

Brett Levenson, Andy Poulos, Rishabh Shah, John Stefan
October 31, 2017
Lab 3

Exercise One

1) $f(x_1, x_2, x_3, x_4) = x_1^2 + x_2^2 + x_3^2 + x_4^2$ $g(x_1, x_2, x_3, x_4) = x_1 + x_2 + x_3 + x_4 = 20$
 $\nabla f(x_1, x_2, x_3, x_4) = \langle 2x_1, 2x_2, 2x_3, 2x_4 \rangle$ $\nabla g(x_1, x_2, x_3, x_4) = \langle 1, 1, 1, 1 \rangle$
 $2x_1 = \lambda$
 $2x_2 = \lambda$ $x_1 = x_2 = x_3 = x_4$ $x_1 + x_1 + x_1 + x_1 = 20$ $4x_1 = 20$ $x_1 = 5$
 $2x_3 = \lambda$ $x_1 = x_2 = x_3 = x_4 = 5$
 $2x_4 = \lambda$
 $f(5, 5, 5, 5) = 5^2 + 5^2 + 5^2 + 5^2 = 100$

Exercise Two

```
Command Window
New to MATLAB? See resources for Getting Started.

>> syms g(x,y)
>> syms f(x,y,z)
>> g = 2-x^2 - 0.5*y^2;
>> f = x^2 + y^2 + z^2;
>> gF = gradient(f);
>> gG = gradient(g);
>> sol = vpasolve([gF(1) == gG(1)*l, gF(2) == gG(2)*l, gF(3) == gG(3)*l, z==2-x^2-0.5*y^2], [x, y, z, l])
Undefined function or variable 'l'.

>> syms l
>> sol = vpasolve([gF(1) == gG(1)*l, gF(2) == gG(2)*l, gF(3) == gG(3)*l, z==2-x^2-0.5*y^2], [x, y, z, l])
Index exceeds matrix dimensions.

Error in sym/subsref (line 881)
    R_tilde = builtin('subsref',L_tilde,Idx);

>> sol = vpasolve([gF(0,1) == gG(0,1)*l, gF(0,2) == gG(0,2)*l, gF(0,3) == gG(0,3)*l, z==2-x^2-0.5*y^2], [x, y, z, l])
Subscript indices must either be real positive integers or logicals.

