)
  **for** $n = 2, \ldots, N$:
    $F \leftarrow \mathtt{diag}(\boldsymbol{p}_{n-1})\,[F + {\boldsymbol{w}_{n-1}}^{\mathrm{T}}\,\boldsymbol{z}_{n-1}]$
    $\boldsymbol{z}_n \leftarrow \boldsymbol{z}_n - \boldsymbol{u}_n\,F$
  **for** $n = 1, \ldots, N$:
    $\boldsymbol{z}_n \leftarrow \boldsymbol{z}_n / d_n$
  **for** $n = N-1, \ldots, 1$:
    $G \leftarrow \mathtt{diag}(\boldsymbol{p}_n)\,[G + {\boldsymbol{u}_{n+1}}^{\mathrm{T}}\,\boldsymbol{z}_{n+1}]$
    $\boldsymbol{z}_n \leftarrow \boldsymbol{z}_n - \boldsymbol{w}_n\,G$
  **return** $Z$, $F$, $G$

## GRADIENTS OF GP MODELS USING CELERITE

The standard method of computing gradients of Equation (1) with respect to the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ is the equation (Rasmussen & Williams 2006)

$$\frac{\mathrm{d}\log\mathcal{L}}{\mathrm{d}\alpha_k} = \frac{1}{2}\,\mathrm{Tr}\left[\left[\tilde{\boldsymbol{r}}\,\tilde{\boldsymbol{r}}^{\mathrm{T}} - K_{\boldsymbol{\alpha}}^{-1}\right]\,\frac{\mathrm{d}K}{\mathrm{d}\alpha_k}\right] \tag{6}$$

where

$$\tilde{\boldsymbol{r}} = {K_{\boldsymbol{\alpha}}}^{-1}\,[\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{\theta}}] \quad . \tag{7}$$

Similar equations exist for the parameters $\boldsymbol{\theta}$. Even with a scalable method of applying ${K_{\boldsymbol{\alpha}}}^{-1}$, the computational cost of Equation (6) scales as $\mathcal{O}(N^2)$. This scaling is prohibitive when applying the *celerite* method to large datasets and I

[2] This matrix is called $\phi$ by Foreman-Mackey et al. (2017).

have not found a simple analytic method of improving this scaling for semi-separable matrices. However, it was recently demonstrated that substantial computational gains can be made by directly differentiating Cholesky factorization algorithms even in the general case (Murray 2016).

Following this reasoning and using the notation from an excellent review of matrix gradients (Giles 2008), I present the reverse-mode gradients of the *celerite* method. While not yet popular within astrophysics, "reverse-mode accumulation" of gradients (also known as "backpropagation") has revolutionalized the field of machine learning over the past few decades (TODO ADD CITATION). The review (Giles 2008) provides a thorough overview of these methods and the interested reader is directed to that discussion for details and for an explanation of the notation.

**function** `celerite_factor_grad`($U$, $P$, $\boldsymbol{d}$, $W$, $S$, $\bar{\boldsymbol{d}}$, $\bar{W}$, $\bar{S}$)
    # If at input $\boldsymbol{d} = \boldsymbol{a}$, $W = V$, and $K$ is given by Equation (2),
    # at output $K = L\,L^\mathrm{T}$ where $L$ is given by Equation (4).
    $S \leftarrow$ `zeros`($J$, $J$)
    $\boldsymbol{w}_1 \leftarrow \boldsymbol{w}_1 / d_1$
    **for** $n = 2, \ldots, N$:
        $S \leftarrow$ `diag`$(\boldsymbol{p}_{n-1})\,[S + d_{n-1}\,{\boldsymbol{w}_{n-1}}^\mathrm{T}\,\boldsymbol{w}_{n-1}]$ `diag`$(\boldsymbol{p}_{n-1})$
        $d_n \leftarrow d_n - \boldsymbol{u}_n\,S\,{\boldsymbol{u}_n}^\mathrm{T}$
        $\boldsymbol{w}_n \leftarrow [\boldsymbol{w}_n - \boldsymbol{u}_n\,S]\,/d_n$
    **return** $\boldsymbol{d}$, $W$, $S$

