CS 421: Fall 1997

Solutions to Assignment 6

1. The global minimum (and the only local minumum) of Rosenbrock's function

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

is at $(x_1, x_2) = (1, 1)$.

```
function [f,g,H]=Rosenb(x)
% Rosenbrock's function
%    f=Rosenb(x)        gives the function value
%    [f,g]=Rosenb(x)    function value and gradient
%    [f,g,H]=Rosenb(x)  function value, gradient, and Hessian

a=x(2)-x(1)^2;
b=1-x(1);
f=100*a^2+b^2;
% If more than one output argument, compute gradient
if nargout > 1
    g=[-400*x(1)*a-2*b;  200*a];
end
% If more than 2 output arguments, compute Hessian
if nargout > 2
    H=[-400*x(2)+1200*x(1)^2+2,  -400*x(1)
              -400*x(1),           200    ];
end
```

The following function is used to generate contour plots. Note that the spacing of the contour lines is logarithmic.

```
function plotRosen(xmin,xmax,ymin,ymax)
% Contour plot for Rosenbrock's function
n=100;
x=linspace(xmin,xmax,n);
y=linspace(ymin,ymax,n);
[X,Y]=meshgrid(x,y);
for i=1:n
    for j=1:n
       Z(i,j)=Rosenb([X(i,j); Y(i,j)]);
    end
end
contour(X,Y,Z,4.^(0:7),'y');
grid off;
```

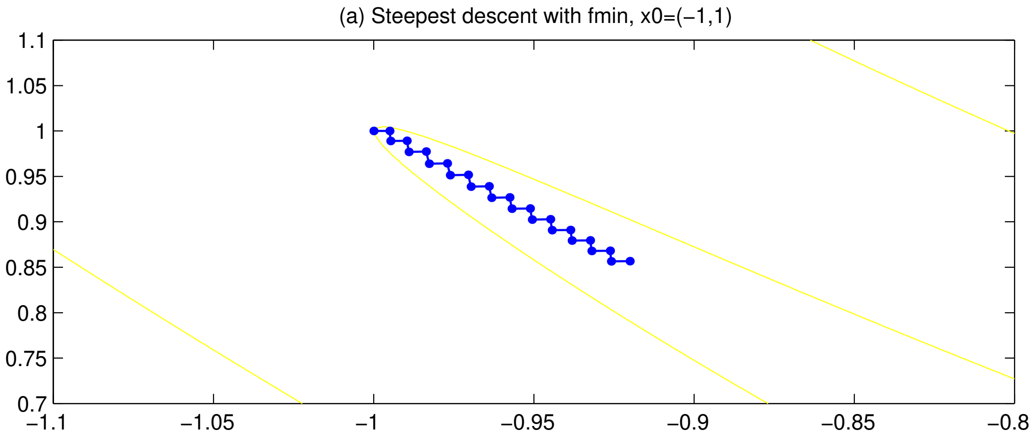
And to be able to use Rosenbrock's function with **fmin** we need this:

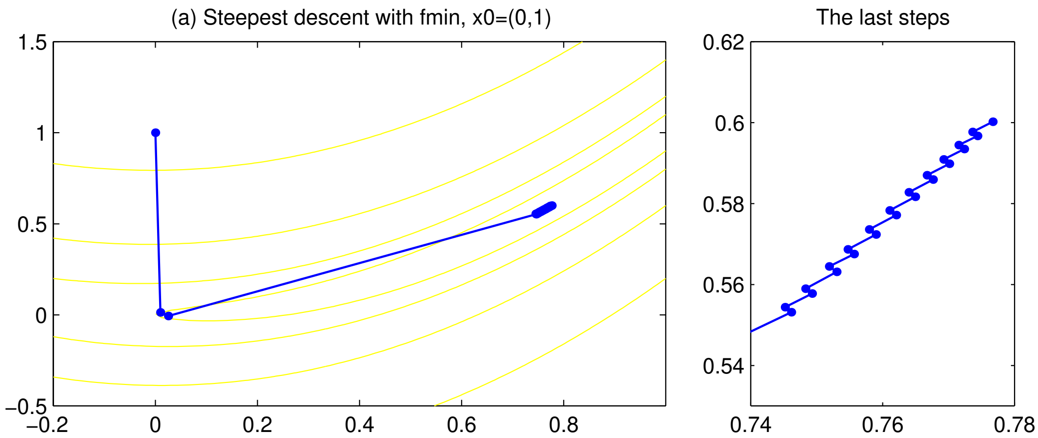
```
function f=Rosenb2(alpha,x,d)
% Used by the function fmin in prob1a
f=Rosenb(x+alpha*d);
```

