Table of Contents

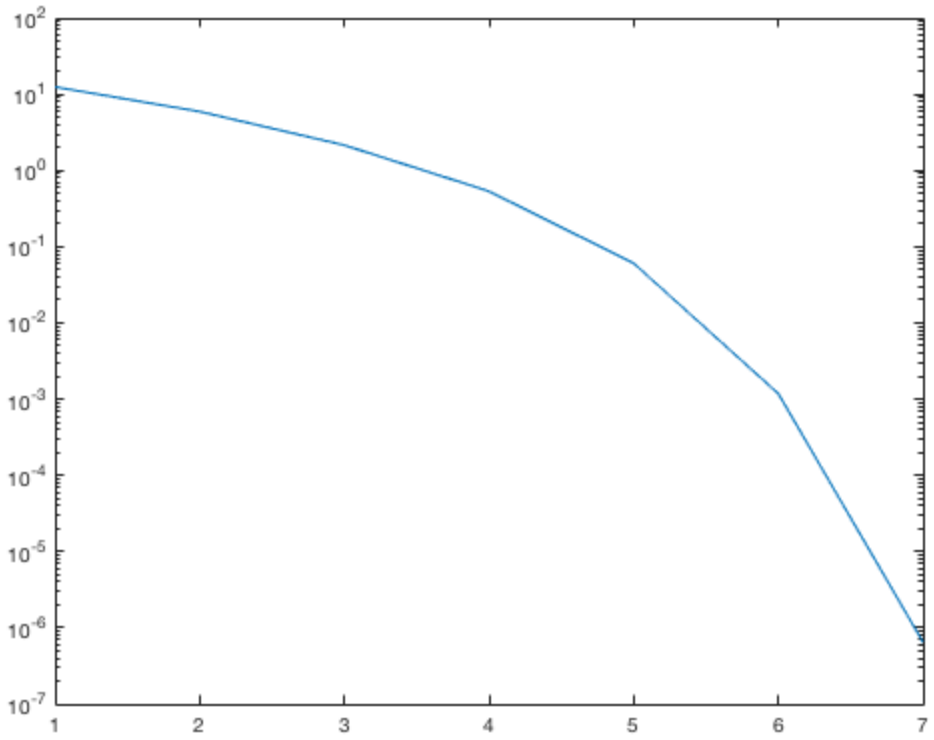
Problem 3, this time in matlab

```
clear all; close all;

% Set parameters
n = 100;
p = 30;
alpha = 0.05;
beta = .5;
iters = 1000;
epsilon = 1e-8;

% Choose A randomly
rng('default')
A = randn(p,n);

% Check if A is full rank
while(rank(A)<p && rank(A)<n)
    A = randn(p,n);
end

% Get random estimate of x
x = rand(n,1); % Choose x in [0,1]

% Set b using the estimated x
b = A*x;

% Get function values
f = @(x) x'*log(x);
grad = @(x) log(x) + ones(length(x),1);
hess = @(x) diag(1./x);
```

Newton's method

```
y = [];
s = [];
d = [];
p_star = f(x);
y = [y; p_star];
h = inv(hess(x));
g = grad(x);

dnt = -h*g;
dec = -1*g'*dnt;
d = [d;dec];
```

```
for(i=1:iters)
    if(dec/2<=epsilon)
        break;
    end

    t = 1;

    while(f(x+t*dnt)>p_star+alpha*t*g'*dnt)
        t = beta*t;
    end

    x = x+t*dnt;

    p_star = f(x);
    g = grad(x);
    h = inv(hess(x));

    dnt = -h*g;
    dec = -1*g'*dnt;
    d = [d;dec];

    s = [s;t];
    y = [y;p_star];

end


semilogy(y-p_star)
hold on
```

