Assignment #1 - Astronomia de Sistemas Planetários

Arthur Araujo Galdino dos Santos

Disponível em: https://github.com/araujoarthur/psastronomyclass/tree/main/A1

Packages and Constant Definition

In [76]:
```
using DataFrames, PlotlyJS
const possibleBodies = ["Mercury", "Venus", "Earth", "Mars", "Jupyter", "
const au = 149597870700
```

```
WARNING: redefinition of constant possibleBodies. This may fail, cause in
correct answers, or produce other errors.
```

Out[76]:
```
149597870700
```

Functions definition

Orbital Element Calculation Functions:

In [53]:
```
function Ω(body::String, t::Float64) # \Omega
    if !(body ∈ possibleBodies)
        return nothing
    elseif body == "Mercury"
        return 48.3333 + 3.24587E-5 * t
    elseif body == "Venus"
        return 76.6799 + 2.46590E-5 * t
    elseif body == "Earth"
        return 0
    elseif body == "Mars"
        return 49.5574 + 2.11081E-5 * t
    elseif body == "Jupyter"
        return 100.4541 + 2.76854E-5 * t
    elseif body == "Saturn"
        return 113.6634 + 2.38980E-5 * t
    elseif body == "Uranus"
        return 74.0005 + 1.3978E-5 * t
    elseif body == "Neptune"
        return 131.7806 + 3.0173E-5 * t
    end
end

function ι(body::String, t::Float64) # \isansi
    if !(body ∈ possibleBodies)
        return nothing
    elseif body == "Mercury"
        return 7.0047 + 5E-8 * t
    elseif body == "Venus"
        return 3.3946 + 2.75E-8 * t
    elseif body == "Earth"
        return 0
    elseif body == "Mars"
        return 1.8497 - 1.78E-8 * t
    elseif body == "Jupyter"
        return 1.3030 - 1.557E-7 * t
    elseif body == "Saturn"
        return 2.4886 - 1.081E-7 * t
    elseif body == "Uranus"
        return 0.7733 + 1.9E-8 * t
    elseif body == "Neptune"
        return 1.7700 - 2.55E-7 * t
    end
end

function ω(body::String, t::Float64) # \omega
    if !(body ∈ possibleBodies)
        return nothing
    elseif body == "Mercury"
        return 29.1241 + 1.01444E-5 * t
    elseif body == "Venus"
        return 54.8910 + 1.38374E-5 * t
    elseif body == "Earth"
        return 282.9404 + 4.70935E-5 * t
    elseif body == "Mars"
        return 286.5016 + 2.92961E-5 * t
    elseif body == "Jupyter"
        return 273.8777 + 1.64505E-5 * t
    elseif body == "Saturn"
        return 339.3939 + 2.97661E-5 * t
    elseif body == "Uranus"
        return 96.6612 + 3.0565E-5 * t
    elseif body == "Neptune"
        return 272.8461 - 6.027E-6 * t
        return 272.8461 - 6.027E-6 * t
    end
end

function 𝒶(body::String, t=0::Float64) # \scra

    if !(body ∈ possibleBodies)
        return nothing
    elseif body == "Mercury"
        return 0.387098*au
    elseif body == "Venus"
        return 0.723330*au
    elseif body == "Earth"
        return 1*au
    elseif body == "Mars"
        return 1.523688*au
    elseif body == "Jupyter"
        return 5.20256*au
    elseif body == "Saturn"
        return 9.55475*au
    elseif (body == "Uranus") && (t ≠ 0)
        return (19.18171 - 1.55E-8 * t) * au
    elseif (body == "Neptune") && (t ≠ 0)
        return (30.05826 + 3.313E-8 * t) * au
    else
        return nothing
    end
end

function 𝐞(body::String, t::Float64) # \bfe
    if !(body ∈ possibleBodies)
        return nothing
    elseif body == "Mercury"
        return 0.205635 + 5.59E-10 * t
    elseif body == "Venus"
        return 0.006773 - 1.302E-9 * t
    elseif body == "Earth"
        return 0.016709 - 1.151E-9 * t
    elseif body == "Mars"
        return 0.093405 + 2.516E-9 * t
    elseif body == "Jupyter"
        return 0.048498 + 4.469E-9 * t
    elseif body == "Saturn"
        return 0.055546 - 9.499E-9 * t
    elseif body == "Uranus"
        return 0.047318 + 7.45E-9 * t
    elseif body == "Neptune"
        return 0.008606 + 2.15E-9 * t
    end
end

function 𝑴(body::String, t::Float64) # \bisansM
    if !(body ∈ possibleBodies)
        return nothing
    elseif body == "Mercury"
        return 168.6562 + 4.0923344368 * t
    elseif body == "Venus"
        return 48.0052 + 1.6021302244 * t
    elseif body == "Earth"
        return 356.0470 + 0.9856002585 * t
    elseif body == "Mars"
        return 18.6021 + 0.5240207766 * t
    elseif body == "Jupyter"
        return 19.8950 + 0.0830853001 * t
    elseif body == "Saturn"
        return 316.9670 + 0.0334442282 * t
    elseif body == "Uranus"
        return 142.5905 + 0.011725806 * t
    elseif body == "Neptune"
        return 260.2471 + 0.005995147 * t
    end
end
```

```
Out[53]: M (generic function with 1 method)
```

