Assignment 5

Table of Contents

Problem 1: 2D Optimization, Steepest Ascent

Performed all work by hand, determining optimal step size (i.e. the root h* of the function g'(h)) using bisection method, as written in Assignment 3

```
clc; close all; clear all;
upper_bound_error = 0.001;
i = 0;
approx_error_bi = [10000]; %placeholder since first approx. error = N/
A
gprime = @(h) -6561*h^3 - 512*h + 117;

% Bisection method, bracketed between 0 and 1
f_lower = gprime(0);
f_upper = gprime(1);

if f_lower == 0
    h = 0;
end
if f_upper == 0
    h = 1;
end
if f_upper*f_lower > 0
    fprintf('Cannot use bisection method if both brackets are the same
 sign.');
end
upper = 1;
lower = 0;
h = .5*(upper + lower);
f_root = gprime(h);

while approx_error_bi(end) > upper_bound_error
    i = i + 1;
    if f_lower*f_root == 0
        h = lower;
        break
    end
    if f_lower*f_root < 0
        upper = h;
    else
```

```
        lower = h;
    end
    f_lower = gprime(lower);
    f_upper = gprime(upper);
    oldroot = h;
    h = .5*(upper + lower);
    approx_error_bi = [approx_error_bi, abs((h-oldroot)/h*100)];
    f_root = gprime(h);
end
fprintf('Using the bisection method (after %d iterations), the optimal
 step size is %f\n', i, h);
```

```
Using the bisection method (after 19 iterations), the optimal step
 size is 0.167882
```

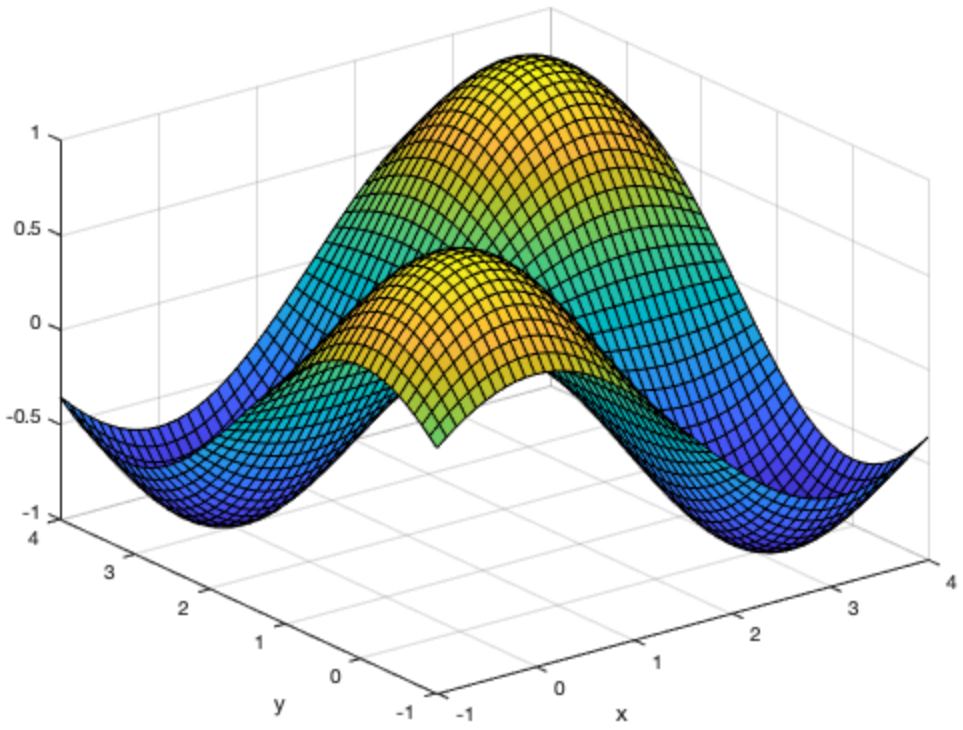
Problem 2: 2D Optimization, Newton's Method