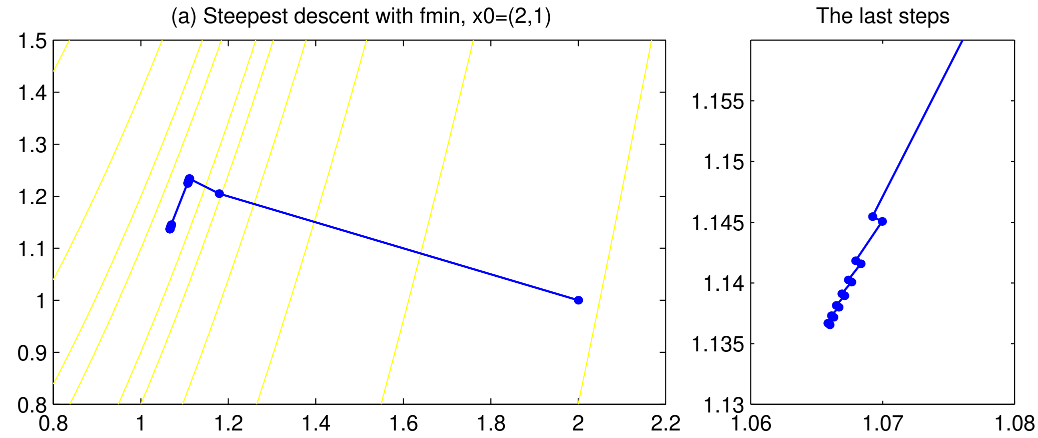
(a) Steepest descent with line search using Matlab's **fmin**:

```
function x=prob1a(x0)
% Steepest descent with line search using fmin
tol=1e-12;
kmax=25;
gamma=1e-4;
rho=0.9;

k=0;
x=x0;
[f,g]=Rosenb(x(:,1));
while (norm(g)>=tol) & (k<kmax)
    k=k+1;
    d=-g;
    alpha_max=min([1,1/norm(d)]);
    alpha=fmin('Rosenb2',0,alpha_max,[],x(:,k),d);
    x(:,k+1)=x(:,k)+alpha*d;
    [f_new,g]=Rosenb(x(:,k+1));
    if (norm(x(:,k+1)-x(:,k))/max([norm(x(:,k)),1]) < tol) ...
       | (abs(f_new-f)/max([abs(f),1]) < tol)
            break;
    end
    f=f_new;;
    fprintf('k=%d  f=%e\n',k,f)
end
fprintf('x=[%f, %f]\n',x(:,k+1))
h=plot(x(1,:),x(2,:),'-');
set(h,'LineWidth',1,'Marker','.','Markersize',12);
```

$x_0 = [-1, 1]^T$	$x_0 = [0, 1]^T$	$x_0 = [2, 1]^T$
k=1 f=3.989976e+00	k=1 f=9.975423e-01	k=1 f=3.431296e+00
k=2 f=3.979114e+00	k=2 f=9.554334e-01	k=2 f=1.260067e-02
k=3 f=3.968203e+00	k=3 f=6.570928e-02	k=3 f=1.225155e-02
k=4 f=3.956311e+00	k=4 f=6.501037e-02	k=4 f=1.219678e-02
k=5 f=3.944404e+00	k=5 f=6.421914e-02	k=5 f=1.214215e-02
k=6 f=3.931369e+00	k=6 f=6.344828e-02	k=6 f=1.207277e-02
k=7 f=3.918262e+00	k=7 f=6.252718e-02	k=7 f=1.200389e-02
k=8 f=3.905400e+00	k=8 f=6.163496e-02	k=8 f=1.190345e-02
k=9 f=3.892471e+00	k=9 f=6.093394e-02	k=9 f=1.180384e-02
k=10 f=3.879773e+00	k=10 f=6.024944e-02	k=10 f=1.158607e-02
k=11 f=3.867012e+00	k=11 f=5.944823e-02	k=11 f=1.137264e-02
k=12 f=3.854469e+00	k=12 f=5.866943e-02	k=12 f=5.287206e-03
k=13 f=3.841870e+00	k=13 f=5.791292e-02	k=13 f=4.900595e-03
k=14 f=3.829475e+00	k=14 f=5.717672e-02	k=14 f=4.787329e-03
k=15 f=3.817044e+00	k=15 f=5.646969e-02	k=15 f=4.675735e-03
k=16 f=3.804805e+00	k=16 f=5.578065e-02	k=16 f=4.627523e-03
k=17 f=3.792549e+00	k=17 f=5.512797e-02	k=17 f=4.579640e-03
k=18 f=3.780472e+00	k=18 f=5.449081e-02	k=18 f=4.543961e-03
k=19 f=3.768396e+00	k=19 f=5.389642e-02	k=19 f=4.508458e-03
k=20 f=3.756486e+00	k=20 f=5.331502e-02	k=20 f=4.478816e-03
k=21 f=3.744490e+00	k=21 f=5.278072e-02	k=21 f=4.449314e-03
k=22 f=3.732651e+00	k=22 f=5.225698e-02	k=22 f=4.423441e-03
k=23 f=3.720734e+00	k=23 f=5.178168e-02	k=23 f=4.397679e-03
k=24 f=3.708965e+00	k=24 f=5.131477e-02	k=24 f=4.378397e-03
k=25 f=3.697127e+00	k=25 f=5.079875e-02	k=25 f=4.359180e-03
x=[-0.920037, 0.856756]	x=[0.776698, 0.600202]	x=[1.065989, 1.136548]


