**Simulating Gravitational Fields: A Journey Through MATLAB**

**Abstract**

In this project, I explored gravitational field simulations for hollow and solid spheres to deepen my understanding of mass distribution and gravitational physics. Using MATLAB, I developed numerical models to visualize how gravitational fields vary inside and outside these objects. Every step in the coding process was carefully thought out to ensure accurate representation of theoretical principles while improving my MATLAB programming skills. This essay describes the development process, including the reasoning behind every line of code, and highlights the insights gained from these simulations.

When I set out to simulate gravitational fields, I wanted to see how mass distribution affects gravitational forces both theoretically and visually. Starting with a hollow sphere was a logical choice—it simplifies the problem by confining all the mass to the surface. From there, I expanded to a solid sphere, which required more complex modeling. MATLAB became my tool of choice for its powerful numerical and visualization capabilities. Below, I document each step of the coding process, explaining my decisions along the way.

**Simulating a Hollow Sphere**

I started by defining the range of radial distances for the hollow sphere. To focus on the outer shell of mass, I used linspace to generate values between 90% of the Earth's radius and its full radius.

matlab

Copy code

% Define the radial values for the hollow sphere, focusing on the outer shell

r = linspace(0.9 \* R, R, 10); % I chose 10 intervals for a balance between computation and smoothness

This range ensures that my model only includes the shell, which is essential for representing a hollow sphere.

Next, I calculated the gravitational field strength using the formula derived from Newton’s law of gravitation.

matlab

Copy code

% Compute the gravitational field strength at each radial point

field\_strength = G \* M ./ (r.^2); % Here, the inverse-square law governs the field's dependence on radius

This line divides the mass and gravitational constant by the square of the radius, aligning with the physical reality that gravity decreases with distance.

I visualized the results with a simple plot to understand how the gravitational field behaves.

matlab

Copy code

% Plot the gravitational field strength as a function of radius

plot(r, field\_strength); % Visualization helps confirm that the field approaches zero inside the hollow sphere

This plot was a critical step in verifying my calculations, showing that the field diminishes to zero inside the hollow sphere.

**Simulating a Solid Sphere**

Modeling a solid sphere introduced more complexity, as I needed to represent mass distributed throughout its volume. I began by dividing the sphere into radial, polar, and azimuthal angles.

matlab

Copy code

% Define angles and radial distances for a solid sphere

theta = linspace(0, pi, 50); % Polar angle: Divides the sphere into latitudinal segments

phi = linspace(0, 2 \* pi, 50); % Azimuthal angle: Divides the sphere into longitudinal segments

r = linspace(0, R, 20); % Radial distances to represent shells within the sphere

These angles and radii divided the sphere into a grid of points, forming the basis for my simulation.

To transform spherical coordinates into Cartesian coordinates, I used:

matlab

Copy code

% Convert spherical coordinates to Cartesian coordinates for 3D visualization

[phi\_grid, theta\_grid, r\_grid] = ndgrid(phi, theta, r); % Create 3D grids for angles and radii

x = r\_grid .\* sin(theta\_grid) .\* cos(phi\_grid); % x-coordinate: Includes radial and angular components

y = r\_grid .\* sin(theta\_grid) .\* sin(phi\_grid); % y-coordinate: Uses sine of azimuthal angle

z = r\_grid .\* cos(theta\_grid); % z-coordinate: Aligns with polar angle for height

This transformation allowed me to visualize the sphere and ensured that every point was accurately placed in 3D space.

Next, I calculated the mass for each shell and segment, proportionally distributing mass based on volume.

matlab

Copy code

% Calculate the volume of each shell and distribute mass proportionally

volume = 4/3 \* pi \* r.^3; % Volume of spheres at each radial distance

mass\_density = M / volume(end); % Uniform density based on total mass and outermost volume

mass\_shell = mass\_density \* diff(volume); % Mass per shell based on volume differences

Finally, I plotted the results to see how the gravitational field behaves inside the sphere.

matlab

Copy code

% Visualize the gravitational field within the solid sphere

surf(x, y, z, 'EdgeColor', 'none'); % Surface plot with no edge lines for smooth visualization

colorbar; % Add color bar to show field strength variations

This visualization revealed how gravitational strength builds linearly toward the surface and then diminishes outward.

**Code Section**

matlab

Copy code

% Define the radial values for the hollow sphere

r = linspace(0.9 \* R, R, 10); % I chose 10 intervals for smoothness

% Compute the gravitational field strength at each radial point

field\_strength = G \* M ./ (r.^2); % Gravity follows the inverse-square law

% Plot the gravitational field strength as a function of radius

plot(r, field\_strength); % Visualization ensures field diminishes inside

% Define angles and radial distances for a solid sphere

theta = linspace(0, pi, 50); % Polar angle for latitude-like divisions

phi = linspace(0, 2 \* pi, 50); % Azimuthal angle for longitude-like divisions

r = linspace(0, R, 20); % Radial distances to divide the sphere into shells

% Convert spherical coordinates to Cartesian coordinates for 3D visualization

[phi\_grid, theta\_grid, r\_grid] = ndgrid(phi, theta, r);

x = r\_grid .\* sin(theta\_grid) .\* cos(phi\_grid); % x-coordinates

y = r\_grid .\* sin(theta\_grid) .\* sin(phi\_grid); % y-coordinates

z = r\_grid .\* cos(theta\_grid); % z-coordinates

% Calculate the volume of each shell and distribute mass proportionally

volume = 4/3 \* pi \* r.^3; % Volumes for each radius

mass\_density = M / volume(end); % Total mass divided by largest volume

mass\_shell = mass\_density \* diff(volume); % Differential mass for each shell

% Visualize the gravitational field within the solid sphere

surf(x, y, z, 'EdgeColor', 'none'); % 3D plot for the field

colorbar; % Color bar to indicate field variations

With this code, I created simulations for both hollow and solid spheres, visualizing gravitational fields in a way that enhances both learning and insight.