3D PRINTING AND DESIGN REFERENCE DOCUMENT	
Document Title:	Generating an analemma svg with Python code
Document No.:	1729092037
Author(s):	jattie
Contributor(s):	

REVISION HISTORY

Revision	Details of Modification(s)	Reason for modification	Date	By
0	Draft release	Python script to generate analemma svg	2024/10/16 15:20	jattie

Analemma SVG

The work here is inspired by the designs and meticulous documentation provided by @yba2cuo3 on printables. [1] I wanted a pure python solutions without using blender.

As a 3D hobbyist we sometime want to do complicated things that is hard to do accurately in 3D design tools. Since I discovered the use of SVG's and python tools to generate them, I wondered if I can reproduce an Analemma to use on a sundial using Python code. After many wasted hours and trail and error working out angular math in python I can report that it is in fact quite feasible.

The catch was converting to radians.

The python code

With lots of credit to copilot, I eventually managed to crack the problem.

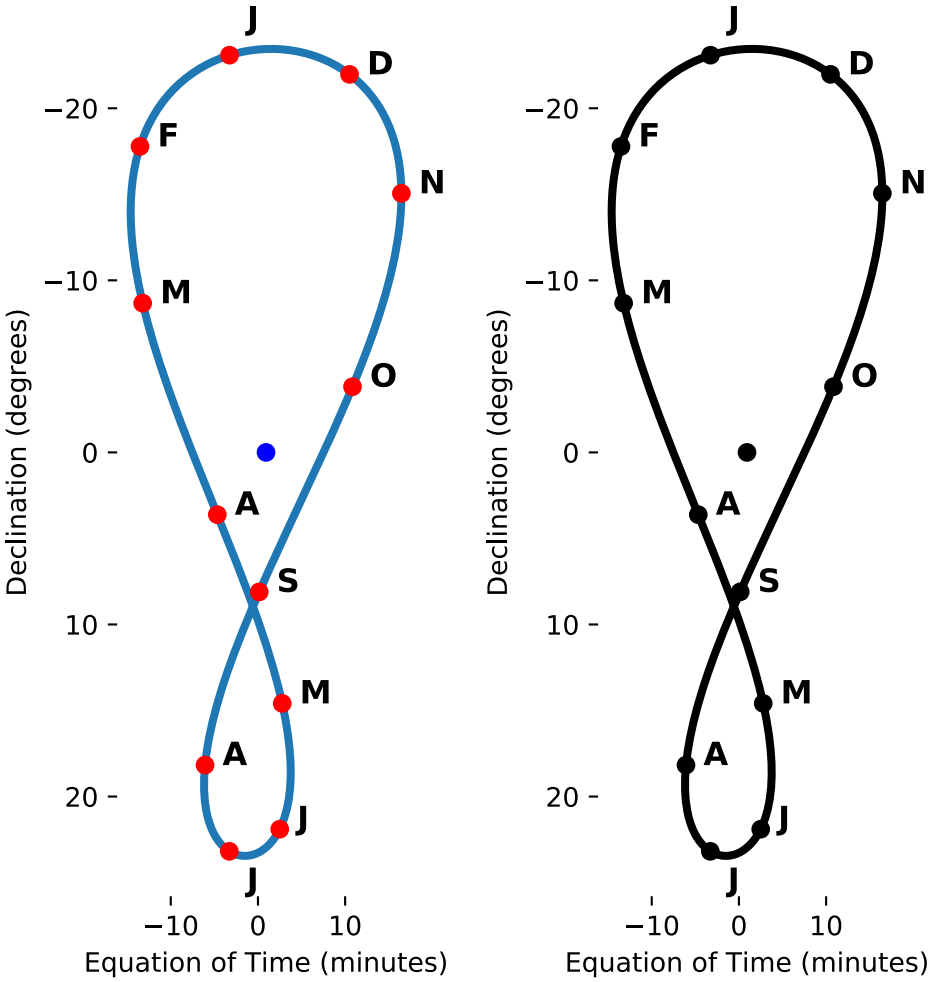
generate_analemma.py

```
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
usecolour=True

# Function to calculate the analemma
def calculate_analemma():
    days = np.arange(0, 365)
    declination = []
    equation_of_time = []

    for day in days:
        B = (360 / 365) * (day - 81)
        EoT = 9.87 * np.sin(np.radians(2 * B)) - 7.53 * np.cos(np.radians(B)) - 1.5 * np.sin(np.radians(B))
        decl = 23.45 * np.sin(np.radians((360 / 365) * (day - 81)))
        equation_of_time.append(EoT)
        declination.append(decl)
```

```
    return days, declination, equation_of_time

# Calculate analemma
days, declination, equation_of_time = calculate_analemma()

# Plot the analemma with reduced width to 1/4
plt.figure(figsize=(2, 6)) # Reduced width to 1/4
if usecolour:
  plt.plot(equation_of_time, declination, label='Analemma', linewidth=3)
else:
  plt.plot(equation_of_time, declination, label='Analemma', linewidth=3, color='black')

# Add plot points for the first day of every month
first_days = [datetime(2023, month, 1) for month in range(1, 13)]
first_days_indices = [(day - datetime(2023, 1, 1)).days for day in first_days]
month_labels = ['J', 'F', 'M', 'A', 'M', 'J', 'J', 'A', 'S', 'O', 'N', 'D']

c=0 # index counter for declanation offset
for i, label in zip(first_days_indices, month_labels):
    dec_offset=[-1.5, 0.0, 0.0, 0.0, 0.0, 0.0, 2.3, 0.0, 0.0, 0.0, 0.0, 0.0] # keep the labels off the lines
    if usecolour:
      plt.plot(equation_of_time[i], declination[i], 'ro')  # Red points for the first day of each month
    else:
      plt.plot(equation_of_time[i], declination[i], 'ko')  # Red points for the first day of each month
    plt.text(equation_of_time[i] + 2.0, declination[i] + dec_offset[c], label, fontsize=12, ha='left',
weight='bold') # Move labels higher
    c+=1 # increment counter

plt.xlabel('Equation of Time (minutes)')
plt.ylabel('Declination (degrees)')
#plt.title('Analemma with First Day of Each Month')
plt.gca().invert_yaxis()  # Flip the plot on the y-axis
plt.grid(False)  # Remove gridlines

# Remove the plot box
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(False)

# Remove the legend
plt.legend().set_visible(False)

# Add a small dot to the center point of the plot for alignment purposes
center_x = (max(equation_of_time) + min(equation_of_time)) / 2
center_y = (max(declination) + min(declination)) / 2

if usecolour:
  plt.plot(center_x, center_y, 'bo') # Blue dot at the center
else:
  plt.plot(center_x, center_y, 'ko') # Black dot at the center

# Save the plot as an SVG file
if usecolour:
  plt.savefig("analemma_plot_colour.svg", format="svg", bbox_inches='tight')
else:
  plt.savefig("analemma_plot.svg", format="svg", bbox_inches='tight')
plt.show()

print("plot saved: 'analemma_plot.svg'.")
```

The final result

The final result is perfect for importing and extruding on a CAD package supporting SVG imports.



Hover over and right click to save the svg

This format converted from a plot renders acceptably, but fails to properly import into at least Fusion 360. It is not very useful for 3D printing purposed. Some online tool may do a better job at converting this properly for CAD use.

1)

https://www.printables.com/model/1036568-curved-analemma-plate-for-a-heliochronometer-sundi