Error in sym/subsref (line 881)
    R_tilde = builtin('subsref',L_tilde,Idx);

>> sol = vpasolve([gF(1) == gG(1)*l, gF(2) == gG(2)*l, gF(3) == gG(3)*l, z==2-x^2-0.5*y^2], [x, y, z, l])
Index exceeds matrix dimensions.

Error in sym/subsref (line 881)
    R_tilde = builtin('subsref',L_tilde,Idx);

>> gF

gF =

    2*x
    2*y
    2*z

>> sol = vpasolve([gF(1,0) == gG(1,0)*l, gF(2,0) == gG(2,0)*l, gF(3,0) == gG(3,0)*l, z==2-x^2-0.5*y^2], [x, y, z, l])
Subscript indices must either be real positive integers or logicals.

Error in sym/subsref (line 881)
    R_tilde = builtin('subsref',L_tilde,Idx);

>> sol = vpasolve([gF(1) == gG(1)*l, gF(2) == gG(2)*l, gF(3) == 0*l, z==2-x^2-0.5*y^2], [x, y, z, l])

sol =

    struct with fields:
        x: [4x1 sym]
```

```
Command Window
New to MATLAB? See resources for Getting Started.

y: [4x1 sym]
z: [4x1 sym]
l: [4x1 sym]

>> x
x =
x

>> sol.x
ans =
0
0
-1.4142135623730950488016887242097
1.4142135623730950488016887242097

>> sol
sol =
struct with fields:
    x: [4x1 sym]
    y: [4x1 sym]
    z: [4x1 sym]
    l: [4x1 sym]

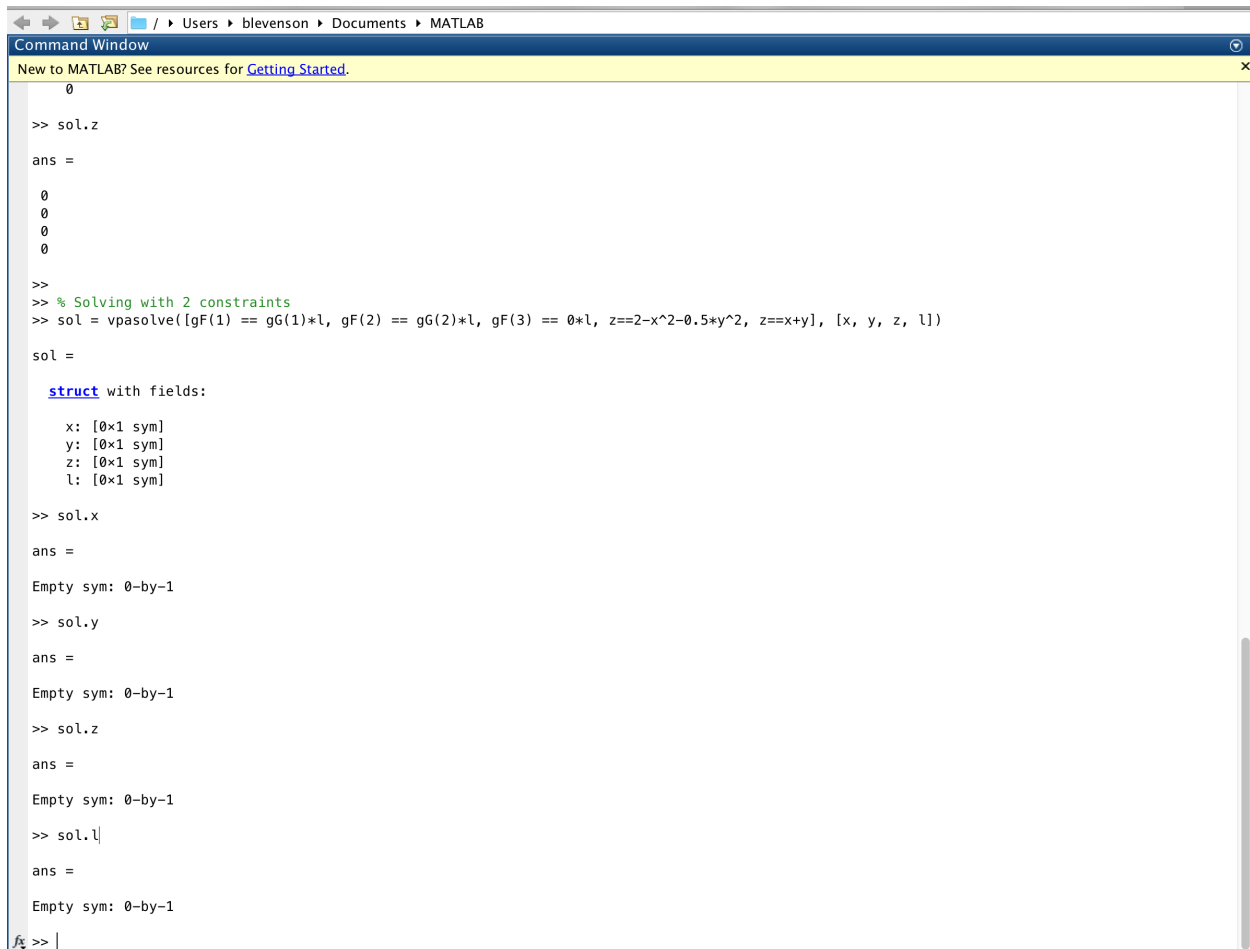
>> sol.y
ans =
-2.0
2.0
0
0

>> sol.z
ans =
0
0
0
0

fx >>
```

The solutions that work are: $(0, -2, 0)$, $(0, 0, 0)$, $(-1.4142, -2, 0)$, $(-1.4142, 0, 0)$, $(1.4142, -2, 0)$.

