Ayush Gaggar HOMEWORK 5 11/12/20

1. $f(x,y) = 4.5x + 3y + x^2 - x^4 - 5xy - 2y^2$; $x=0,\ y=0$

$\frac{\partial f}{\partial x} = 4.5 + 2x - 4x^3 - 5y \Rightarrow \left.\frac{\partial f}{\partial x}\right|_{\substack{y=0\\x=0}} = 4.5$

$\frac{\partial f}{\partial y} = 3 - 5x - 4y \Rightarrow \left.\frac{\partial f}{\partial y}\right|_{\substack{y=0\\x=0}} = 3$. Expressing $f(x,y)$ as a function of h

$$g(h) = f(0 + 4.5h,\ 0 + 3h)$$
$$= \frac{81}{4}h + 9h + \frac{81}{4}h^2 - (4.5)^4 h^4 - 15h\left(\frac{9}{2}h\right) - 2(9h^2)$$
$$= \frac{117}{4}h + \frac{81}{4}h^2 - \frac{135}{2}h^2 - \frac{72}{4}h^2 - \left(\frac{9}{2}\right)^4 h^4$$
$$= -\left(\frac{9}{2}\right)^4 h^4 - \frac{261}{4}h^2 + \frac{117}{4}h$$
$$g'(h^*) = 0 = -\frac{9^4}{4}h^3 - \frac{261}{2}h + \frac{117}{4} = 0$$
$$-6561h^3 - 512h + 117 = 0$$

Bisection method: Between $h=0$ & $h=1/2$

$g'(0.25) = -6561\left(\frac{1}{4}\right)^3 - 512\left(\frac{1}{4}\right) + 117 = -113.5156 < 0$

$g'(1/8) = -6561\left(\frac{1}{8}\right)^3 - 512\left(\frac{1}{8}\right) + 117 = 40.1855 > 0$

$g'\left(\frac{1}{2}\left(\frac{1}{8} + \frac{1}{4}\right)\right) = -6561\left(\frac{3}{16}\right)^3 - 512\left(\frac{3}{16}\right) + 117 = -22.2488 < 0$

$g'(5/32) = 11.9718 > 0$

$g'\left(\frac{1}{2}\left(\frac{5}{32} + \frac{3}{16}\right)\right) = -4.3126$

$\rightarrow$ After some iterations by hand, see exact root h^* (optimal step size) in Matlab

2. $f(x) = \cos(x)\cos(y)$

$\frac{\partial f}{\partial x} = -\sin(x)\cos(y)$; $\frac{\partial f}{\partial y} = -\sin(y)\cos(x)$

$\frac{\partial^2 f}{\partial x^2} = -\cos(x)\cos(y)$; $\frac{\partial^2 f}{\partial y^2} = -\cos(y)\cos(x)$

$\frac{\partial^2 f}{\partial x \partial y} = \sin(x)\sin(y)$ $\quad$ $\frac{\partial^2 f}{\partial y \partial x} = \sin(y)\sin(x)$

$$H = \begin{bmatrix} -\cos(x)\cos(y) & \sin(x)\sin(y) \\ \sin(x)\sin(y) & -\cos(y)\cos(x) \end{bmatrix}$$

