`Assignment 1` Data Visualization

Shuifan Wang `shepherd.wang@foxmail.com`

March 20, 2025

Problem 1

$$f(x,y) = x^3y^2\cos(x+y), \quad \frac{x^2}{4} + \frac{y^2}{9} \leq 1.$$

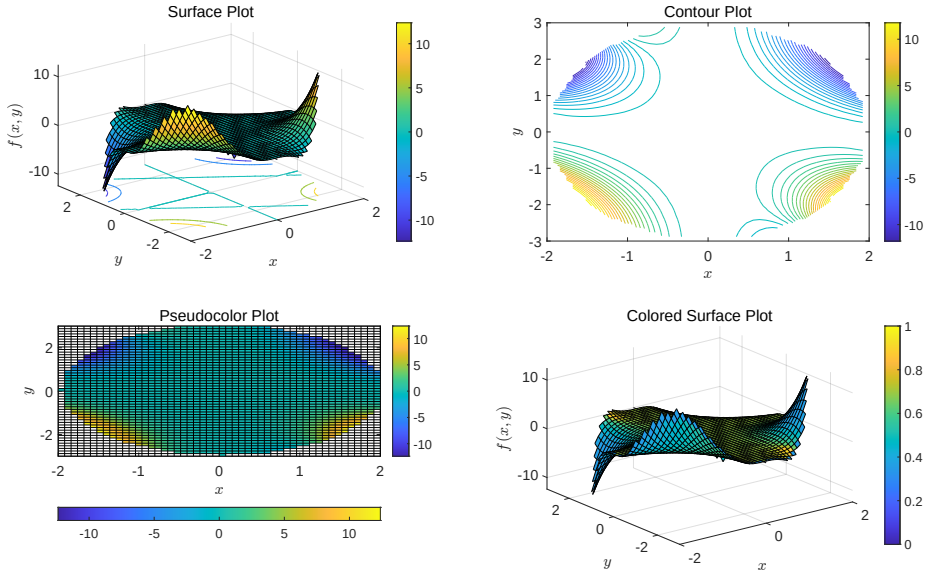

Figure 1: Problem 1

```
clc; clear; close all;

% Define x, y and Generate Grid
x = -2: 0.08: 2;
y = -3: 0.12: 3;
[X, Y] = meshgrid(x, y);

% Define function: f(x,y) = x^3 y^2 cos(x+y)
F = X.^3 .* Y.^2 .* cos(X + Y);

% Define Region Mask: x^2/4 + y^2/9 <= 1
region_mask = (X.^2 / 4 + Y.^2 / 9) <= 1;

% Restrict the Figure in the Region Mask
F(¬region_mask) = NaN;
figure;

% Surface Plot
subplot(2,2,1);
surfc(X, Y, F);
colorbar;
title('Surface Plot');
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$y$', 'Interpreter', 'latex');
zlabel('$f(x,y)$', 'Interpreter', 'latex');
set(gca, 'FontSize', 14);

% Contour Plot
subplot(2,2,2);
contour(X, Y, F, 40); % Plot 40 lines.
colorbar;
title('Contour Plot');
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$y$', 'Interpreter', 'latex');
set(gca, 'FontSize', 14);

% Pseudocolor Plot
subplot(2,2,3);
pcolor(X, Y, F);
colorbar('horiz');
colorbar('vert');
title('Pseudocolor Plot');
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$y$', 'Interpreter', 'latex');
set(gca, 'FontSize', 14);

% Colored Surface Plot with Light
subplot(2,2,4);
surfl(X, Y, F, [45 60]);
material shiny; % More Shiny
colorbar;
title('Colored Surface Plot');
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$y$', 'Interpreter', 'latex');
zlabel('$f(x,y)$', 'Interpreter', 'latex');
set(gca, 'FontSize', 14);

```

```
% Save and Export the Figure
set(gcf, 'Position', [100, 100, 1600, 900]);
exportgraphics(gcf, 'Problem1.pdf', 'ContentType', 'vector');
```

Problem 2

$$f(x,y,t) = e^{1-t}\sqrt{x^2+y^2}, \quad \frac{x^2}{36} + \frac{y^2}{16} \leq 1, \quad t \text{ from } 0 \text{ to } -2.$$