Exercise Three

A screenshot of the MATLAB Command Window. The title bar shows the path 'Users > blevenson > Documents > MATLAB'. The window contains a yellow banner with the text 'New to MATLAB? See resources for [Getting Started](#).' Below this, the command history shows the following: a '0' is entered; the command '>> sol.z' is entered, resulting in 'ans = 0'; the command '>> % Solving with 2 constraints' is entered; the command '>> sol = vpasolve([gF(1) == gG(1)*l, gF(2) == gG(2)*l, gF(3) == 0*l, z==2-x^2-0.5*y^2, z==x+y], [x, y, z, l])' is entered, resulting in a 'struct' with fields: x: [0x1 sym], y: [0x1 sym], z: [0x1 sym], l: [0x1 sym]; the command '>> sol.x' is entered, resulting in 'ans = Empty sym: 0-by-1'; the command '>> sol.y' is entered, resulting in 'ans = Empty sym: 0-by-1'; the command '>> sol.z' is entered, resulting in 'ans = Empty sym: 0-by-1'; the command '>> sol.l' is entered, resulting in 'ans = Empty sym: 0-by-1'. The prompt 'f3 >> |' is visible at the bottom left.

```
0
>> sol.z
ans =
0
0
0
0
>>
>> % Solving with 2 constraints
>> sol = vpasolve([gF(1) == gG(1)*l, gF(2) == gG(2)*l, gF(3) == 0*l, z==2-x^2-0.5*y^2, z==x+y], [x, y, z, l])
sol =
    struct with fields:
        x: [0x1 sym]
        y: [0x1 sym]
        z: [0x1 sym]
        l: [0x1 sym]
>> sol.x
ans =
Empty sym: 0-by-1
>> sol.y
ans =
Empty sym: 0-by-1
>> sol.z
ans =
Empty sym: 0-by-1
>> sol.l
ans =
Empty sym: 0-by-1
f3 >> |
```

When solving the system with two constraints, there are no critical points of $f(x, y, z)$ constrained by the boundary. This is because the space has been overrefined, preventing any points from fulfilling both the constraints and the LaGrange multiplier equations. Because there are no points in the boundary, we are unable to determine if the gradients are linearly independent at every point on the boundary.

Exercise Four

```
>> syms g(x,y)
>> syms f(x,y,z)
>> g = 2-x^2 - 0.5*y^2;
>> f = x^2 + y^2 + z^2;
>> gF = gradient(f);
>> gG = gradient(g);
>> syms l
>> sol = vpasolve([gF(1) == gG(1)*1, gF(2) ==gG(2)*1, gF(3) == 0*1, z==2-x^2-0.5*y^2], [x,y,z,l])

sol =

    struct with fields:

        x: [4×1 sym]
        y: [4×1 sym]
        z: [4×1 sym]
        l: [4×1 sym]

>> f = @(x,y,z) (x^2) + (y^2) + (z^2);
>> f(sol.x(1), sol.y(1), sol.z(1))

ans =

4.0

>> f(sol.x(2), sol.y(2), sol.z(2))

ans =

4.0

>> f = @(x,y,z) (x^2) + (y^2) + (z^2)

f =

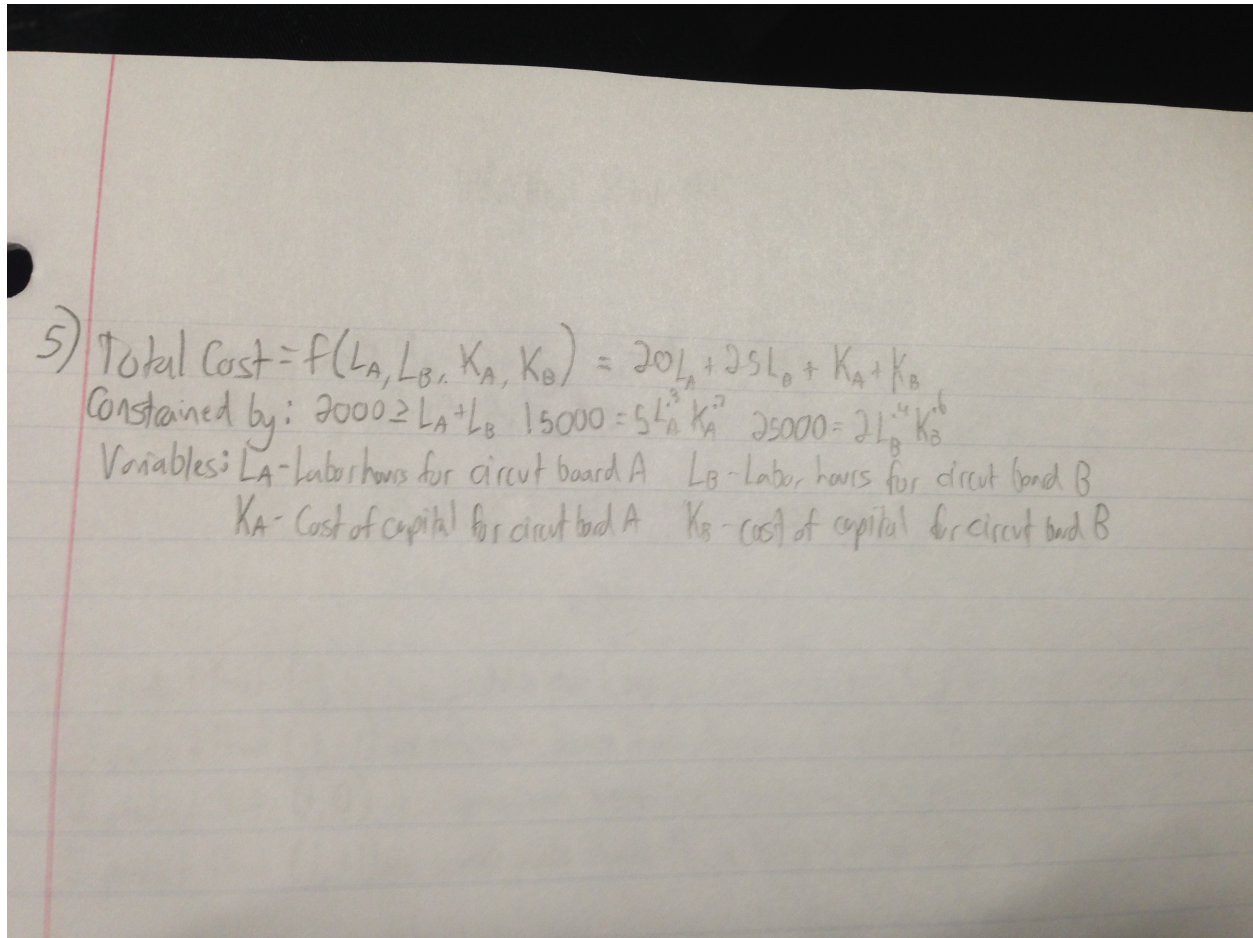
    function_handle with value:

    @(x,y,z) (x^2)+(y^2)+(z^2)

>> |
```

