Dear Didier Paillard (cc Quizhen Yin),

I have read your manuscript with great interest, as I am myself have written a Python package for calculating past irradiance (https://github.com/bryanlougheed/orbitalchime/), although I haven't gotten around to finishing a manuscript yet due to my current employment situation. Therefore, I had a look through your package and found that it works as described and will be helpful for researchers using Python.

In addition to the helpful comments from reviewers Marie-France Loutre and Michel Crucifix, I have some further minor comments:

On referring to ϖ as "climatic precession"

In your manuscript, longitude of perihelion (ϖ) is referred to as "climatic precession" (line 23, 92 and 472). However, ϖ is not in itself precession (climatic or otherwise), but rather the orbital angle corresponding to longitude of perihelion, which is governed by changes in general precession (the combination of axial and apsidal precession), as you note in the second half of line 23. I would suggest to refer to ϖ as "longitude of perihelion", as is usually done om the literature.

The term "climatic precession" has typically been used in the palaeoclimate literature for e*sin(ϖ), an index that visualises both eccentricity and longitude of perihelion in tandem (Imbrie and Imbrie, 1982; Vernekar, 1972; most likely also earlier works), which you have described in line 476 as the "climatic precession paramater", but can be simply described as "climatic precession".

Elliptical vs numerical integration

Regarding your text that I have quoted below:

> *The simplest way to produce the corresponding time series for Quaternary climates is to compute the daily insolation for every day of the year, then select and sum the ones above the chosen threshold. This might not be the most efficient way. Besides, discretizing the year into an integer number of days (360 or 365 days) is not entirely rigorous. Discretization itself might lead to slight numerical errors.*

Here, you are discussing the disadvantages of using an a numerical integration instead of an elliptical integration. A numerical integration would indeed involve calculating irradiance across discrete intervals, but typically this does not involve breaking down the year into 360 or 365 discrete days. Mean daily irradiance can be calculated at any desired discrete interval resolution of the year (e.g. 1/10000th of a year resolution, or better), after which numerical integration can be carried out on the intervals using, e.g., a midpoint, trapezoidal, etc., approach. Berger et al. (2010) compare the results of elliptical and numerical integration for various irradiation quantities in their Tables 1a-c, 2a-c, where differences between the two methods are found to be near-negligible.

Therefore, I would argue that the "numerical errors" or lack of "rigour" are not practical reasons for using elliptical integration instead of numerical integration. In any case, there is already a very small loss of precision in palaeo-irradiance calculations due a number of approximations, including, for example, the

assumption of the orbital paramaters remaining constant throughout the tropical year (in particular in the case of ϖ). Therefore, precision beomes a somewhat irrelevant reason for choosing either elliptical or numerical integration.

You mention efficiency, and I agree that this could be a decisive motivation for using elliptical integration, depending on computational resources and the size of the dataset (e.g. number of kyr and/or latitudes) needing to be computed.

Approach used to solve the Kepler equation for *E*

When solving the Kepler equation for *E*, there is no closed-form algebraic solution for *E,* so other methods must be used, typically involving an iterative approach that converges on the correct value for *E* to within a desired precision. I saw that in your py files you make use of the Halley's method option within *SciPy*'s *optimize.newton* function. Perhaps you could consider citing the necessary literature as well as the *SciPy* package itself, seeing as solving the Kepler equation is of such importance for many irradiance calcuations.

Python documentation strings

You have provided some documentation for your python functions as commented out text (using #) in your py files, above the *def* line of the functions. This documentation can only be accessed by the user by opening the py file in a text editor. You could instead include the documentation as a python docstring block (bookended using ″″″) just below the *def* line, which would also make your documentation callable to the user's python environment using the *help(functionname)* or *?functionname* commands.

I look forward to seeing the work published in *Climate of the Past* and hope that my comments have been of some help.

Kind regards,
Bien cordialement,

Bryan Lougheed