Click here for the video.

```
clc; clear; close all;

% Define x, y, t and Generate Grid
x = -6: 0.24: 6;
y = -4: 0.2: 4;
t = 0: -0.05: -2;
[X, Y] = meshgrid(x, y);

% Define Region Mask: x^2/36 + y^2/16 <= 1
region_mask = (X.^2 / 36 + Y.^2 / 16) <= 1;

% Generate Video Object and Save as *.avi
v = VideoWriter('Problem2.avi');
v.FrameRate = 10;
open(v);

figure;
for T = t
    % Define function: f(x,y,t) = exp(1-t)sqrt(x^2+y^2)
    f = exp(1 - T) .* sqrt(X.^2 + Y.^2);

    % Restrict the Figure in the Region Mask
    f(~region_mask) = NaN;

    % Begin Plotting
    surf(x, y, f, 'EdgeColor', 'none');
    colormap parula;
    colorbar;
    xlabel('$x$', 'Interpreter', 'latex');
    ylabel('$y$', 'Interpreter', 'latex');
    zlabel('$f(x,y,t)$', 'Interpreter', 'latex');
    axis([-6 6 -4 4 0 10]); % Fix the Axis
    view(135, 30); % Adjust Viewing Angle

    % Record the Frames
    frame = getframe(gcf);
    writeVideo(v, frame);
```

```
    pause(0.01); % Adjust the Speed
end

% Close Video Object
close(v);
```

Problem 3

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \cos(x^2 + y^2)$$

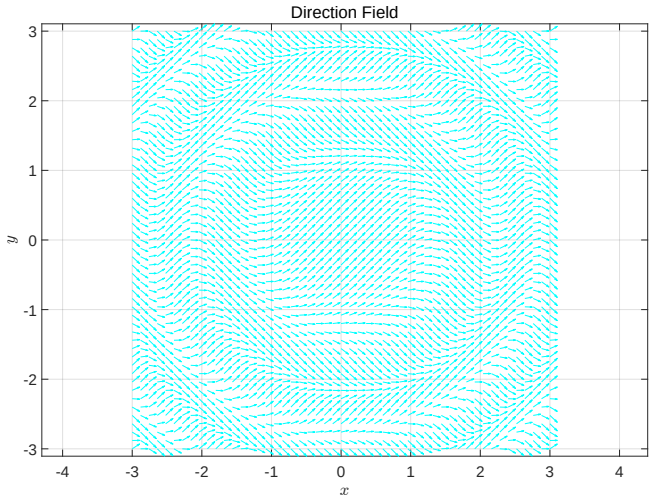

Figure 2: Problem 3

```
clc; clear; close all;

% Define x, y and Generate Grid
x = -3: 0.12: 3;
y = -3: 0.12: 3;
[X, Y] = meshgrid(x, y);

% Calculate the Given ODE dy/dx = cos(x^2 + y^2)
dYdX = cos(X.^2 + Y.^2);

% Fix dx as 1, Then Calculate dy
dx = ones(size(X));
dy = dYdX;

```

```
% Plot the Direction Field
figure;
quiver(x, y, dx, dy, 'Color', [0, 1, 1]); % Change the Color of ...
    Arrows
hold on;
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$y$', 'Interpreter', 'latex');
title('Direction Field');
set(gca, 'FontSize', 18);
grid on;
axis equal;

% Save and Export the Figure
set(gcf, 'Position', [100, 100, 1600, 900]);
exportgraphics(gcf, 'Problem3.pdf', 'ContentType', 'vector');
```

