CSI 747 – Midterm Submission

Jayasurya Kanukurthy jkanukur@gmu.edu G00863457

Solution 1:

The coefficients of the glued transformation was found by equating the value of the function at -0.5, the first derivative of the functions at -0.5 and the second derivative at -0.5.

$$at^2 + bt + c = \ln(1+t)\,, t = -0.5$$

$$2at + b = \frac{1}{1+t} = 2, at\ t = -0.5$$

$$2a = \frac{-1}{(1+t)^2} = -4\,, at\ t = -0.5$$

The values obtained after solving these for a, b and c were:

a = -2, b = 0 , c = ln(0.5)+0.5

Solution 2.a:

Non-linear Rescaling method with glued transformation implemented in Matlab for equation 1 is as follows:

Solution_2_a.m

```
function [ output_args ] = constraint( x )

    output_args = -x(1)^2-x(2)^2+25;

end


function [ output_args ] = Grad_PHI( x,y,k )
    x1 = x(1);
    x2 = x(2);
    grad_psik = psik(x,y,k);

    grad_phi1 = 2 + 2*x1*y*grad_psik(2);
    grad_phi2 = -3 + 2*x2*y*grad_psik(2);

    output_args = [grad_phi1;grad_phi2];

end
```

```matlab
function [ output_args ] = Hessian_PHI( x,y,k)


    psik_values = psik(x,y,k);

    output_args = [ 2*y*psik_values(2)-4*k*y*psik_values(3)*x(1)^2,    -
4*k*y*psik_values(3)*x(1)*x(2);
                    -4*k*y*psik_values(3)*x(1)*x(2),
2*psik_values(2)*y-4*k*y*psik_values(3)*x(2)^2];

end


function  [output_args] = PHI( x,y,k)
    x1 = x(1);
    x2 = x(2);
    psik_values = psik(x,y,k);
    output_args = 2*x1-3*x2-(1.0/k)*y*psik_values(1);
end


function [ output_args ] = psik( x,y,k )
%Return the function value, first differential
%second differential of glued transformation function
    psi = 0;
    psi_diff = 0;
    psi_2_diff = 0;

    a = -2;
    b =0;
    c = log(0.5)+0.5;

    t = k*constraint(x);

    if t > -0.5
        psi = log(1+t);
        psi_diff = 1.0/(1+t);
        psi_2_diff = -1.0/(1+t)^2;
    else
        psi = a*(t^2)+b*t+c;
        psi_diff = 2*a*t+b;
        psi_2_diff = 2*a;
    end

    output_args = [psi;psi_diff;psi_2_diff];

end
```

```
%Nonlinear Rescaling Method for Problem 2(a)
clear all; clc;

epsilon = 0.001;
k = 100;          %Change this later and see how the execution converges

x = [0.1 0.7]';
y = [1];

x_step = x;
y_step = y;

PHI_Values = PHI(x_step,y_step,k);

Grad = Grad_PHI(x_step,y_step,k);

while max([norm(Grad,2),norm(y_step*constraint(x_step),2),-
constraint(x_step)]) > 10^-2

    n_steps = 0;
    psi_values = psik(x_step,y_step,k);
    %Implementing Newton's Method
    eta = 0.3;
    while norm(Grad,2) > max([10^-7,(1.0/k)*norm(y_step-
psi_values(2)*y_step,2)]) %Check this for matrix dimensions!!!!
        %--------------Finding Direction----------------
        %Regularization of Hessian
        Hessian = Hessian_PHI(x_step,y_step,k);
   %     norm(Grad,2)
        %Regularize Hessian Here and return regularized Hessian
        lambda = 0.0001;

        [R,p] = chol(Hessian);      %Checking for positive definiteness using
cholsky factorization
        while p >0                  %p > 0 => not positive definite
            Hessian = Hessian + lambda*eye(length(Hessian));
            [R,p] = chol(Hessian);
            lambda = 10*lambda;
        end


        %Grad = Grad_PHI(x_step,y_step,k);

        %Finding Direction

        delta_xs = Hessian\-Grad;

        %Finding x_step: Armijio Rule

         alpha = 1;
         while PHI(x_step+alpha*delta_xs,y_step,k)-PHI(x_step,y_step,k) >=
eta * alpha * Grad'*delta_xs
              alpha = alpha/2;
```

```
         end
% %          %Updating x_step

        x_step = x_step+alpha*delta_xs;

        %Updating grad_PHI, psi_values

        PHI_Values = PHI(x_step,y_step,k);
        Grad = Grad_PHI(x_step,y_step,k);
        psi_values = psik(x_step,y_step,k);
        n_steps = n_steps+1;
    end

    %display number of Newtons iterations
    n_steps;

    %Updating y_step
    PHI_Values = PHI(x_step,y_step,k);
    Grad = Grad_PHI(x_step,y_step,k);
    psi_values = psik(x_step,y_step,k);
    psi_diff = psi_values(2);
    y_step = y_step * psi_diff;
    fprintf('Newton Iteration = % d; x1 = % f; x2 = % f; function = % f; y=%
f \n', n_steps, x_step(1), x_step(2), 2*x_step(1) - 3*x_step(2), y_step);

end

x_step;
y_step;
```