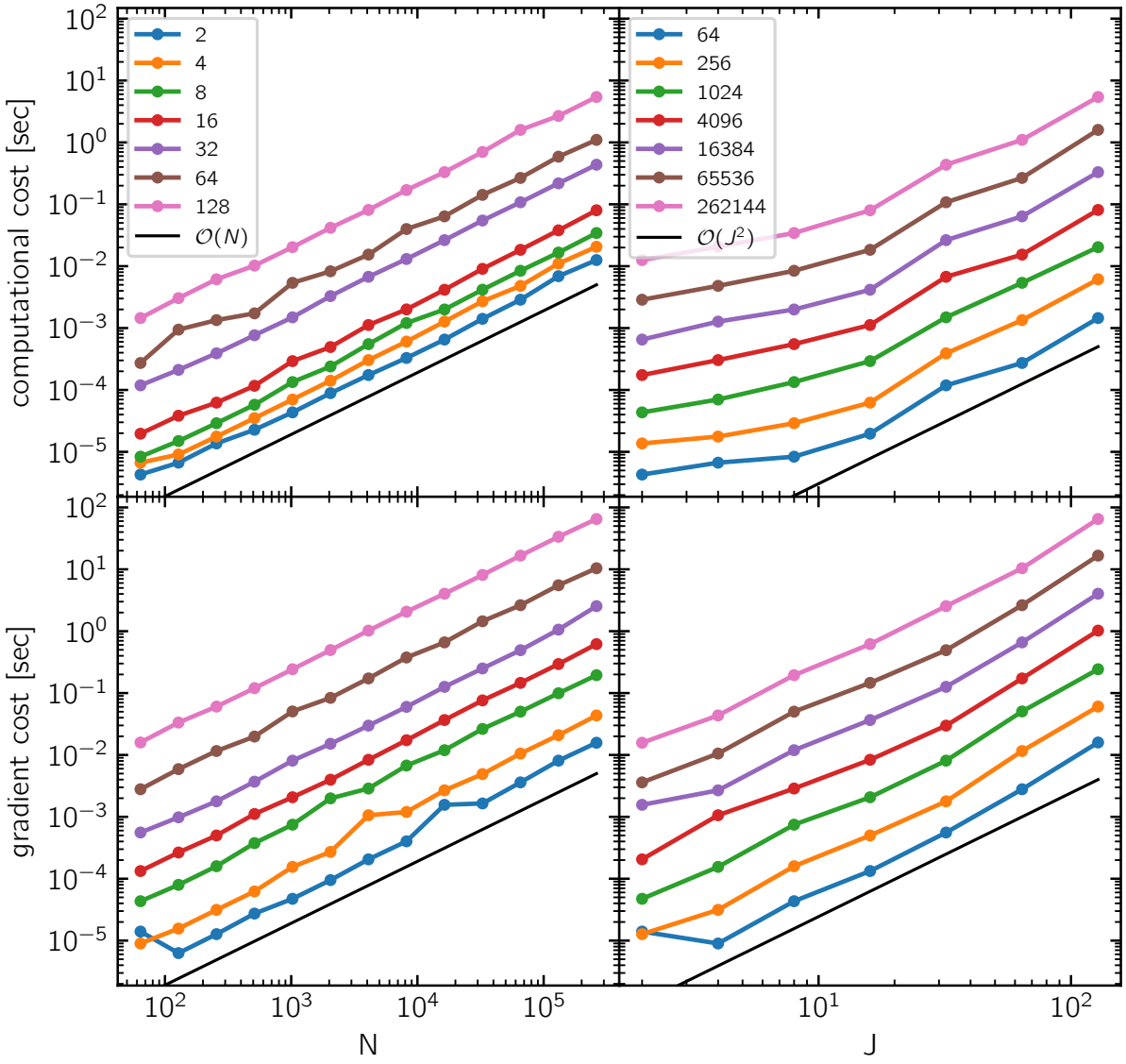


**Figure 1.** Scaling.

## REFERENCES


Bond, J. R., & Efstathiou, G. 1987, MNRAS, 226, 655

Brewer, B. J., & Stello, D. 2009, MNRAS, 395, 2226

Czekala, I., et al. 2017, ApJ, 840, 49

Foreman-Mackey, D., et al. 2017, AJ, 154, 220

Giles, M. 2008, An extended collection of matrix derivative results for forward and reverse mode automatic differentiation, Tech. Rep. NA-08/01, Oxford University Computing Laboratory

Haywood, R. D., et al. 2014, MNRAS, 443, 2517

Kelly, B. C., et al. 2014, ApJ, 788, 33

Luger, R., et al. 2016, AJ, 152, 100

Murray, I. 2016, ArXiv, 1602.07527

Rajpaul, V., et al. 2015, MNRAS, 452, 2269

Rasmussen, C. E., & Williams, K. I. 2006, Gaussian Processes for Machine Learning (MIT Press)

Wandelt, B. D., & Hansen, F. K. 2003, PhRvD, 67, 023001

Wang, Y., et al. 2012, ApJ, 756, 67