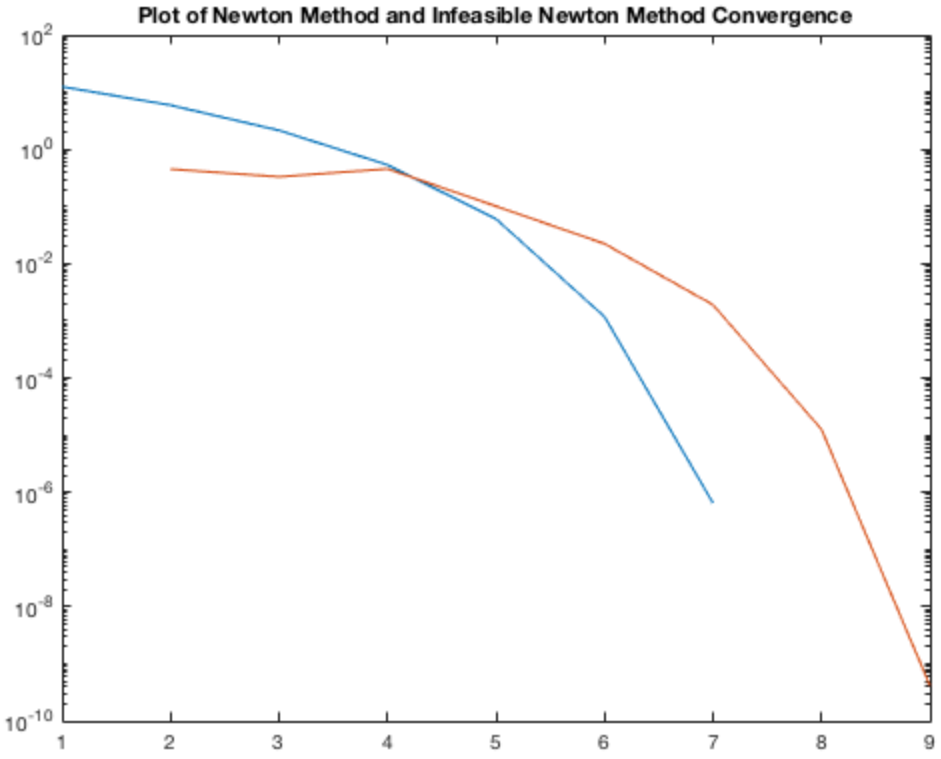


Now want to use the infeasible start newton method

```
% Compute the primal and dual netwon steps
nu = 10*ones(p,1);
y = [];
d1 = [];

% Compute residuals
r_d = @(x,nu) A'*nu + grad(x);
r_p = @(x) A*x-b;
R = @(x,nu) -1*[r_d(x,nu); r_p(x)];

% Create matrix to solve to deltas
Obj = @(hx) [hx,A';A,zeros(p,p)];
y = [y;f(x)];

for(i=1:iters)

    % Get primal and dual deltas
    hx = hess(x);
    O_dum = Obj(hx);
    R_dum = R(x,nu);
    D = O_dum\R_dum;
```

```
    dnt = D(1:n);
    nnt = D(n+1:n+p);

    % Backtracking
    t = 1;
    while(norm(R(x+t*dnt,nu+t*nnt))>(1-alpha*t)*norm(R(x,nu)) && min(x
+t*dnt)<=0)
        t = beta*t;
    end

    x = x+dnt;
    nu = nu+nnt;

    p_star1 = f(x);
    y = [y;p_star1];

    if r_p(x)<=1e-8 & norm(R(x,nu))<=epsilon
        break;
    end
end


for(i=1:iters)
    if(dec/2<=epsilon)
        break;
    end

    t = 1;

    while(f(x+t*dnt)>p_star1+alpha*t*g'*dnt)
        t = beta*t;
    end

    x = x+t*dnt;

    p_star1 = f(x);
    g = grad(x);
    h = inv(hess(x));

    dnt = -h*g;
    dec = -1*g'*dnt;

    s = [s;t];
    y = [y;p_star1];

end

semilogy(real(y-p_star1))
title('Plot of Newton Method and Infeasible Newton Method
 Convergence')
```

Warning: Negative data ignored



Published with MATLAB® R2018b