Arc Correction for E and M

```
In [65]: function arcCorrection(ARC::Float64)
             println("ARC: ", ARC)
             if ARC >= 360.0
                 println("CORR: ", mod(ARC,360.0))
                 return mod(ARC,360.0)
             else
                 println("NOT CORR")
                 return ARC
             end
         end
```

```
Out[65]: arcCorrection (generic function with 1 method)
```

Time Function:

```
In [55]: function t(d::Integer, m::Integer, y::Integer, h::Integer, mm::Integer) #
             return 367 * y - floor(7*(y + ((m+9)/12) )/4) + floor(275*m/9) + d -
         end
```

```
Out[55]: t (generic function with 1 method)
```

Eccencentric Anomaly and True Anomaly

```
In [56]: function E(M::Float64, ec::Float64, Ẽ=M::Float64) # \bisansE
             ΔE = (M - Ẽ + (ec * sind(Ẽ)))/(1 - (ec * cosd(Ẽ)))
             E = Ẽ + ΔE;

             if (ΔE > 5E-6)
                 return E(M, ec, E)
             else
                 return E
             end
         end

         function ν(ec::Float64, E::Float64) # \nu
             return atand(sqrt((1 + ec)/(1 - ec)) * tand(E/2))
         end
```

```
Out[56]: ν (generic function with 1 method)
```

Distance from planet to Sun

```
In [57]: function r(axis::Float64, ec::Float64, nu::Float64)
             return (axis * (1 - (ec^2)))/(1 + (ec * cosd(nu)))
         end
```

Out[57]:
```
r (generic function with 1 method)
```

Cartesian Coordinates

In [63]:
```
function cartesian_x(_r::Float64, _Ω::Float64, _ω::Float64, _ν::Float64,
    return _r * ( (cosd(_Ω) * cosd(_ω + _ν)) - (sind(_Ω) * sind(_ω + _ν)
end

function cartesian_y(_r::Float64, _Ω::Float64, _ω::Float64, _ν::Float64,
    return _r * ( (sind(_Ω) * cosd(_ω + _ν)) + (cosd(_Ω) * sind(_ω + _ν)
end

function cartesian_z(_r::Float64, _ω::Float64, _ν::Float64, _𝜄::Float64)
    return _r * sind(_ω + _ν) * sind(_𝜄)
end
```

Out[63]:
```
cartesian_z (generic function with 1 method)
```

Heliocentric Ecliptic Coordinates

In [59]:
```
function ℓ(X::Float64, Y::Float64) # \ell
    return atand(X/Y)
end

function 𝒷(X::Float64, Y::Float64, Z::Float64) # \bscrb
    return Z/(sqrt((X^2) + (Y^2)))
end
```

Out[59]:
```
𝒷 (generic function with 1 method)
```

Calculation of time for my date (01/07/2053 00:35)

In [60]:
```
givenTime = 𝑡(1,7,2053, 0, 35)
```

Out[60]:
```
19540.024305555555
```

Orbital Elements of Mercury

In [70]:
```
orbitalElements = Dict()
for planet ∈ possibleBodies
    orbitalElements[planet] = Dict()
    orbitalElements[planet]["Ω"] = Float64(Ω(planet, givenTime))
    orbitalElements[planet]["𝜄"] = Float64(𝜄(planet, givenTime))
    orbitalElements[planet]["ω"] = Float64(ω(planet, givenTime))
    orbitalElements[planet]["𝒶"] = Float64(𝒶(planet, givenTime))
    orbitalElements[planet]["𝐞"] = 𝐞(planet, givenTime)
    orbitalElements[planet]["𝑴"] = arcCorrection(𝑴(planet, givenTime))
    orbitalElements[planet]["𝑬"] = 𝑬(orbitalElements[planet]["𝑴"], orbita
    orbitalElements[planet]["ν"] = ν(orbitalElements[planet]["𝐞"], orbita
    orbitalElements[planet]["r"] = r(orbitalElements[planet]["𝒶"], orbita
    orbitalElements[planet]["X"] = cartesian_x(orbitalElements[planet]["r
    orbitalElements[planet]["Y"] = cartesian_y(orbitalElements[planet]["r
    orbitalElements[planet]["Z"] = cartesian_z(orbitalElements[planet]["r
    orbitalElements[planet]["ℓ"] = ℓ(orbitalElements[planet]["X"], orbita
    orbitalElements[planet]["𝒷"] = 𝒷(orbitalElements[planet]["X"], orbita
end
```

```
ARC: 80132.97056153399
CORR: 212.9705615339917
ARC: 31353.668725441174
CORR: 33.66872544117359
ARC: 19614.700006651838
CORR: 174.7000066518376
ARC: 10257.980811380097
CORR: 177.9808113800973
ARC: 1643.3837833883772
CORR: 203.3837833883772
ARC: 970.4680319085465
CORR: 250.46803190854655
ARC: 371.7130342422291
CORR: 11.713034242229128
ARC: 377.39241809537845
CORR: 17.392418095378446
```

