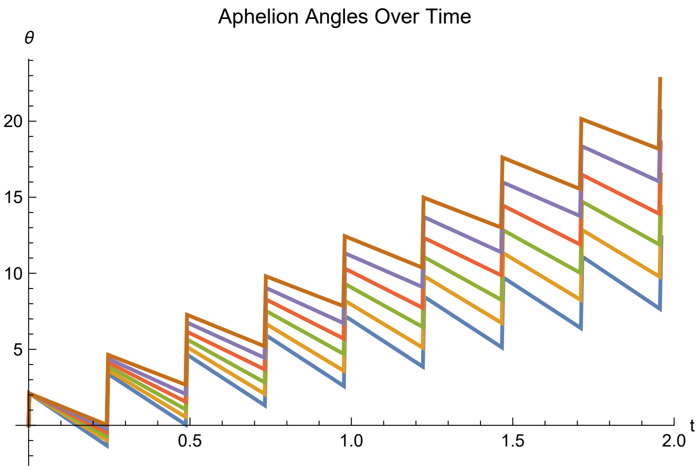
Simulation of Orbital Motion: An Aphelion Angle Study.

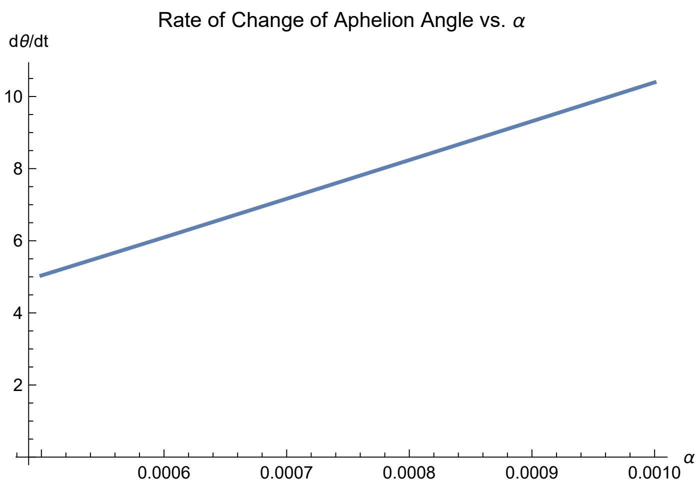
In[39]:=
```
(*Initial conditions*)x0 = 0.47;          (*Initial x position (meters)*)
y0 = 0;           (*Initial y position (meters)*)
vx0 = 0;           (*Initial x velocity (m/s)*)
vy0 = 8.2;          (*Initial y velocity (m/s)*)

(*Alpha values for different scenarios*)
α = Table[i * 10^-4, {i, 5, 10, 1}]; (*Range of alpha values from 0.0005 to 0.0010*)

(*Time parameters*)
ts = 0;            (*Start time (seconds)*)
tf = 2;            (*End time (seconds)*)
dt = 0.0001;          (*Time step (seconds)*)

(*Euler-Cromer Function Definition*)
EulerCromer[ti_, tf_, dt_, x0_, y0_, vx0_, vy0_, α_] :=
  Module[{x, y, vx, vy, r, tRange, tLength, trajectory},
   (*Create time range and initialize length*)tRange = Range[ti, tf, dt];
   (*List of time values from ti to tf with step dt*)tLength = Length[tRange];
   (*Length of the time list*)
   (*Initialize position and velocity arrays*)x = ConstantArray[0, tLength + 1];
   (*Array for x positions*)y = ConstantArray[0, tLength + 1];
   (*Array for y positions*)vx = ConstantArray[0, tLength + 1];
   (*Array for x velocities*)vy = ConstantArray[0, tLength + 1];
   (*Array for y velocities*)(*Set initial conditions*)x[[1]] = x0;
   (*Initial x position*)y[[1]] = y0;
   (*Initial y position*)vx[[1]] = vx0;
   (*Initial x velocity*)vy[[1]] = vy0;
   (*Initial y velocity*)(*Update positions and velocities using the Euler-
    Cromer method*)Do[r = Sqrt[x[[i]]^2 + y[[i]]^2];
    (*Calculate distance from the origin*)
    (*Update x velocity with gravitational force and alpha effect*)
    vx[[i + 1]] = vx[[i]] - (4 * Pi^2 * x[[i]] / (r^3)) * (1 + α / r^2) * dt;
    (*Update x position*)x[[i + 1]] = x[[i]] + vx[[i + 1]] * dt;
    (*Update y velocity with gravitational force and alpha effect*)
    vy[[i + 1]] = vy[[i]] - (4 * Pi^2 * y[[i]] / (r^3)) * (1 + α / r^2) * dt;
    (*Update y position*)y[[i + 1]] = y[[i]] + vy[[i + 1]] * dt, {i, tLength}];
   (*Loop over time steps*)trajectory = Transpose[{x, y}];
   (*Create a trajectory list of (x,y) pairs*)
```

```
   trajectory          (*Return the trajectory*)];

(*Initialize arrays to store positions and aphelion angles*)
pos = ConstantArray[{}, Length[α]]; (*List to hold positions of aphelion*)
aphelionPairs = {};           (*List to hold aphelion angle pairs*)
tL = Range[ts, tf, dt];         (*Time list for analysis*)

(*Loop over each alpha value to simulate trajectory*)
Do[trajectory = EulerCromer[ts, tf, dt, x0, y0, vx0, vy0, α[[i]]];
  (*Compute trajectory for current alpha*)x = trajectory[[All, 1]];
  (*Extract x positions from trajectory*)y = trajectory[[All, 2]];
  (*Extract y positions from trajectory*)(*Find positions where the distance
   is near the initial distance (aphelion)*)Do[r = Sqrt[x[[j]]^2 + y[[j]]^2];
   (*Calculate distance from the origin*)(*Check if at aphelion and record the
    time index*)If[Abs[r - x0] ≤ 0.0001, AppendTo[pos[[i]], j]], {j, Length[tL]}];
  (*Store time and angle at aphelion*)aphelionPairs = Append[aphelionPairs,
    Table[{tL[[n]], 180. / Pi * ArcSin[y[[n]] / Sqrt[x[[n]]^2 + y[[n]]^2]]}, {n, pos[[i]]}]
];
, {i, Length[α]}]; (*Loop over all alpha values*)

(*Plot the aphelion angles over time*)
ListPlot[aphelionPairs, Joined → True, AxesLabel → {"t", "θ"},
 PlotLabel → "Aphelion Angles Over Time"] (*Title for the plot*)

(*Calculate slopes for dTheta/dt for each alpha*)
slopes =
  Table[LinearModelFit[aphelionPairs[[i]], t, t]["BestFitParameters"][[2]], {i, Length[α]}];

(*Prepare data for plotting the rate of change of aphelion angle vs alpha*)
dThetaVsAlpha = Table[{α[[i]], slopes[[i]]}, {i, Length[α]}];

(*Plot the rate of change of aphelion angle vs alpha*)
ListPlot[dThetaVsAlpha, Joined → True, AxesLabel → {"α", "dθ/dt"},
 PlotLabel → "Rate of Change of Aphelion Angle vs. α"] (*Title for the plot*)

(*Linear Fit for dTheta/dAlpha*)
linearFit = LinearModelFit[dThetaVsAlpha, t, t];
(*Fit a linear model to dTheta/dAlpha data*)
Print["Best fit slope for dθ/dα: ", linearFit["BestFitParameters"][[2]],
 " Degrees per year per unit α"](*Output the slope*)
```

Out[52]=



Out[55]=



Best fit slope for dθ/dα: 10 719.5 Degrees per year per unit α