

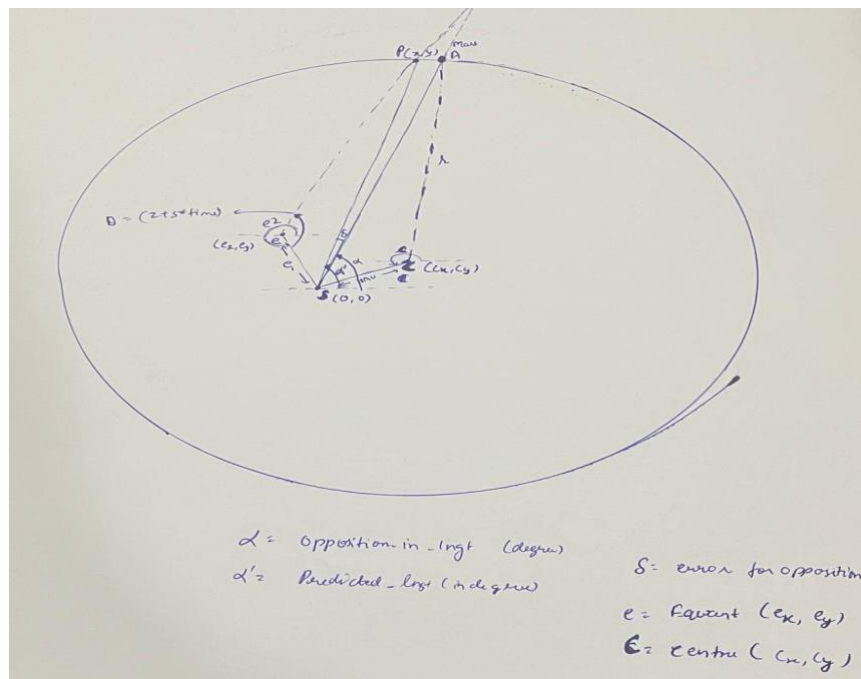
Assignment -2 : Mars Orbit

Harsh Gupta (Sr. No : 20961)

Sept 7, 2022

Problem 1:

Here, Is the general Diagram which give basic idea about how our picture looks like:



Main Idea:

Here we have to find pred_angle_for_opposition on the basis of certain parameters

C = centre with (c_x, c_y) Cartesian coordinate with distance 1 unit from sun

E = equant Point with (e_x, e_y) Cartesian coordinate with distance e_1 from sun

To get pred_angle, we have to solve two equation then find the slope of that line

i.e. (x, y) is the intersectionPoint on Circle that followed by Mars.

So,

$$ex, ey = e1 * \cos(e2), e1 * \sin(e2)$$

$$cx, cy = 1 * \cos(c), 1 * \sin(c)$$

We have two equations for each opposition:

$$(x-cx)^2 + (y-cy)^2 = r^2 \text{ ----- (i) and } D = (z + s * \text{time})$$

$$(y - ey) = (x - ex) * \tan(D) \text{ ----- (ii)}$$

On putting value of y from eq(ii) in eq(i), we will get,

$$(1 + (\tan D)^2) x^2 + (-2 * cx + 2 * \tan D * (ey - cy - ex * \tan D)) x + (ey - cy - ex * \tan D)^2 + cx^2 + r^2 = 0$$

On Solving this equation we will get x and y ,

Then ,

$$\text{pred_angle} = \tan^{-1}(y/x)$$
$$\text{Error} = \text{pred_angle} - \text{actual_angle}$$

Function :

Errors ,maxError = MarsEquantModel (c,r,e1,e2,z,s,times,oppositions)

Result:

```
c: 142,  
e1: 2,  
e2: 150  
z: 75,  
r: 4,  
s: 0.5240174672489083,  
Errors : [32.91337259 23.72922061 13.9227638 5.26297032 0.40114213 7.22970597  
33.43681242 35.41758935 27.19055958 17.40072236 8.05568416 1.22557743],  
maxError: 35.417589346075225
```

Problem 2:

As , In this we have to find Best Orbit Inner Parameters ie. (c,e1,e2,z) for given r and s

So, I have used Exhaustive search, I have taken each value on certain range independent but in cyclic order(kind of round Robin Approach) for each parameter

After this, we have used minimize function on given loss_function to get max_error to be minimum to get these parameters.Exhaustive Search is done by fixing remaining parameter fixed ,and searching for only single parameter.

