**Simulating Retrograde Motion in MATLAB**

**Abstract**

In this project, I explored a MATLAB script for simulating and visualizing retrograde motion by plotting the relative orbits of Earth, Mars, and the Sun. The script leverages parametric equations to calculate celestial coordinates and uses plotting techniques to dynamically animate orbital paths. To optimize the performance of the simulation, I implemented a "skip" mechanism to improve rendering efficiency when handling a large number of points. Through this process, I gained insights into orbital mechanics, dynamic visualization techniques, and performance optimization strategies in MATLAB.

When studying orbital mechanics, I find it fascinating to simulate and visualize complex phenomena like retrograde motion. Retrograde motion describes how a planet, such as Mars, appears to move backward in the sky from Earth's perspective due to their relative orbits. In this project, I worked on a MATLAB script designed to demonstrate this motion using parametric equations and dynamic plotting.

**Initializing Orbital Parameters**

To start, I set the radii and orbital periods for Earth and Mars relative to the Sun. For Earth, I defined a radius of 1 (in arbitrary units) and a period of 365.25 days. For Mars, I scaled its orbital radius to 1.5 times that of Earth and set its period to 687 days. These parameters allow me to capture the relative motion of Mars and Earth accurately.

matlab

Copy code

% Define orbital parameters for Earth and Mars

r\_earth = 1; % Earth's orbital radius

T\_earth = 365.25; % Earth's orbital period (days)

r\_mars = 1.5; % Mars's orbital radius relative to Earth

T\_mars = 687; % Mars's orbital period (days)

t\_max = 4 \* T\_earth; % Total simulation time (4 Earth years)

dt = 7 / 2; % Time step in days (half a week)

**Calculating Orbital Positions**

To model orbital motion, I used parametric equations for circular orbits. I calculated the x and y positions of Earth and Mars relative to the Sun based on their respective angular velocities. This approach ensures precise tracking of their motion over time.

matlab

Copy code

% Generate time vector

time = 0:dt:t\_max;

% Earth's orbital positions

x\_earth = r\_earth \* cos(2 \* pi \* time / T\_earth);

y\_earth = r\_earth \* sin(2 \* pi \* time / T\_earth);

% Mars's orbital positions

x\_mars = r\_mars \* cos(2 \* pi \* time / T\_mars);

y\_mars = r\_mars \* sin(2 \* pi \* time / T\_mars);

**Switching to Earth's Reference Frame**

To simulate Mars's retrograde motion, I transformed the coordinates into Earth's reference frame. This required subtracting Earth's position from the positions of the Sun and Mars, effectively placing Earth at the center of the plot.

matlab

Copy code

% Transform to Earth's reference frame

x\_sun\_rel = 0 - x\_earth; % Sun relative to Earth

y\_sun\_rel = 0 - y\_earth;

x\_mars\_rel = x\_mars - x\_earth; % Mars relative to Earth

y\_mars\_rel = y\_mars - y\_earth;

**Dynamic Visualization with Trails**

To visualize the motion, I plotted trails for the Sun and Mars, dynamically updating the plots at each time step. This involved connecting points from the starting position up to the current time step, creating a growing path for each celestial body.

matlab

Copy code

% Initialize plot

figure;

hold on;

% Loop through time steps for animation

for t = 1:length(time)

% Plot Sun's trail

plot(x\_sun\_rel(1:t), y\_sun\_rel(1:t), 'm-', 'LineWidth', 1);

% Plot Earth's position

plot(0, 0, 'bo', 'MarkerSize', 6, 'MarkerFaceColor', 'b');

% Plot Mars's trail

plot(x\_mars\_rel(1:t), y\_mars\_rel(1:t), 'r-', 'LineWidth', 1);

% Pause for animation effect

pause(0.01);

end

**Performance Optimization with Skipping**

During the simulation, I noticed that smaller time steps significantly slowed down the rendering. To address this, I implemented a "skip" mechanism to skip frames and reduce the number of points plotted at each step, improving performance without sacrificing visual clarity.

matlab

Copy code

% Define skip value

skip = 100;

% Use skip in plotting

for t = 1:skip:length(time)

% Plot Sun's trail with skipping

plot(x\_sun\_rel(1:skip:t), y\_sun\_rel(1:skip:t), 'm-', 'LineWidth', 1);

% Plot Mars's trail with skipping

plot(x\_mars\_rel(1:skip:t), y\_mars\_rel(1:skip:t), 'r-', 'LineWidth', 1);

% Update the plot

drawnow;

end

**Observations**

Through this project, I gained a deeper understanding of retrograde motion and how it arises from the relative orbits of planets. The dynamic visualization provided an intuitive representation of this phenomenon, showcasing how Mars appears to move backward when Earth overtakes it in their respective orbits.

Implementing the "skip" mechanism was particularly enlightening, as it demonstrated the trade-off between computational accuracy and rendering efficiency. By balancing these factors, I optimized the simulation to run smoothly while retaining its informative value.

This project highlighted the power of MATLAB for both numerical simulation and visualization, allowing me to explore celestial mechanics in a hands-on and engaging way.