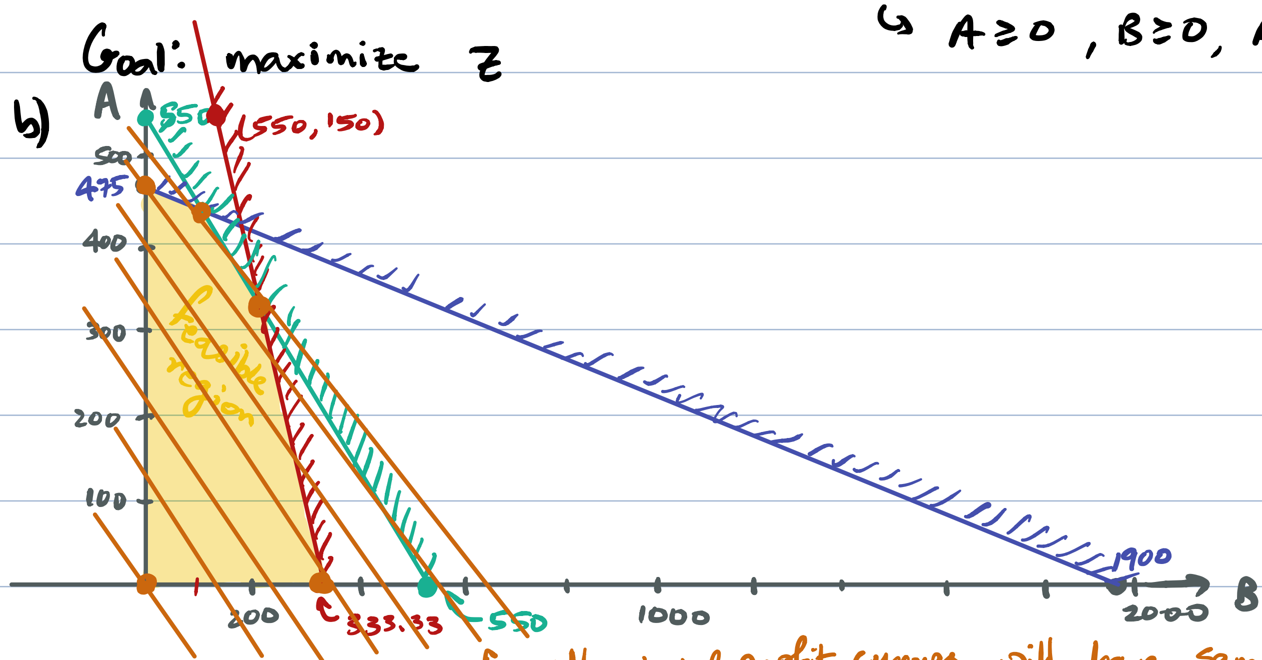
3. a) $20A + 5B \le 9500$ ← plastic $A + B \le 550$ ← storage

$0.04A + 0.12B \le 40$ ← hours $45A + 20B = z =$ Profit

↳ $A \ge 0$, $B \ge 0$, $A, B \in \mathbb{Z}^+$

Goal: maximize z

b)



↳ all potential profit curves will have same slope

To maximize z, we just need to check the 5 vertices of the feasible region.

1) $z(0,0) \rightarrow z = 0 \rightarrow$ Trivial

2) $z(0,475) \rightarrow z = 45(475) + 0 = \$21,375$

3) $z(333\frac{1}{3}, 0) \rightarrow z = 0 + 20(333\frac{1}{3}) = \$6666.67$

4) Intersection of blue & green:

$20A + 5B = 9500$

$A + B = 550 \Rightarrow 5B = 2750 - 5A$

$15A + 2750 = 9500 \rightarrow A = 450 \rightarrow B = 100$

$z(450, 100) \rightarrow z = 45(450) + 20(100) = \$22,250$

5) Intersection of red & green:

$4A + 12B = 4000$

$A + B = 550 \rightarrow 4A = 2200 - 4B$

$8B + 2200 = 4000 \rightarrow B = 225 \rightarrow A = 325$

$z(225, 225) \rightarrow z = 45(325) + 20(225) = \$19,125$

Maximum profit is \$22,250, when making 450 units of A & 100 units of B.

c) See Excel screenshots

d) The point (450, 100) lies on the intersection of the blue (plastic) and green (storage), so changing production time will not affect the max profit. Playing around with the Simplex LP solver, profit increases more when increasing raw material constraint as compared to increasing more storage

4. a) Use points closest & centered around $x = 2.8$

First Order: $f_1(x) = f(x_0) + \left[\frac{f(x_1) - f(x_0)}{x_1 - x_0}\right](x - x_0)$

