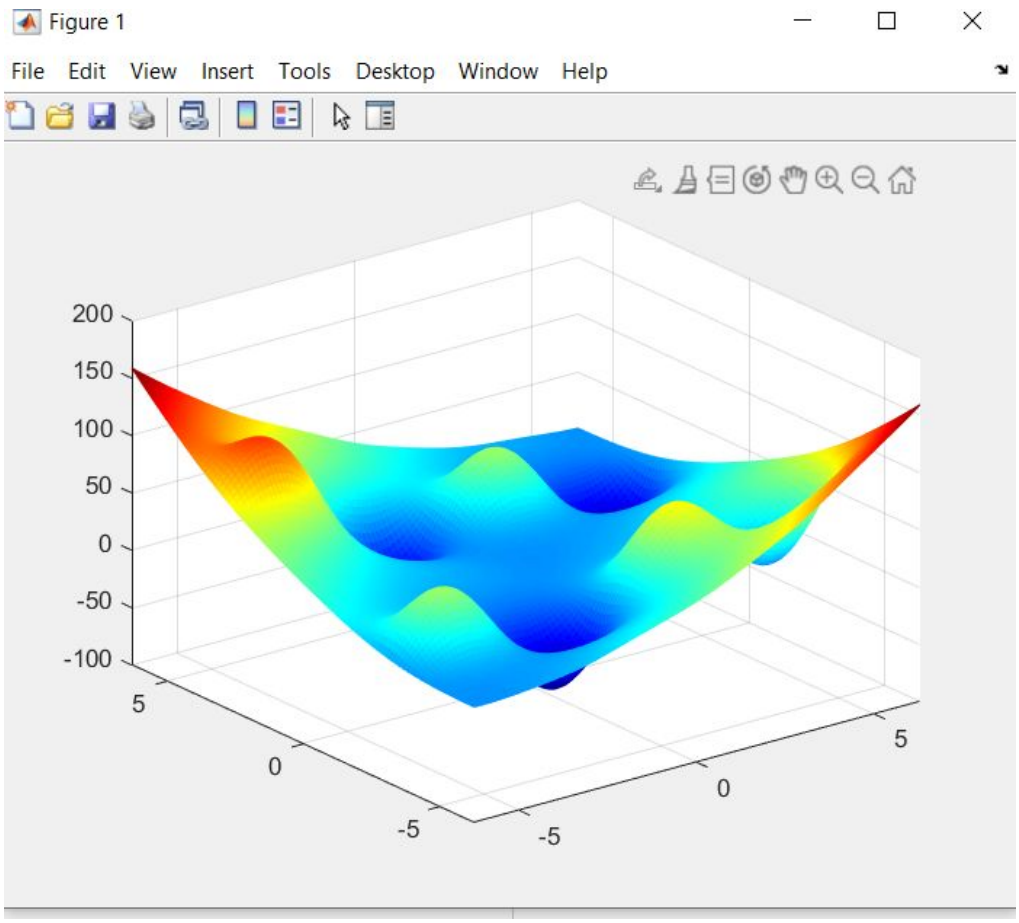
```
% Andrew Garwood
% Written HW 5
clc; clear all; close all;
% Problem 1

% 1a)
% We cannot use Newton's search because the operation fprime(k) / fdprime(k
% + 1) is not defined.
% for n = 114, section search takes 0.5341 seconds, while golden takes
% 0.3777. For n = 1000, section search takes 75.2134 seconds, while golden
% takes 56.9890 seconds. This is a significant improvement, and if it we
% increased n we would see that more clearly.

