

A report on "EMRI_MC: A GPU-based code for Bayesian inference of EMRI waveforms"

by Ippocratis D. Saltas and Roberto Oliveri

////////////////////////////////////

This is a very interesting work that is based on enhancing existing analysis frameworks (waveform modelling), and improving performance with the use of GPU acceleration methods. This paper is quite technical, and reads more as a technical report rather than a typical manuscript. In my view this is totally fine, as I find technical works to be extremely valuable, because they are easier to understand and reproduce

Therefore, I believe that this paper is worth publishing after addressing some of the points raised below.

A. Comments on the manuscript

1. First general comment: I find that the text is in need of more detailed discussion on the comparison of past implementations of EMRI analyses and the one introduced in this work. I believe that a discussion section (or at least a few paragraphs) should be written, in order to stress the novel ideas introduced here.
2. In section 2, the authors write "*Parameter forecasting for EMRI signals is not an easy task, because of the challenge to model their waveforms and the high-dimensional parameter space that needs be explored*". This is true, and another challenge is the multimodality of the likelihood, and possible degeneracies (e.g. see Chua & Cutler 2022). Another is the potential overlap and confusion with other signals (transient or stochastic, e.g. see works about the Global Fit of the LISA data).
3. End of section 2, item (iv): Up to this point it is not entirely clear which elements of the analysis are parallelized with GPU hardware. Before section 3, it would be useful to see a short list of items that the authors have improved with GPU parallelisation (e.g. the likelihood and/or the different parts of waveform). Otherwise the reader must go through the complete text, or even the code, in order to read this information.
4. Section 4: Considering the challenges of the analysis of EMRI signals, I believe that the section 4 is a bit short and lacks details (especially compared to the rest of the manuscript). In particular, it is not clear whether the analysis is performed by using the Time Delay Interferometry (TDI) channels, or it is done directly in h_c (or similar) units. For the former, a more detailed description is required about the number of channels,

and whether a noise orthogonal TDI combination is adopted (this could be assumed from eq. 8, but it should be properly described in the text). If the analysis is done in h_c units it should also be clearly stated and described, because it could introduce simplifying assumptions in the parameter estimation process. Usually this is done by assuming ideal instrument and perfect knowledge of the overall system (for example ignoring transfer function variations for the various noises and signals, as well as other analysis complications such as correlations between channels). A simple plot of some mock data in frequency domain could be useful in this section.

5. Same section: It is not stated whether the analysis is "noiseless", or if the instrumental noise is simulated from the noise curve (or any other method). Maybe it's evident from the code implementation, but I think it should also be clear in the text.
6. Same section on the 'waveform computation' paragraph: Extrinsic parameters are kept fixed at true values for the parameter estimation analysis. This is another simplification in that needs to be stated in the introduction and/or the discussion sections.
7. Same section, 'Functionality overview' paragraph: One can assume that the MCMC walkers are parallelized with multiprocessing (CPU), but it would be better to clearly state it in the text for the readers not familiar with the emcee software.
8. Same paragraph as above: The efficiency and performance of the software is reported. While it's very challenging to directly compare with other software implementations from previous works, I think it would be beneficial to present some ballpark estimates for comparison.
9. Figure 2: Crosses that mark the true value (zero in this particular case), are useful in order to visually check for correlations directly from the figure. Also, the authors state that "*The constraints are somewhat tighter than those in the literature [13], as our MCMC exploration covers a smaller EMRIs' parameter space*". This is indeed probably one of the reasons to get smaller relative errors on the parameters, but other simplifying assumptions on the analysis (as speculated in previous points) could also contribute. Another reason could be the different version of instrumental noise. The one used here is quite outdated. In summary, I believe that more possible explanations should be added here and in the main text as a short discussion on the results.

B. Comments on the software

In this section of the review I comment on the implementation of the software that accompanies this work. I should note here that I am not an expert on software development. For that reason, all the comments below are purely suggestive, and up to the authors to decide whether they want to implement them for a revised version of the manuscript, or leave them for a future update of the software.

1. I believe that there are many benefits when (at least part of) the code is pip-installable. In my eyes, the most important element is the waveform implementation, which could be used as a direct plugin in other likelihoods and analysis pipelines. This could also

expand the potential user-base of the software.

2. About the likelihood computation: Unless I am mistaken, the likelihood function is computed serially for each of the walkers. However, the emcee (or similar MCMC implementations) support vectorized likelihood outputs. This means that it is possible to build a likelihood function that has as input a matrix of parameter values [$n_{\text{walkers}} \times n_{\text{parameters}}$], computes an array of residuals [$n_{\text{walkers}} \times n_{\text{datapoints}}$] and then outputs a vector of likelihood [n_{walkers} ,] values, each corresponding to each walker. This allows for even higher efficiency with the GPU hardware, which is ideal for such vectorized operations.
3. From the code it is clear that the analysis is performed on 'noiseless' data, i.e. no noise is simulated. Then, the likelihood can be simplified to $(d|h) - 0.5 (h|h)$. E.g. see eq. (31) from Cañizares et al 2013.
4. Unless I am mistaken, the noise is computed at each iteration, which is probably redundant, unless the noise is to be inferred from the data. A solution would be to precompute it and use it like the rest of the global constants. Another idea would be to transform the likelihood function into a likelihood class, which would compute the noise vector once and store it for use at each evaluation.
5. A very useful idea (but would require some extra work), is to make the code CPU/GPU agnostic. At the beginning of the script one can check if a GPU device is detected. If not, then the usual numpy library can be imported as `import numpy as cp`, and continue with the computations using the available CPUs. This adjustment would make the software more robust, and probably quite useful to the potential users with no access to GPUs.