EEPS2910W Final Paper
Dan Wexler
December 2022

Visualizing Milankovitch Cycles and Insolation with Python

I Introduction

The Milankovitch Cycles are changes in the Earth's orbit and position relative to the sun over time that affect the amount of incoming radiation at each point on the surface of the planet throughout the year. There are three different cycles (eccentricity, obliquity, and precession), and they play a primary role in forcing the glacial-interglacial cycle. Eccentricity describes the ellipticity of the Earth's orbital path around the Sun, obliquity describes the tilt of Earth's rotational axis relative to the line perpendicular to the orbital plane, and precession describes the circular wobble of the rotational axis over time. Understanding these cycles is key to understanding the Earth's climate system, but they are difficult to visualize. This paper presents an interactive animation developed in Python that will help the user visualize the individual cycles, as well as how these cycles affect solar insolation at the Earth's atmosphere over time.

II Goals

The primary goal of this animation is to help the user visualize the Milankovitch Cycles. When trying to understand the cycles for my own coursework, I remember being frustrated at the lack of online educational material dedicated to visualizing the cycles. It is hard to visualize the three-dimensional movement of a sphere in space, and access to a quality animation would have accelerated my understanding. Most of the visualization resources that exist online today are pictures, many including a 'before' and 'after' image of the Earth or the Earth's orbit at two different points in time. This is somewhat helpful, but it does not answer the question of how one state changes over time into another. Of the animations that do exist, they are most are only a few seconds long, and I was unable to find an animation that showed all three cycles operating at once. Also, aside from one website [1] and one paper [2], there are no visualizations of insolation at different latitudes throughout the year.

The animation presented in this paper attempts to remedy this lack of resources by showing eccentricity, obliquity, precession, and insolation all operating simultaneously, and uses numerical solutions [3] to these orbital cycles over the past 20 Ma in the visualization. It is coded in Python as to not be prohibitive in terms of software, and there are detailed instructions on how to use and interact with the animation in a later section of the paper.

III Animation

III.1 Description

The animation is split into quadrants (Figure 1). The top left panel shows eccentricity (and the time that has passed; the animation currently moves backwards in time from the present

to 20 Ma), the top right panel shows obliquity, the bottom left panel shows precession (as well as its modulation by obliquity), and the bottom right panel shows insolation, which is calculated using the current values of the three orbital parameters. The time that passes each frame is specified by the user at runtime and must be an integer between 100 and 5,000 years. Each panel containing an orbital cycle also has an animated graph beneath the animation that shows how the specific parameter will change in the future. For example, in Figure 1, the obliquity is currently increasing. These graphs help the user understand the different timescales that the cycles operate on (eccentricity cycles every ~100,000 years, obliquity cycles every ~41,000 years, and precession cycles every ~26,000 years). The insolation panel shows insolation at latitudes from 90 south to 90 north throughout the year. Also, this animation uses numerical solutions to the orbital cycles [3], meaning that the changes shown in the Earth's orbit, its axis, and insolation are all generated from real data. The changes are greatly exaggerated, however, as the purpose of this animation is to be an educational visualization tool, rather than perfectly accurate.