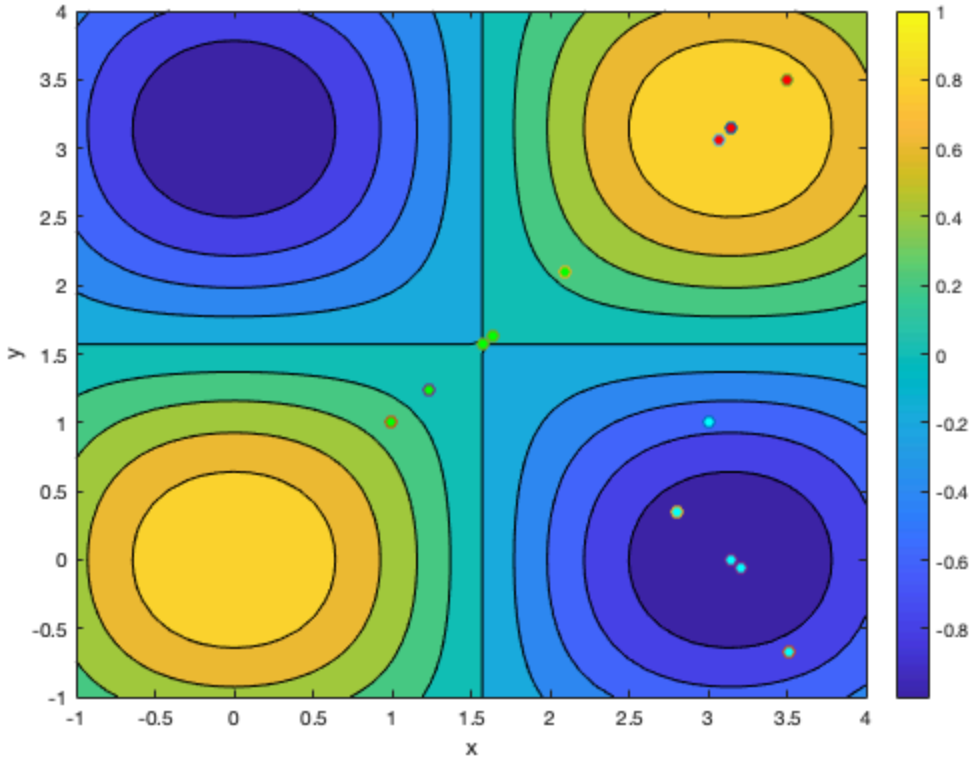
```
close all; clear all; clc;
[X,Y] = meshgrid([-1:0.1:4]);
fxy = cos(X).*cos(Y);

figure(1)
surf(X,Y,fxy)
xlabel('x')
ylabel('y')

figure(2)
contourf(X,Y,fxy)
colorbar;
xlabel('x')
ylabel('y')

f=@(x,y) cos(x)*cos(y);
grad_f=@(x,y) [-sin(x)*cos(y); -sin(y)*cos(x)]; % 2 X 1 gradient
 vector
Hessian_f=@(x,y) [-cos(x)*cos(y), sin(x)*sin(y); sin(x)*sin(y), -
cos(x)*cos(y)]; % 2 X 2 Hessian matrix

contourf(X,Y,fxy)
colorbar;
xlabel('x')
ylabel('y')
hold all

% Starting Guess #1 is in blue
x0= 3;
y0= 1;
z0=[x0;y0];
blue_sequence1 = [z0];
z_original = z0;
plot(z0(1),z0(2),'o','MarkerFaceColor','c')

% Newton’s method
```

```
for i=1:10
    z1 = z0 - Hessian_f(z0(1), z0(2))\grad_f(z0(1), z0(2));
    z0 = z1;
    blue_sequence1 = [blue_sequence1, z0];
    pause(.5)
    plot(z0(1),z0(2),'o','MarkerFaceColor','c')
end
fprintf('After 10 iterations, [%d, %d] converges to [%f, %f]\n',
 z_original(1), z_original(2), z1(1), z1(2));
fprintf('Using surface and contour plots, [%f, %f] is a relative
 minima.\n\n', z1(1), z1(2));


% Starting Guess #2 is in red
x0= 3.5;
y0= 3.5;
z0=[x0;y0];
red_sequence2 = [z0];
z_original = z0;
plot(z0(1),z0(2),'o','MarkerFaceColor','r')

% Newton's method
for i=1:10
    z1 = z0 - Hessian_f(z0(1), z0(2))\grad_f(z0(1), z0(2));
    z0 = z1;
    red_sequence2 = [red_sequence2, z0];
    pause(.5)
    plot(z0(1),z0(2),'o','MarkerFaceColor','r')
end
fprintf('After 10 iterations, [%d, %d] converges to [%f, %f]\n',
 z_original(1), z_original(2), z1(1), z1(2));
fprintf('Using surface and contour plots, [%f, %f] is a relative
 maxima.\n\n', z1(1), z1(2));

% Starting Guess #3 is in green
x0= 1;
y0= 1;
z0=[x0;y0];
green_sequence3 = [z0];
z_original = z0;
plot(z0(1),z0(2),'o','MarkerFaceColor','g')

% Newton's method
for i=1:10
    z1 = z0 - Hessian_f(z0(1), z0(2))\grad_f(z0(1), z0(2));
    z0 = z1;
    green_sequence3 = [green_sequence3, z0];
    pause(.5)
    plot(z0(1),z0(2),'o','MarkerFaceColor','g')
end
fprintf('After 10 iterations, [%d, %d] converges to [%f, %f]\n',
 z_original(1), z_original(2), z1(1), z1(2));
fprintf('Using surface and contour plots, [%f, %f] is a saddle point.
\n\n', z1(1), z1(2));
```

```
% To see sequence of estimates for each guess, see variables named
% color_sequence(number).
```

```
After 10 iterations, [3, 1] converges to [3.141593, 0.000000]
Using surface and contour plots, [3.141593, 0.000000] is a relative
 minima.

After 10 iterations, [3.500000e+00, 3.500000e+00] converges to
 [3.141593, 3.141593]
Using surface and contour plots, [3.141593, 3.141593] is a relative
 maxima.

After 10 iterations, [1, 1] converges to [1.570796, 1.570796]
Using surface and contour plots, [1.570796, 1.570796] is a saddle
 point.
```