In [72]:

```
DataTable = DataFrame(PLANET=possibleBodies)
DataTable[! ,"Ω (°)"] = [orbitalElements[planet]["Ω"] for planet in possi
DataTable[!, "𝓲 (°)"]= [orbitalElements[planet]["𝓲"] for planet in possibl
DataTable[!, "ω (°)"] = [orbitalElements[planet]["ω"] for planet in possi
DataTable[!, "𝓪 (m)"] = [orbitalElements[planet]["𝓪"] for planet in possil
DataTable.𝐞 = [orbitalElements[planet]["𝐞"] for planet in possibleBodies]
DataTable[!, "𝑴 (°)"] = [orbitalElements[planet]["𝑴"] for planet in poss:
DataTable[!, "𝑬 (°)"] = [orbitalElements[planet]["𝑬"] for planet in possi
DataTable[!, "ν (°)"] = [orbitalElements[planet]["ν"] for planet in possi
DataTable[!, "r (m)"] = [orbitalElements[planet]["r"] for planet in possi
DataTable[!, "X (m)"] = [orbitalElements[planet]["X"] for planet in possi
DataTable[!, "Y (m)"] = [orbitalElements[planet]["Y"] for planet in possi
DataTable[!, "Z (m)"] = [orbitalElements[planet]["Z"] for planet in possi
DataTable[!, "ℓ (°)"] = [orbitalElements[planet]["ℓ"] for planet in possi
DataTable[!, "𝓫 (°)"] = [orbitalElements[planet]["𝓫"] for planet in possil

show(DataTable, allcols=true)
```

```
8×15 DataFrame
 Row │ PLANET   Ω (°)     𝑖 (°)     ω (°)     𝑎 (m)       e            𝑀
(°)     𝐸 (°)     ν (°)      r (m)       X (m)       Y (m)       Z (m)
ℓ (°)     𝒷 (°)
     │ String   Float64   Float64   Float64   Float64     Float64     Flo
at64   Float64   Float64    Float64     Float64     Float64     Float64
Float64   Float64
─────┼─────────────────────────────────────────────────────────────────────
───────────────────────────────────────────────────────────────────────────
────────────────────────────

   1 │ Mercury   48.9675  7.00568    29.3223  5.7909e10   0.205646    21
2.971   212.875   -76.5327   5.29253e10   5.26817e10   1.81325e9   -4.737
15e9   88.0287  -0.089867
   2 │ Venus     77.1617  3.39514    55.1614  1.08209e11  0.00674756   3
3.6687   33.6725   16.9437   1.0751e11   -9.22356e10   5.4902e10    6.058
89e9   -59.2374   0.0564463
   3 │ Earth      0.0     0.0       283.861   1.49598e11  0.0166865   17
4.7     174.702    87.3946   1.49443e11   1.46569e11   2.91681e10   0.0
78.7448   0.0
   4 │ Mars      49.9699  1.84935   287.074   2.2794e11   0.0934542   17
7.981   177.984    89.0822   2.25612e11   9.13358e10   2.06287e11   2.025
96e9    23.8819   0.00898021
   5 │ Jupyter  100.995   1.29996   274.199   7.78292e11  0.0485853   20
3.384   203.365   -78.8576   7.69232e11   3.41214e11  -6.89398e11  -4.617
12e9   -26.3329  -0.00600235
   6 │ Saturn   114.13    2.48649   339.976   1.42937e12  0.0553604   25
0.468   250.417   -56.2729   1.38249e12   1.09074e12   8.47453e11  -5.827
07e10   52.1546  -0.0421865
   7 │ Uranus    74.2736  0.773671   97.2584  2.8695e12   0.0474636    1
1.713    11.7227    6.14432  2.73401e12  -2.73153e12   1.10781e11   3.591
11e10  -87.6776   0.0131361
   8 │ Neptune  132.37    1.76502   272.728   4.49675e12  0.00864801   1
7.3924   17.395     8.77188  4.45831e12   2.62715e12   3.59951e12  -1.345
61e11   36.1243  -0.0301959
```

New DataFrame to Graph in Polar Coordinates

Generating dataframe

In [82]:
```
graphData = DataFrame(PLANET=possibleBodies, RADIUS=[i for i in 1:length(
```

Out[82]: 8 rows × 3 columns

	PLANET	RADIUS	POLARCOORD
	String	Int64	Float64
1	Mercury	1	88.0287
2	Venus	2	-59.2374
3	Earth	3	78.7448
4	Mars	4	23.8819
5	Jupyter	5	-26.3329
6	Saturn	6	52.1546
7	Uranus	7	-87.6776
8	Neptune	8	36.1243

Graphing

```
In [83]: plot(scatterpolar(graphData, r=:RADIUS, theta=:POLARCOORD, color=:PLANET,
```

```
Out[83]:
```

WebIO not detected.

Please read the troubleshooting guide for more information on how to resolve this issue.

https://juliagizmos.github.io/WebIO.jl/latest/troubleshooting/not-detected/

```
In [ ]:
```