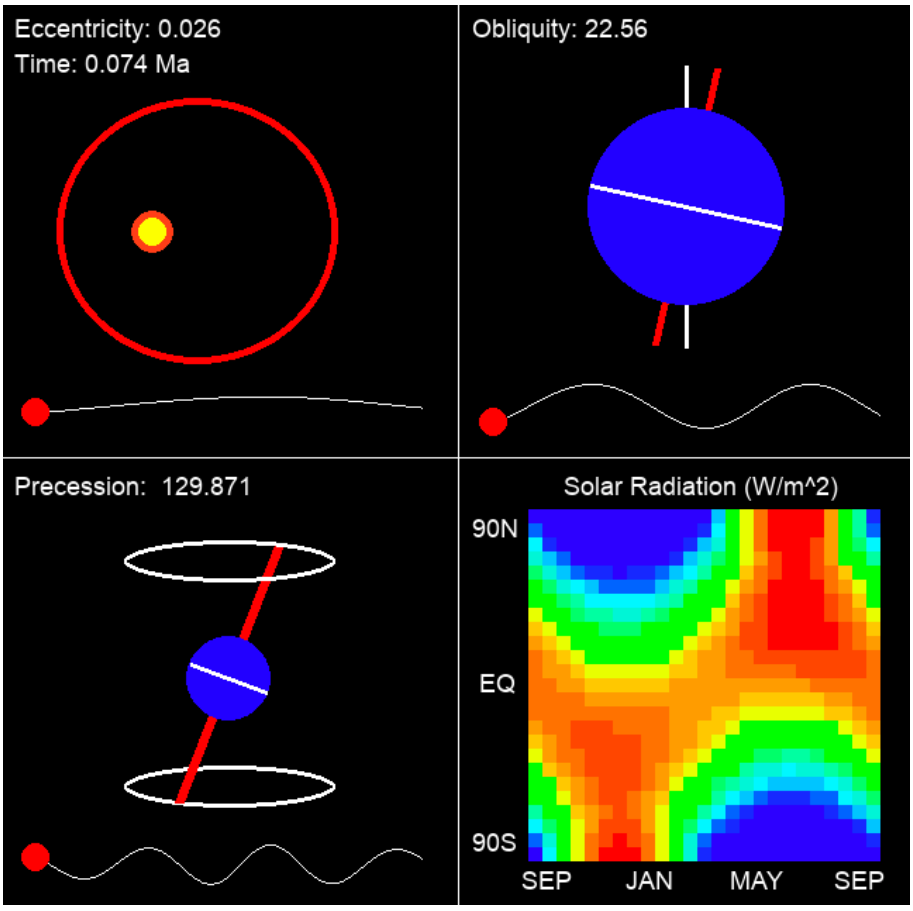

Figure 1. A screen capture of the animation at a certain point in time. The top left panel displays eccentricity and the time that has elapsed. The top right panel displays obliquity. The bottom left panel displays precession. The bottom right panel displays insolation at different latitudes throughout the year. The three panels displaying orbital cycles also have animated graphs.

Finally, the user can pause the animation using the spacebar, and move it backwards or forwards in time by the timestep specified at runtime using the left and right arrow keys.

III.2 Eccentricity

Eccentricity is parameter between zero and one that measures the elongation of an ellipse and is specified as the distance from the center of ellipse to a focal point divided by the length of the semi-major axis (Equation 1) and is currently sitting at a value of 0.01671. In the case of a circle, the two focal points are located at the center, so the eccentricity becomes zero.

$$e = \frac{c}{a}$$

Equation 1. An equation for the eccentricity parameter of an ellipse. In this equation, *e* is the eccentricity, *c* is the distance from the center of the ellipse to the focal points, and *a* is the length of the semi-major axis.

The eccentricity of the Earth's orbit around the sun varies between ~0 and ~0.06. This animation assumes that the length semi-major axis of the orbit never changes (which is mostly true), so in the equation above, *a* is constant. Knowing the eccentricity and the length of the semi-major axis, one can calculate the distance from the center of the ellipse to a focal point. The sun sits at one of the two focal points of the orbit, so we can calculate the distance between the center of the ellipse and the sun. Assuming the sun remains at a fixed location, we can calculate where the center of the ellipse should be. Using the following equation (Equation 2) and knowing *e* and *a*, we can calculate the length of the semi-minor axis, *b*.

$$e = \sqrt{1 - \frac{b^2}{a^2}}$$

Equation 2. Another equation for the eccentricity parameter of an ellipse. In this equation, *e* is the eccentricity, *b* is the length of the semi-minor axis, and *a* is the length of the semi-major axis.

Now we know the location of the center of the ellipse, the length of the semi-major axis, and the length of the semi-minor axis. With all this information, we can draw the ellipse. Of course, in the animation this drawing is greatly exaggerated for visualization purposes.

III.3 Obliquity

Obliquity is the measure of the tilt of the Earth's rotational axis relative to the line perpendicular to the orbital plane. For example, if the Earth had an obliquity of zero, the axis of rotation would be perfectly perpendicular to the orbital plane and there would be no seasons. The obliquity of the axis of rotation varies between ~22.1 and ~24.5 degrees and is currently sitting at

a value of 23.4 degrees. In the animation, the data is normalized to values between 5 and 40 degrees to exaggerate the variability, and the real value is shown above the animation.

III.4 Precession

Precession measures the wobble of the Earth's rotational axis. Over the course of ~26,000 years, the axis traces out a circle. Precession is modulated by eccentricity, and thus the equation for the precession index (Equation 3) includes eccentricity. The other value in the equation, ω, represents the longitude of the perihelion, and refers to the changing point along Earth's orbit around the Sun that the northern hemisphere experiences midsummer [4]. This longitude is the value we want for the animation and varies between 0 and 360 degrees.

$$p = e * \sin(\omega)$$

Equation 3. The equation for the precession index. In this equation, p is the precession index, e is the eccentricity, and ω is the longitude of the perihelion.

Knowing the precession index and the eccentricity at a given time, we can solve for the longitude of the perihelion. This longitude refers where along the Earth's orbit around the Sun the northern hemisphere experiences midsummer, and equivalently refers to where in the 'traced circle' the rotational axis is currently pointing. The size of this traced circle is determined by obliquity (a higher obliquity means a greater tilt and a larger traced circle), and this modulation is reflected in the animation (once again, exaggerated).