Function:

c,e1,e2,z,errors,maxError = BestOrbitInnerParams(r,s,times,oppositions)

Result:

```
c,e1,e2,z , errors, maxError = BestOrbitInnerParams(r,s,times,oppositions)

print("c: {}, \ne1: {}, \ne2: {} \nz: {}, \nr: {}, \ns: {}, \nErrors : {}, \nmaxError: {}".format(c,e1,e2,z,r,s,errors,maxError))
✓ 1.1s

c: 145.31253646691158,
e1: 1.8374473457147045,
e2: 148.86864384217463
z: 56.17185751386123,
r: 10,
s: 0.5240174672489083,
Errors : [ 0.11620526 0.21068574 0.20832503 0.12868693 0.16400765 0.21068564
-0.00173464 -0.21068566 -0.14623771 -0.10726618 -0.1881287 -0.21068575],
maxError: 0.2106857548787957
```

Problem 3:

So, I have used Exhaustive search, I have taken each value on certain range independent but in cyclic order(kind of round Robin Approach) for each parameter

Here, we have searched for s in range of $(360/(678+1))$ to $((360/(678-1))$ for about 100 of values and check for good one to give result on the basis of function that is Generated in Problem 2

Function:

s,errors,maxError = BestS(r,s,times,oppositions)

Result:

```
s, errors, maxError =BestS(r,times,oppositions)
print("\nr: {}, \ns: {}, \nErrors : {}, \nmaxError: {}".format(r,s,errors,maxError))
[75] ✓ ✓ 21.8s
...

r: 10,
s: 0.5240587229821112,
Errors : [ 0.00544714 0.12265208 0.10056051 0.03972753 0.11883743 0.1240421
-0.12404236 -0.12404226 -0.01295638 -0.00878314 -0.09113694 -0.05756545],
maxError: 0.12404236214013054
```

Problem 4:

So, I have used Exhaustive search, I have taken each value on certain range independent but in cyclic order(kind of round Robin Approach) for each parameter

Here, we have searched for r in range of (4 to 10) for about 100 of values and check for good one to give result on the basis of function that is Generated in Problem 2

Function:

r,errors,maxError = BestR(s,times,oppositions)

Result:

```
r,errors, maxError = BestR(s,times,oppositions)
print("\nr: {}, \ns: {}, \nErrors : {}, \nmaxError: {}".format(r,s,errors,maxError))

✓ 1m 5.3s

/var/folders/_h/5bg4_75534q1s_dc1vxxdh1w0000gn/T/ipykernel_7814/603931631.py:19: RuntimeWarning: invalid value encountered in sqrt
delta = np.sqrt(b**2 - 4 * a * d)

r: 8.16949152542373,
s: 0.5240587229821112,
Errors : [ 0.05106009  0.04905623  0.07257818  0.07186335  0.07253875 -0.07252674
 -0.00587079  0.00707774 -0.06422803 -0.06605741 -0.0725782  -0.05209416],
maxError: 0.07257820422367445
```

Problem 5:

It is wrapper function, in this function, I have exhaustively searched for best values of r and s from some value in order to minimize the maximum error.

I have called BestR() and BestS() function , one after other in iterative manner to update value of r and s , break the iteration , if I get error less than 4 minute or value of r and s not updated in further iterations.

Function:

r,s,c,e1,e2,z,errors,maxError= BestMarsOrbitParams(times,oppositions)

Result:

- I am getting max_Error as **4.35 minutes** rounded upto 2 decimal places which is 21 seconds away from desired results.

```
MaxError 0.17203617358518386 with bestR = 8.16949152542373 and S = 0.5240174672489083
MaxError 0.07257820422367445 with R = 8.16949152542373 and BestS = 0.5240587229821112
MaxError 0.07257820422367445 with bestR = 8.16949152542373 and S = 0.5240587229821112
MaxError 0.07257820422367445 with R = 8.16949152542373 and BestS = 0.5240587229821112
Fit parameters: r = 8.1695, s = 0.5241, c = 148.7596, e1 = 1.5184, e2 = 149.0184, z = 55.9302
The maximum angular error = 0.0726
```