Output:

Newton Iteration = 9; x1 = -2.772529; x2 = 4.158774; function = -18.021382; y= 0.361311

Solution 2.b:

The Implementations is as follows:

Solution_2_b.m

```
function [ output_args ] = constraint( x )
%Returns the constraint

    output_args = -3*x(1)^2-x(2)^2+9;
```

```
end


function [ output_args ] = psik( x,y,k )
%Return the function value, first differential
%second differential of glued transformation function
    psi = 0;
    psi_diff = 0;
    psi_2_diff = 0;

    a = -2;
    b =0;
    c = log(0.5)+0.5;

    t = k*constraint(x);

    if t > -0.5
        psi = log(1+t);
        psi_diff = 1.0/(1+t);
        psi_2_diff = -1.0/(1+t)^2;
    else
        psi = a*(t^2)+b*t+c;
        psi_diff = 2*a*t+b;
        psi_2_diff = 2*a;
    end

    output_args = [psi;psi_diff;psi_2_diff];

end

function  [output_args] = PHI( x,y,k)
    x1 = x(1);
    x2 = x(2);
    psik_values = psik(x,y,k);
    output_args = x1^2+2*x1*x2+x2^2-(1.0/k)*y*psik_values(1);
end

function [ output_args ] = Grad_PHI( x,y,k )
    x1 = x(1);
    x2 = x(2);
    grad_psik = psik(x,y,k);

    grad_phi1 = 2*x1 + 6*x1*y*grad_psik(2)+2*x2;
    grad_phi2 = 2*x1+2*x2+2*x2*grad_psik(2)*y;

    output_args = [grad_phi1;grad_phi2];

end

function [ output_args ] = Hessian_PHI( x,y,k)

    psik_values = psik(x,y,k);
    x1 = x(1);
    x2 = x(2);
```

```
    output_args = [ 2+6*y*psik_values(2)-k*y*psik_values(3)*36*x1*x1,
2-k*y*psik_values(2)*12*x1*x2;
                    2-k*y*psik_values(2)*12*x1*x2,
2+2*y*psik_values(2)-k*y*psik_values(3)*4*x2*x2];

end
```