Use points $x_0 = 2.5$, $f(x_0) = 14$ & $x_1 = 3.2$, $f(x_1) = 15$ to estimate $f(2.8)$

$$f_1(2.8) = 14 + \left(\frac{15-14}{3.2-2.5}\right)(2.8-2.5) = 14.4286$$

Second Order:

$$f_2(x) = f(x_0) + \left(\frac{f(x_1) - f(x_0)}{x_1 - x_0}\right)(x - x_0) + \left(\frac{\frac{f(x_2)-f(x_1)}{x_2 - x_1} - \frac{f(x_1)-f(x_0)}{x_1 - x_0}}{x_2 - x_0}\right) \cdot (x-x_0)(x-x_1)$$

$x_0 = 2.5$, $x_1 = 3.2$, $x_2 = 2.0$; $f(x_0) = 14$, $f(x_1) = 15$, $f(x_2) = 8$

$$f_2(2.8) = 14 + \left(\frac{15-14}{3.2-2.5}\right)(2.8-2.5) + \left(\frac{\frac{8-15}{2.0-3.2} - \frac{15-14}{3.2-2.5}}{2.0 - 2.5}\right)(2.8-2.5)(2.8-3.2)$$

$= 15.4857$

Third order

$x_0 = 2.5$, $[f_0] = 14$

$> [f_0, f_1] = \frac{10}{7}$

$x_1 = 3.2, [f_1] = 15$ $\quad > [f_0, f_1, f_2] = \frac{-305}{21}$

$> [f_1, f_2] = \frac{35}{6}$

$x_2 = 2.0, [f_2] = 8$ $\quad > [f_1, f_2, f_3] = \frac{-175}{24}$

$> [f_2, f_3] = 0$

$x_3 = 4.0, [f_3] = 8$

See the rest of the work in MATLAB

b) $R_n = \frac{f_{n+1}(x) - f_n(x)}{f_{n+1}(x)}$. See answers in MATLAB

MAE 105 Homework 5, Problem 3c) i.

	A	B			
Number to Make	450	100			
Required Inputs			Total		Constraint
Plastic	20	5	9500	<=	9500
Total Hours	0.04	0.12	30	<=	40
Storage	1	1	550	<=	550
Profit/Unit ($)	45	20	$22,250.00		

MAE 105 Homework 5, Problem 3c) ii.

	A	B			
Number to Make	450	100			
Required Inputs			Total		Constraint
Plastic	20	5	=SUMPRODUCT($B$4:$C$4,B7:C7)	<=	9500
Total Hours	0.04	0.12	=SUMPRODUCT($B$4:$C$4,B8:C8)	<=	40
Storage	1	1	=SUMPRODUCT($B$4:$C$4,B9:C9)	<=	550
Profit/Unit ($)	45	20	=SUMPRODUCT($B$4:$C$4,B14:C14)		