Problem 4

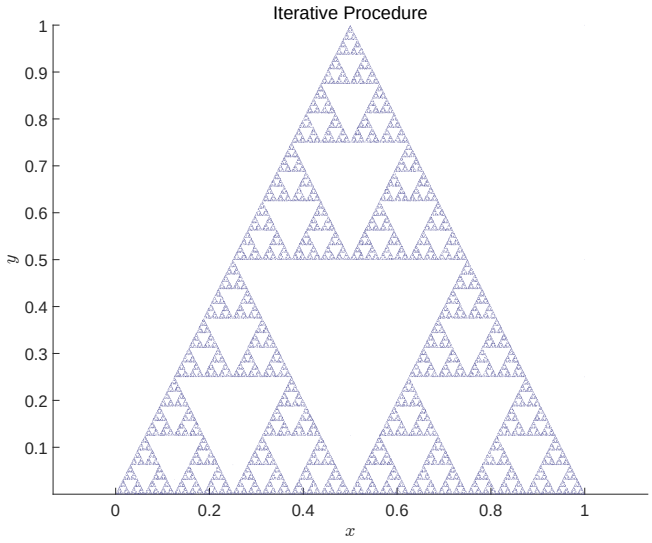

Figure 3: Problem 4

```
clc; clear; close all;

% Define and Initialize Parameters
N = 10^5;
x = zeros(N, 1);
y = zeros(N, 1);
x(1) = 1;
```

```
x(1) = 1;
y(1) = 1;
m = [
    0.5, 0, 0, 0.5, 0,    0;
    0.5, 0, 0, 0.5, 0.5,  0;
    0.5, 0, 0, 0.5, 0.25, 0.5
];

% Define Probability and Randomize Procedure
prob = [1/3, 1/3, 1/3];
r = rand(N, 1);                     % Generate N Random Numbers
                                    % within [0, 1)
cum_prob = cumsum(prob);            % [1/3, 2/3, 1]
idx = sum(r ≥ cum_prob, 2) + 1;     % Add the Numbers by Columns

% Iterative Procedure
for n = 2:N
    k = idx(n);
    a = m(k, 1); b = m(k, 2); c = m(k, 3);
    d = m(k, 4); e = m(k, 5); f = m(k, 6);
    x(n) = a * x(n-1) + b * y(n-1) + e;
    y(n) = c * x(n-1) + d * y(n-1) + f;
end

% Plot the Figure
figure;
scatter(x, y, 1, '.', 'MarkerEdgeColor', [0.3 0.3 0.6]);
axis equal;
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$y$', 'Interpreter', 'latex');
title('Iterative Procedure');
set(gca, 'FontSize', 18);

% Save and Export the Figure
set(gcf, 'Position', [100, 100, 1600, 900]);
exportgraphics(gcf, 'Problem4.pdf', 'ContentType', 'vector');
```

Problem 5

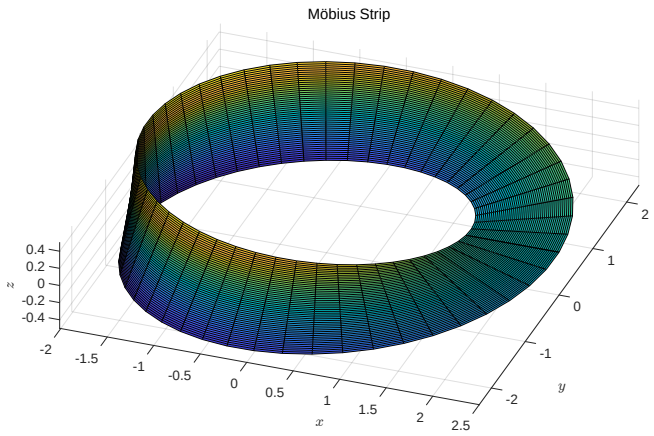

Figure 4: Problem 5(a)

```
clc; clear; close all;

% Define u, v and Generate Grid
u = linspace(-0.5, 0.5, 50);
v = linspace(0, 2*pi, 50);
[U, V] = meshgrid(u, v);

% Use Parameter Function
r = 2 + U .* cos(V/2);
x = r .* cos(V);
y = r .* sin(V);
z = U .* sin(V/2);

% Plot the Figure
surf(x, y, z);
axis equal;
view(20, 30);
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$y$', 'Interpreter', 'latex');
zlabel('$z$', 'Interpreter', 'latex');
title('Möbius Strip');
set(gca, 'FontSize', 18);

% Save and Export the Figure
set(gcf, 'Position', [100, 100, 1600, 900]);
exportgraphics(gcf, 'Problem5.pdf', 'ContentType', 'vector');
```