III.5 Insolation

The insolation calculation that follows was adopted from an online visualization of the Milankovitch Cycles [1]. The following equations describe how to calculate insolation for a given latitude and longitude (in this case, longitude refers to the time of year, or the angular distance of the Earth along its orbital path around the Sun). The following equation (Equation 4) describes how to calculate the hour angle of the Sun at sunrise, which is the angular displacement of the sun east or west of the local meridian. The next equation (Equation 5) describes how to set the parameter *h0*. The next equation calculates the ratio of the mean distance between the Earth and Sun to the actual distance and uses the eccentricity and longitude of the perihelion (Equation 6). Finally, the last equation details how to calculate insolation using the latitude, longitude, and all the previously calculated quantities (Equation 7).

$$arg = \tan(lat) * \tan(obliquity * \sin(lon))$$

Equation 4. The equation for the hour angle of the Sun at sunrise for a given latitude and longitude. The hour angle is the angular displacement of the sun east or west of the local meridian, which is a circle of constant latitude passing through a place on Earth's surface [5].

$$h0 = pi \; if \; \arg > 1$$
$$h0 = 0 \; if \; \arg < -1$$
$$h0 = (-\arg) \; otherwise$$

Equation 5. The variable *h0* is then set depending on the value of *arg*.

$$ratio = 1 + eccentricity * \cos\left(lon - perihelion\right)$$

Equation 6. Used to calculate the ratio of the mean distance between the Earth and Sun to the actual distance at the current eccentricity, longitude, and longitude of perihelion.

$$dec = obliquity * \sin(lon)$$
$$insolation = \frac{1367}{pi} * ratio^2 * (\mathrm{h0} * \sin(lat) * \sin(\mathrm{dec}) + \cos(\mathrm{lat}) * \cos(\mathrm{dec}) * \sin(h0))$$

Equation 7. The pair of equations used to calculate the insolation for a given latitude and longitude. The first equation calculates declination using the current obliquity and time of year, and the second equation calculates insolation at a certain latitude and time of year.

Using this sequence of equations, we can calculate the amount of sunlight received at the upper atmosphere at every latitude on Earth throughout the year given the current values of eccentricity, obliquity, and precession. This insolation animation is shown in the bottom left.

IV Future

If given more time, several features and updates could be added to improve the animation. First, I would like to add the functionality for the user to display only one of the four quadrants at once, taking up the whole screen. For example, at runtime the user could specify that they would like to see only the obliquity animation, and then the obliquity animation would occupy the entirety of the screen and the other animations would be absent. The same option would be available for each individual animation. Also, I would like to add the functionality for the user to be able to toggle between 'exaggerated' and 'realistic' at runtime, meaning that the animations would be exaggerated (as they are now), or truer to life. The animations are currently exaggerated for visualization purposes, but the option to show the actual variation of the orbital parameters could be useful as well. I would also like to add some sort of sliding bar that the user could interact with while the animation was running to move between different points in time. Currently the user can pause the animation with the spacebar and move backwards and forwards in time with the left and right arrow keys, but this does not allow the user to jump quickly between distant points in time rapidly. Lastly, I think the precession animation could benefit from a few tweaks. I do not think it is currently obvious that the Earth's rotational axis is tracing out a circle – adding some animated curved lines to the Earth would help make this clearer.

V Instructions

The code for the animation can be downloaded from the GitHub link pasted at the bottom of this paper. Once the code is downloaded, the Python modules NumPy and Pygame must be installed. After this download and setup is complete, the animation can be run in the terminal with the command below (Command 1), where -timestep is replaced by an integer in between 100 and 5000. This timestep value represents the time that passes with each frame (the animation runs at 60 frames per second). Once the animation is running, it can be paused with the spacebar key, and the user can move backwards or forwards in time (only when paused) using the left and right arrow keys, respectively. The animation runs for 20 Ma.

python milankovitch.py -timestep

Command 1. The command for running the animation locally. The *python* part of the command indicates to the compiler that the user is attempting to run a Python file, the *milankovitch.py* part of the command specifies the file, and the *-timestep* specifies the amount of time that the user wants to pass with each frame. Depending on your local installation of Python, or whether you are using a virtual environment, the *python* part of the command may vary slightly.

VI References

[1] https://biocycle.atmos.colostate.edu/shiny/Milankovitch/

[2] Kostadinov, T. S., and R. Gilb. “Earth Orbit V2.1: A 3-D Visualization and Analysis Model of Earth's Orbit, Milankovitch Cycles and Insolation.” *Geoscientific Model Development*, vol. 7, no. 3, 2014, pp. 1051–1068., https://doi.org/10.5194/gmd-7-1051-2014.

[3] Laskar, J., et al. “LA2010: A New Orbital Solution for the Long-Term Motion of the Earth.” *Astronomy & Astrophysics*, vol. 532, 2011, https://doi.org/10.1051/0004-6361/201116836.

[4] https://ntrs.nasa.gov/citations/20040082139

[5] https://www.sciencedirect.com/topics/engineering/solar-hour-angle#:~:text=The%20hour%20angle%20is%20the,times%201.5%20hours%20before%20noon).

VII GitHub and Contact

The following link is to the GitHub repository with all the data and code…
https://github.com/DanW321/Milankovitch-Cycles-Animation/tree/main/milankovitch
Email any questions to…
danwexler32@gmail.com or *daniel_wexler@brown.edu*
Thank you to Timothy Herbert and Weimin Si for their help with this project!