A114 = construct_A_n(114);
D = diag(diag(A114));
L = tril(A114) - D;
U = triu(A114) - D;
tolerance = 1e-8;

f = @(omega)(get_max_lambda(omega, D, L, U));

% section search 114
% a = 1;
% b = 2;
% c = .5001;
% tic
% x_star = section_search(f, a, b, c, tolerance);
% reg_section_time = toc; % = 0.5341
%
% % golden section search 114
% c = (-1 + sqrt(5)) / 2;
% tic
% [calls, x_star] = golden_section_search(f, a, b, c, tolerance);
% gold_section_time = toc; % 0.3777

% section search A1000
A1000 = construct_A_n(1000);
D = diag(diag(A1000));
L = tril(A1000) - D;
U = triu(A1000) - D;
f = @(omega)(get_max_lambda(omega, D, L, U));

% a = 1;
% b = 2;
```

```
% c = .5001;
% tic
% x_star = section_search(f, a, b, c, tolerance);
% reg_section_time = toc; % = 75.2134 seconds

% golden section search 1000
% c = (-1 + sqrt(5)) / 2;
% tic
% [calls, x_star] = golden_section_search(f, a, b, c, tolerance);
% gold_section_time = toc; % = 56.989 seconds

% 1b
f = @(x)(sin(tan(x)) - tan(sin(x)));
format long;
tolerance = 1e-16;
a = 1.5646;
b = 1.5647;
c = (-1 + sqrt(5)) / 2;
% tic
% [calls, x_star] = golden_section_search(f, a, b, c, tolerance);
% q_1b = toc
% x_star
% Newton Method bad. Please killl me



% Problem 2
% a)
f = @(x, y)(sin(x) .* exp(1 - cos(y)).^2 + cos(y) .* (1 - sin(x)).^2 + (x - y).^2);
% f = @(v)(sin(v(1)) .* exp(1 - cos(v(2))).^2 + cos(v(2)) .* (1 - sin(v(1))).^2 + (v(1) - v(2)).^2);

% b
x = linspace(-2 * pi, 2 * pi, 100);
y = linspace(-2 * pi, 2 * pi, 100);
[X, Y] = meshgrid(x, y);

% c
Z = f(X, Y);

% d
contour(X, Y, Z);

% e
% contour(X, Y, Z, levels); levels = vect z values, increasing order
```

```
% or levels is whole number = num of levels
contour(X, Y, Z, 10);

% f
% colorolroor lololo color of level curves
colormap('jet');

% g
% add label to each level curve to show corespodonaf;sldkfjasldkfj z alue
contour(X, Y, Z, 'Showtext', 'on');

%h
% plot plots on top of contour plottttttt just call plot

x0 = [0, 0];
min = fminsearch( @(v)(f ( v(1), v(2) )), x0);
% plot());
zmin = f(min(1), min(2));
hold on
plot3(min(1), min(2), zmin, 'ko')
contour(X, Y, Z, 'Showtext', 'on');
hold off
% j(
surf(X, Y, Z)

%k
surf(X, Y, Z, 'EdgeColor', 'interp');

% m
% view(az, el)
```




```
function max_lambda = get_max_lambda(omega, D, L, U)
    P = (1 / omega) * D + L;
    T = ((omega - 1) / omega) * D + U;
    M = -P\T;
    max_lambda = max(abs(eig(M)));
end


function A_n = construct_A_n(n)
    A_n = zeros(n, n);

    v2 = zeros(1, n);
    v2(1:end) = 2;

    v_neg1 = zeros(1, n - 1);
    for i = 1:(n - 1)
        v_neg1(i) = -1;
    end
```

```
    diag_2 = diag(v2);

    diagPlusOne = diag(v_neg1, 1);
    diagMinusOne = diag(v_neg1, -1);

    A_n = A_n + diagMinusOne + diagPlusOne + diag_2;
end


% regular section search
function x_star = section_search(f, a, b, c, tolerance)
    x = c * a + (1 - c) * b;
    y = (1 - c) * a + c * b;
    intolerable = true;
    while intolerable
        if f(x) < f(y)
            b = y;
        else
            a = x;
        end
        x = c * a + (1 - c) * b;
        y = (1 - c) * a + c * b;
        if abs(b - a) < tolerance
            x_star = b;
            intolerable = false;
        end
    end
end

% golden section search. .. .
function [f_calls, x_star] = golden_section_search(f, a, b, c, tolerance)
    x = c * a + (1 - c) * b;
    fx = f(x);
    y = (1 - c) * a + c * b;
    fy = f(y);
    f_calls = 2;
    intolerable = true;
    % recall a<x<y<b
    while intolerable
        if fx < fy
            % assign new bounds, throw away right side
            b = y;
            y = x;
```

```
            fy = fx;
            x = c * a + (1 - c) * b;
            fx = f(x);
            f_calls = f_calls + 1;
        else % fy <= fx
            % assign new bounds, throw away left side
            a = x;
            x = y;
            fx = fy;
            y = (1 - c) * a + c * b;
            fy = f(y);
            f_calls = f_calls + 1;
        end

        if abs(b - a) < tolerance % found x_star
            intolerable = false;
            x_star = b;
            % break;
        end
    end
end
```

I’m sorry. I can’t