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

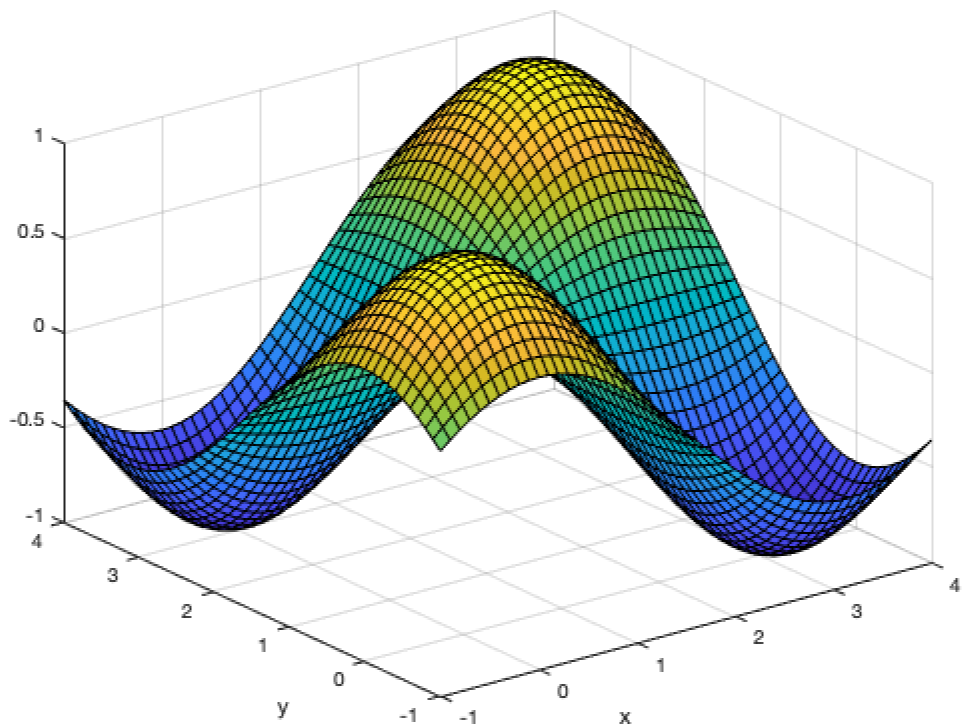
Problem 2: 2D Optimization, Newton's Method

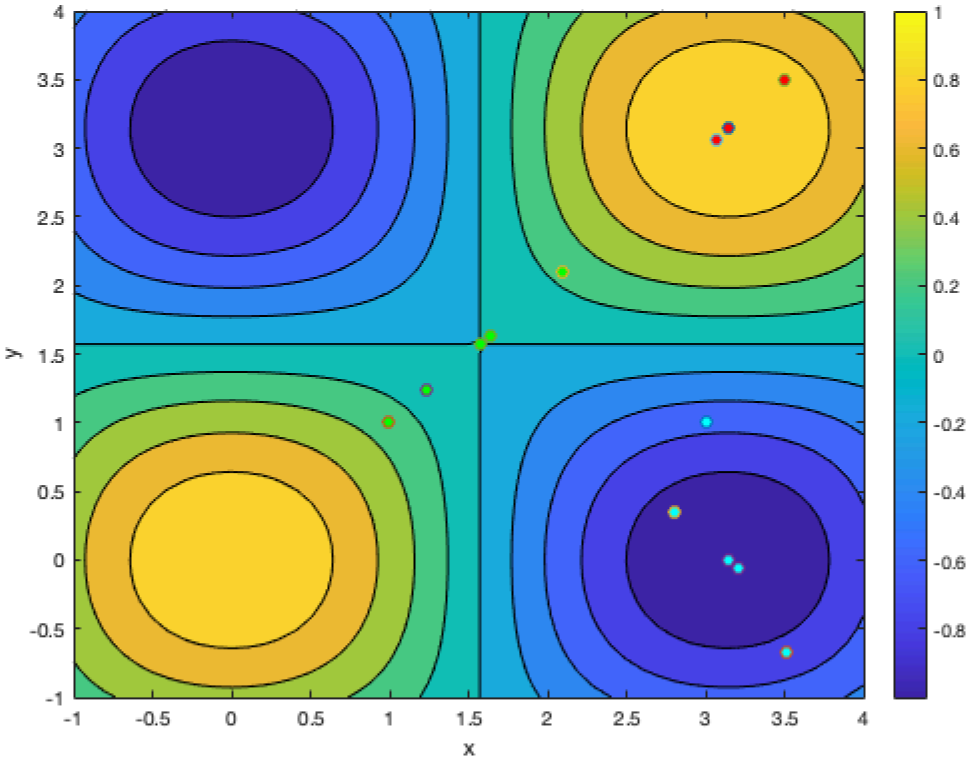
```
close all; clear all; clc;
[X,Y] = meshgrid([-1:0.1:4]);
fxy = cos(X).*cos(Y);

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