Problem 3: Constrained Optimization

See attached screenshot and work on paper

Problem 4: Interpolation with Newton's Method

```
close all; clear all; clc;
% Part a
x = 2.8;
fprintf('Part a: Estimating 2.8 using Newton FDD\n');
% First order
x0 = 2.5; x1 = 3.2; fx0 = 14; fx1 = 15;
fdd1 = (fx1 - fx0)/(x1 - x0);
nm1 = fx0 + fdd1*(x-x0);
fprintf('First Order estimate of 2.8: %f\n', nm1);
% Second Order
x2 = 2.0; fx2 = 8;
fdd2 = ((fx2 - fx1)/(x2 - x1) - fdd1)/(x2 - x0);
nm2 = nm1 + fdd2*(x-x0)*(x-x1);
fprintf('Second Order estimate of 2.8: %f\n', nm2);
% Third order
x3 = 4.0; fx3 = 8;
fdd3 = (((fx3 - fx2)/(x3 - x2) - (fx2 - fx1)/(x2 - x1))/(x3 - x1) -
 fdd2)/(x3 - x0);
nm3 = nm2 + fdd3*(x-x0)*(x-x1)*(x-x2);
```

```
fprintf('Third Order estimate of 2.8: %f\n', nm3);

% Part b
fprintf('\nPart b: Estimating error R for each order.\n');
R1 = abs((nm2 - nm1)/nm2 * 100);
fprintf('The error for first order approx is: %f%%\n', R1);
R2 = abs((nm3 - nm2)/nm3 * 100);
fprintf('The error for second order approx is: %f%%\n', R2);

% To find third order error, need to find fourth order approximation
 first.
x4 = 1.6; fx4 = 2.0;
fdd4 = ((((fx4 - fx3)/(x4 - x3) - (fx3 - fx2)/(x3 - x2))/(x4 - x2) -
 ((fx3 - fx2)/(x3 - x2) - (fx2 - fx1)/(x2 - x1))/(x3 - x1)) - fdd3)/
(x4 - x1);
nm4 = nm3 + fdd4*(x-x0)*(x-x1)*(x-x2)*(x-x3);
R3 = abs((nm4 - nm3)/nm4 * 100);
fprintf('The error for third order approx is: %f%%\n', R3);
```

```
Part a: Estimating 2.8 using Newton FDD
First Order estimate of 2.8: 14.428571
Second Order estimate of 2.8: 15.485714
Third Order estimate of 2.8: 15.388571

Part b: Estimating error R for each order.
The error for first order approx is: 6.826568%
The error for second order approx is: 0.631266%
The error for third order approx is: 0.013927%
```

Published with MATLAB® R2020a