REPORT – DA Assignment 01

Author – Anshul Kumrawat

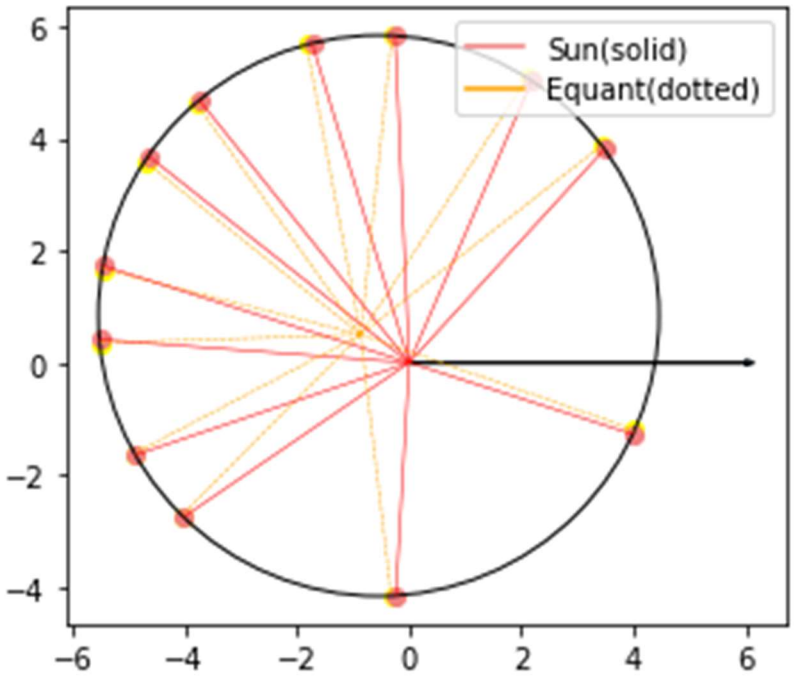
Results:

Best Parameter Values:

- **c** = 152.902143
- **r**= 8.31703
- **e1**= 1.54455535
- **e2**= 93.197523
- **z**= 55.7427834
- **s**= 0.52408795
- **errors** = [0.04971412, 0.05501278, 0.05569381, 0.02783268, 0.05569223, 0.05569325, 0.05479902, 0.05490818, 0.00512666, 0.04647835, 0.05014042, 0.04731288]
- **maxError** = 0.05569380651371519

Diagram: Mars Orbit

According to my code implementation, the results are visualised as:



Implementation Summary:

Assumptions made:

- The Sun is at origin. (0,0)
- The orbit has radius r.
- The reference longitude (equant 0) is given by angle z (degrees) with respect to Aries.
- The angular velocity of Mars around the equant is s degrees per day.

Implementation Details:

- Firstly, I assumed the first date of data as a reference and it intersects the circle at angle **z** from the equant point with respect to Aries.
- Then, calculated **Oppositions** 2Darray which consist of 12 rows and 2 columns:
    - **Days**: Difference between each date with the first date taken as reference.
    - **Longitude**: The actual angle in degrees and in decimals of the mars from the sun.
- Implemented helper functions:
    - **line_function():** This function takes either two points or one point and a angle, for finding the slope and intercept of a line with the standard formulae.
    - **Intersection_func(circle, line, angle):** This function is used to find the exact point(x, y) at which the line intersects with a circle.

- Approach used inside **MarsEquantModel():**
    - To find the angles of all the oppositions from the equant, iteratively I used the formula:
        - Angle **= (oppositions[i][0] * s + z) % 360.**
        - Here, **oppositions[i][0]** returns first column values of opposition array defined as days-difference, which is further multiplied with angular velocity of mars**(s)** *[given in degrees]* and it adds up with the angle **z** *[given in degrees]* which is taken as a reference angle from the equant w.r.t Aries. Likewise at each iteration, angle of oppositions is calculated.
    - Calculated the slope and intercept of **Solid- line** from the sun with help of line_function().
    - Similarly calculated the slope and intercept of **dotted-line from the equant**.
    - Then, calculated the intersection point (x, y) between the dotted-line (from equant) and the circle.
    - Then with this point (x, y), I calculate the slope of **dotted-line (from sun).**
    - Then for finding the angle difference between the dotted-line (from sun) and solid line (from sun), I used the standard formula:
        - $m = \tan^{-1}(\text{absolute}((m_2 - m_1)/ (1 + m_1 * m_2)))$.
        - This m calculated is the final angle difference which is taken as error.
    - This procedure is done for each opposition data and finally this function returns the **errors** array (size of 12) and the **maxError** (from errors array).

- For rest of functions,
    - **bestOrbitInnerParams(r, s , oppositions) ,**
    - **bestS(r , oppositions),**
    - **bestR(s, oppositions),**
    - **bestMarsOrbitParams(oppositions)**

        - I've used **scipy.optimize** function first to find the best values of parameters which minimizes our objective function.
        - Then, I've applied discretised exhaustive method by manually initializing the parameters and giving the range of parameters with deeper precision to each function for getting the optimal minimal error.

Derivation:

To find Intersection Point between a Line and a Circle:

Equation of Line: $y = mx + c$ (1)

Equation of Circle: $(x - a)^2 + (y-b)^2 = r^2$ (2)

Put the values of equation (1) in equation (2), We get,

- $x^2 + a^2 – 2ax + y^2 + b^2 – 2yb - r^2 = 0$
- $x^2 + a^2 – 2ax + (mx+c)^2 + b^2 – 2(mx+c)*b – r^2= 0$
- $x^2 + a^2 -2ax + m^2 x^2 + c^2 + 2mcx + b^2 – 2mxb – 2bc – r^2= 0$
- $[1 + m^2]*x^2 + 2*[mc – a - mb]x + (a^2 + b^2 + c^2 – 2bc – r^2) = 0$

Now, we can solve this quadratic equation to get the values of x. After getting the values of x, put it in equation (1) to get the values of y.

This way we can find the intersection points between a circle and a line.