%Nonlinear Rescaling Method for Problem 2(b)

```
clear all; clc;

epsilon = 0.001;
k = 100;          %Change this later and see how the execution converges

x = [0.1 0.7]';
y = [1];

x_step = x;
y_step = y;

PHI_Values = PHI(x_step,y_step,k);

Grad = Grad_PHI(x_step,y_step,k);

while max([norm(Grad,2),norm(y_step*constraint(x_step),2),-
constraint(x_step)]) > 10^-4

    n_steps = 0;
    psi_values = psik(x_step,y_step,k);
    %Implementing Newton's Method
    eta = 0.3;
    while norm(Grad,2) > max([10^-7,(1.0/k)*norm(y_step-
psi_values(2)*y_step,2)]) %Check this for matrix dimensions!!!!
        %--------------Finding Direction----------------
        %Regularization of Hessian
        Hessian = Hessian_PHI(x_step,y_step,k);
   %     norm(Grad,2)
        %Regularize Hessian Here and return regularized Hessian
        lambda = 0.0001;

        [R,p] = chol(Hessian);      %Checking for positive definiteness using
cholsky factorization
        while p >0                  %p > 0 => not positive definite
            Hessian = Hessian + lambda*eye(length(Hessian));
            [R,p] = chol(Hessian);
            lambda = 10*lambda;
        end


        %Grad = Grad_PHI(x_step,y_step,k);

        %Finding Direction
```

```
        delta_xs = Hessian\-Grad;

        %Finding x_step: Armijio Rule

         alpha = 1;
         while PHI(x_step+alpha*delta_xs,y_step,k)-PHI(x_step,y_step,k) >=
eta * alpha * Grad'*delta_xs
              alpha = alpha/2;
         end
% %         %Updating x_step

        x_step = x_step+alpha*delta_xs;

        %Updating grad_PHI, psi_values

        PHI_Values = PHI(x_step,y_step,k);
        Grad = Grad_PHI(x_step,y_step,k);
        psi_values = psik(x_step,y_step,k);
        n_steps = n_steps+1;
    end

    %display number of Newtons iterations
    n_steps;

    %Updating y_step
    PHI_Values = PHI(x_step,y_step,k);
    Grad = Grad_PHI(x_step,y_step,k);
    psi_values = psik(x_step,y_step,k);
    psi_diff = psi_values(2);
    y_step = y_step * psi_diff;
    fprintf('Newton Iteration = % d; x1 = % f; x2 = % f; function = % f; y=%
f \n', n_steps, x_step(1), x_step(2),
x_step(1)^2+2*x_step(1)*x_step(2)+x_step(2)^2, y_step);
end

x_step;
y_step;
```

Output:

```
Newton Iteration =  3; x1 =  0.000228; x2 = -0.000229; function =
0.000000; y= 0.001110
Newton Iteration =  0; x1 =  0.000228; x2 = -0.000229; function =
0.000000; y= 0.000001
```

Solution 2.c:

The implementation is as follows:

Solution_2_c.m

```
function [ output_args ] = constraint( x )
%Returns the constraint
    x1 = x(1);
    x2 = x(2);
    x3 = x(3);
    x4 = x(4);
    output_args = [-x1^2-x2^2-x3^2-x4^2+4; x1+x2+2*x3+3*x4-1];

end

function [ output_args ] = psik( x,y,k,t )
%Return the function value, first differential
%second differential of glued transformation function
    psi = 0;
    psi_diff = 0;
    psi_2_diff = 0;

    a = -2;
    b =0;
    c = log(0.5)+0.5;

    con = constraint(x);
    t = k*con(t);
    if t > -0.5
        psi = log(1+t);
        psi_diff = 1.0/(1+t);
        psi_2_diff = -1.0/(1+t)^2;
    else
        psi = a*(t^2)+b*t+c;
        psi_diff = 2*a*t+b;
        psi_2_diff = 2*a;
    end

    output_args = [psi;psi_diff;psi_2_diff];

end

function  [output_args] = PHI( x,y,k)
    x1 = x(1);
    x2 = x(2);
    x3 = x(3);
    x4 = x(4);
    y1 = y(1);
    y2 = y(2);
    psik1 = psik(x,y,k,1);
    psik2 = psik(x,y,k,2);
```

```
    output_args = 3*x1^3+2*x2^3+x3^3+x4^3 - (1.0/k)*y1*psik1(1)-
(1.0/k)*y2*psik2(2);
end

function [ output_args ] = Grad_PHI( x,y,k )
    x1 = x(1);
    x2 = x(2);
    x3 = x(3);
    x4 = x(4);

    psik1 = psik(x,y,k,1);
    psik2 = psik(x,y,k,2);
    grad_phi = [9*x1^2;6*x2^2;3*x3^2;3*x4^2]-[-2*x1,1;-2*x2,1;-2*x3,2;-
2*x4,3]*([psik1(2),0;0,psik2(2)]*[y(1);y(2)]);

    output_args = grad_phi;

end


function [ Hessian ] = Hessian_PHI( x,y,k)

    x1 = x(1);
    x2 = x(2);
    x3 = x(3);
    x4 = x(4);
    y1 = y(1);
    y2 = y(2);

    psik1 = psik(x,y,k,1);
    psik2 = psik(x,y,k,2);

    hessf = [18*x1 0 0 0;0 12*x2 0 0;0 0 6*x3 0; 0 0 0 6*x4];

    Hessian = hessf - psik1(2)*y(1)*-2*eye(4) - k * [-2*x1,1;-2*x2,1;-
2*x3,2;-2*x4,3]*([y(1),0;0,y(2)]*[psik1(3),0;0,psik2(3)])*[-2*x1,1;-2*x2,1;-
2*x3,2;-2*x4,3]';
end
```