The maximum value is 4.0 from the point (0,0,2) and the minimum is 1.75 from the point (-1.2247,0,0.5).

Exercise Five



Exercise Six

```
>> syms La Kb Lb Kb lam mu
>> sol=vpasolve([15000==5*La^3*Ka^2, 25000==2*Lb^4*Kb^6, 20==lam*1.5*La^(-0.7)*Ka^0.7, 25==mu*0.8*Lb^(-0.6)*Kb^0.6, 1==lam*3.5*La^0.3*Ka^(-0.3), 1==mu*1.2*Lb^0.4*Kb^(-0.4)], [Ka, Kb, La, lam, Lb, mu]);
>> sol.La;
>> sol.Lb;
>> sol.Ka;
>> sol.Kb;
>> Cost=@(La, Lb, Ka, Kb) 20*La+25*Lb+Ka+Kb;
>> Cost(sol.La, sol.Lb, sol.Ka, sol.Kb)

ans =
102366.01881599831982906924326579
```

Running the function gave the critical point (203.62, 1420.7, 9502.2, 53275), returning a total cost of \$102,366.02.

Exercise Seven

```
>> pyms La Lb Ka Kb lam mu chi
>> sol=mpsolve([15000==5*La*0.3*Ka*0.7, 25000==2*Lb*0.4*Kb*0.6, 2000==La+Lb, 20==1.5*lam*La*(-0.7)*Ka*0.7+chi, 25==0.8*mu*Lb*(-0.6)*Kb*0.6+chi, 1==3.5*lam*La*0.3*Ka*(-0.3), 1==1.2*mu*Lb*0.4*Kb*(-0.4)], [chi, Ka, Kb, La, lam, Lb, mu]);
>> sol.La;
>> sol.Lb;
>> sol.Ka;
>> sol.Kb;
>> Cost=@(La, Lb, Ka, Kb) 20*La+25*Lb+Ka+Kb;
>> Cost(sol.La, sol.Lb, sol.Ka, sol.Kb);
>> Cost(sol.La, sol.Lb, sol.Ka, sol.Kb)

ans =

87630.783104032364276701023377223 - 3592.60570771971240871885162259461i
>> |
```

Running the function returns a critical point consisting of imaginary numbers, meaning there are no critical points on the boundary constraint $La + Lb = 2000$. Therefore the minimum cost is the cost found in Exercise 6, \$102,366.02.