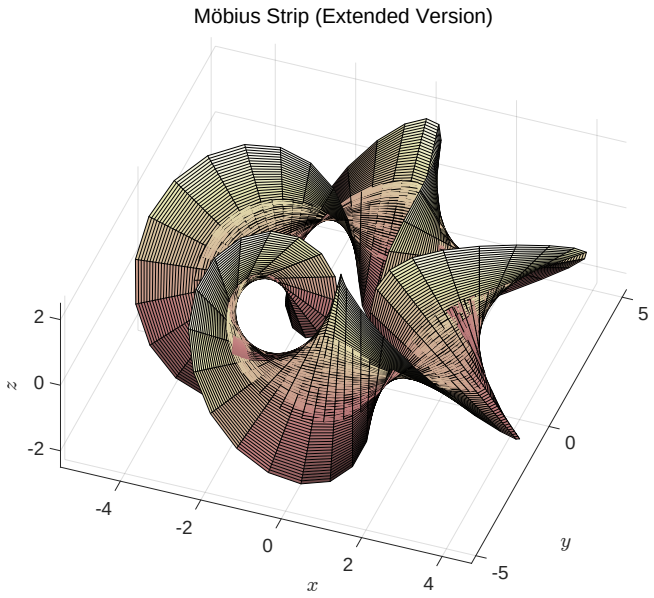

Figure 5: Problem 5(b)

```
clc; clear; close all;

% Define Parameters
theta = linspace(0, 4*pi, 100);
t = linspace(-1, 5/2, 50);
[Theta, T] = meshgrid(theta, t);

% Möbius Parameter: Count Loops
mob = 5/2;

% Initialize x, y, z
x0 = 3 + T .* sin(mob * Theta);
y0 = zeros(size(Theta));
z0 = T .* cos(mob * Theta);

% Rotate for theta Angle
X = x0 .* cos(Theta) - y0 .* sin(Theta);
Y = x0 .* sin(Theta) + y0 .* cos(Theta);
Z = z0;

% Plot the Figure
figure;
surf(X, Y, Z);
colormap('pink');
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$y$', 'Interpreter', 'latex');
zlabel('$z$', 'Interpreter', 'latex');
title('Möbius Strip (Extended Version)');
set(gca, 'FontSize', 18);
```

```
axis equal;
view(20, 40);

% Save and Export the Figure
set(gcf, 'Position', [100, 100, 1600, 900]);
exportgraphics(gcf, 'Problem5.2.pdf', 'ContentType', 'vector');
```

Problem 6

Click here for the video.

```
clc; clear; close all;

%------------ Initialize Obstacles ------------
% Outer Circle
R_outer = 10;
circle_outer = @(t) [R_outer*cos(t); R_outer*sin(t)];

% Inner Circle
obstacles = [
    6, 0, 3;   % x Coordinate, y Coordinate and Radius
   -4, -2, 1.2;
   -1, 4, 4.5];

%-------------Initialize Parameters------------
dt = 0.05;
T = 40;
t = 0:dt:T;

% Initialize Position and Velocity
pos = [-10, 0];
vel = [4, 2];

%-------------Start Recording ----------------
v = VideoWriter('Problem6.avi');
v.FrameRate = length(t) / T;
open(v);

%-------------Plot Obstacles-----------------
figure;
hold on; axis equal; grid on;
theta = linspace(0, 2*pi, 100);

% Plot Outer Circle
plot(R_outer*cos(theta), R_outer*sin(theta), 'k', 'LineWidth', 1);

% Plot Inner Circle
for i = 1:size(obstacles, 1)
    cx = obstacles(i,1);
    cy = obstacles(i,2);
```

```
    r = obstacles(i,3);
    plot(cx + r*cos(theta), cy + r*sin(theta), 'r', 'LineWidth', 1);
end

%--------------- Simulate Movement ---------------
ball = scatter(pos(1), pos(2), 50, 'b', 'filled');
trajectory = animatedline('Color', 'b', 'LineWidth', 0.55);
pause(1);

for i = 1:length(t)
    % Update Position Before Hitting
    new_pos = pos + vel * dt;

    % Hit the Outer Circle
    if norm(new_pos) >= R_outer % Center of Out Circle is (0, 0)
        vel = vel - 2 * dot(vel, new_pos/norm(new_pos)) * ...
            (new_pos/norm(new_pos));
    end

    % Hit the Inner Circle
    for j = 1:size(obstacles, 1)
        cx = obstacles(j,1);
        cy = obstacles(j,2);
        r = obstacles(j,3);
        if norm(new_pos - [cx, cy]) <= r
            % Calculate Unit Normal Vector
            normal = (new_pos - [cx, cy]) / norm(new_pos - [cx, ...
                cy]);
            % Substract 2 * Projection on Norm Vector
            vel = vel - 2 * dot(vel, normal) * normal;
        end
    end

    % Update Position and Trajectory
    pos = pos + vel * dt;
    addpoints(trajectory, pos(1), pos(2));
    set(ball, 'XData', pos(1), 'YData', pos(2));

    % Save the Frame
    frame = getframe(gcf);
    writeVideo(v, frame);
    pause(0.01);
end

close(v);
```