%Nonlinear Rescaling Method for Problem 2(c)

```
clear all; clc;

epsilon = 0.001;
k = 100;          %Change this later and see how the execution converges

x = [0.1 0.7 0.8 0.5]';
y = [1,2];

x_step = x;
y_step = y;

PHI_Values = PHI(x_step,y_step,k);
```

```
Grad = Grad_PHI(x_step,y_step,k);

while max([norm(Grad,2),norm(y_step*constraint(x_step),2),max(-
constraint(x_step))]) > 10^-2

    n_steps = 0;
    psik1 = psik(x_step,y_step,k,1);
    psik2 = psik(x_step,y_step,k,2);
    %Implementing Newton's Method
    eta = 0.3;
    while norm(Grad,2) > max([10^(-2),(1.0/k)*norm((y_step'-
[psik1(2),0;0,psik2(2)]*y_step'),2)])

        Hessian = Hessian_PHI(x_step,y_step,k);
        lambda = 0.0001;
        %Regularization of Hessian

        [R,p] = chol(Hessian);      %Checking for positive definiteness using
cholsky factorization
        while p >0                  %p > 0 => not positive definite
            Hessian = Hessian + lambda*eye(length(Hessian));
            [R,p] = chol(Hessian);
            lambda = 10*lambda;
        end

        %Finding Direction

        delta_xs = Hessian\-Grad;

        %Finding x_step: Armijio Rule

       alpha = 1;

       while PHI(x_step+alpha*delta_xs,y_step,k)-PHI(x_step,y_step,k) >=
abs(eta * alpha * Grad'*delta_xs) || (alpha == 0)      %Trying to prevent
armijio rule loop executing infinitely

            alpha = alpha/2;
       end
         %Updating x_step

        x_step = x_step+alpha*delta_xs;

        %Updating grad_PHI, psi_values

        PHI_Values = PHI(x_step,y_step,k);
        Grad = Grad_PHI(x_step,y_step,k);
        psik1 = psik(x_step,y_step,k,1);
        psik2 = psik(x_step,y_step,k,2);
        n_steps = n_steps+1;
    end

    %display number of Newtons iterations
```

```
    n_steps;

    %Updating y_step
    PHI_Values = PHI(x_step,y_step,k);
    Grad = Grad_PHI(x_step,y_step,k);
    psik1 = psik(x_step,y_step,k,1);
    psik2 = psik(x_step,y_step,k,2);
    y_step = [y_step(1)*psik1(2),y_step(2)*psik2(2)];
    fprintf('Newton Iteration = % d; x1 = % f; x2 = % f;x3 = % f; x4 = % f;
function = % f; y1=% f , y2=% f \n', n_steps, x_step(1),
x_step(2),x_step(3),x_step(4), 2*x_step(1) - 3*x_step(2),
y_step(1),y_step(2));
end
```

The output obtained are as follows:

Output:

Newton Iteration = 4; x1 = 0.081790; x2 = 0.101398;x3 = 0.200189; x4 = 0.245357; function = -0.140616; y1= 0.002569 , y2= 0.060673

Newton Iteration = 0; x1 = 0.081790; x2 = 0.101398;x3 = 0.200189; x4 = 0.245357; function = -0.140616; y1= 0.000007 , y2= 0.001841

Newton Iteration = 3; x1 = 0.059350; x2 = 0.072688;x3 = 0.145377; x4 = 0.178050; function = -0.099364; y1= 0.000000 , y2= 0.031701

Newton Iteration = 0; x1 = 0.059350; x2 = 0.072688;x3 = 0.145377; x4 = 0.178050; function = -0.099364; y1= 0.000000 , y2= 0.546007

Newton Iteration = 8; x1 = 0.069229; x2 = 0.084788;x3 = 0.169577; x4 = 0.207688; function = -0.115906; y1= 0.000000 , y2= 0.043253