(a) Steepest descent with fmin, x0=(−1,1)


(a) Steepest descent with fmin, x0=(0,1)
The last steps


(a) Steepest descent with fmin, x0=(2,1)
The last steps

(b) Newton's method:

```
function x=prob1b(x0)
% Newton's method
tol=1e-12;
kmax=25;

k=0;
x=x0;
[f,g,H]=Rosenb(x(:,1));
while (norm(g)>=tol) & (k<kmax)
    k=k+1;
    d=-H\g;
    x(:,k+1)=x(:,k)+d;
    f_new=Rosenb(x(:,k+1));
    if (norm(x(:,k+1)-x(:,k))/max([norm(x(:,k)),1]) < tol) ...
       | (abs(f_new-f)/max([abs(f),1]) < tol)
            break;
    end
    [f,g,H]=Rosenb(x(:,k+1));
    fprintf('k=%d  f=%e\n',k,f)
end
fprintf('x=[%f, %f]\n',x(:,k+1))
h=plot(x(1,:),x(2,:),'-');
set(h,'LineWidth',1,'Marker','.','Markersize',12);
```

$x_0 = [-1, 1]^T$

```
k=1  f=1.600000e+03
k=2  f=6.163420e-27
x=[1.000000, 1.000000]
```

$x_0 = [0, 1]^T$

```
k=1  f=1.010076e+00
k=2  f=9.998998e+01
k=3  f=2.525187e-05
k=4  f=6.376565e-08
k=5  f=1.016692e-23
x=[1.000000, 1.000000]
```

$x_0 = [2, 1]^T$

```
k=1  f=9.966750e-01
k=2  f=9.911639e+01
k=3  f=3.021924e-07
k=4  f=9.132023e-12
k=5  f=0.000000e+00
x=[1.000000, 1.000000]
```

(c) Newton's method with backtracking line search:

```
function x=prob1c(x0)
% Newton's method with backtracking
tol=1e-12;
kmax=25;
gamma=1e-4;
rho=0.9;

k=0;
x=x0;
[f,g,H]=Rosenb(x(:,1));
while (norm(g)>=tol) & (k<kmax)
    k=k+1;
    d=-H\g;
    h=gamma*g'*d;
    alpha=1;
    f_new=Rosenb(x(:,k)+alpha*d);
    while f_new > f+alpha*h
        alpha=rho*alpha;
        f_new=Rosenb(x(:,k)+alpha*d);
    end
    x(:,k+1)=x(:,k)+alpha*d;
    if (norm(x(:,k+1)-x(:,k))/max([norm(x(:,k)),1]) < tol) ...
       | (abs(f_new-f)/max([abs(f),1]) < tol)
            break;
    end
    [f,g,H]=Rosenb(x(:,k+1));
    fprintf('k=%d  f=%e\n',k,f)
end
fprintf('x=[%f, %f]\n',x(:,k+1))
h=plot(x(1,:),x(2,:),'-');
set(h,'LineWidth',1,'Marker','.','Markersize',12);
```

$x_0 = [-1, 1]^T$

```
k=1  f=3.701403e+00
k=2  f=2.599937e+00
k=3  f=2.575220e+00
k=4  f=1.555155e+00
k=5  f=1.428924e+00
k=6  f=8.569174e-01
k=7  f=7.701005e-01
k=8  f=4.103025e-01
k=9  f=3.588024e-01
k=10  f=1.559197e-01
k=11  f=1.468702e-01
k=12  f=3.583822e-02
k=13  f=3.241002e-02
k=14  f=2.109119e-03
k=15  f=3.868716e-04
k=16  f=2.068083e-07
k=17  f=4.279836e-12
k=18  f=2.548724e-23
x=[1.000000, 1.000000]
```

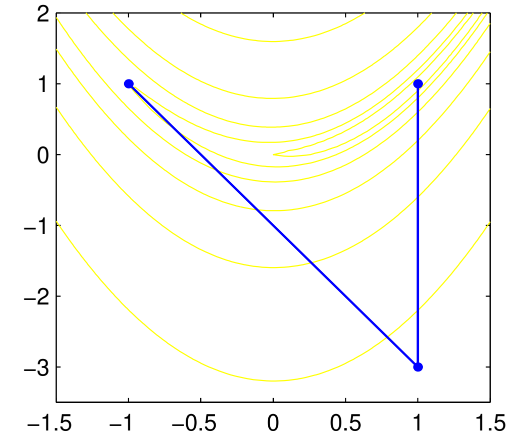
$x_0 = [0, 1]^T$

```
k=1  f=1.010076e+00
k=2  f=9.814253e-01
k=3  f=4.865502e-01
k=4  f=4.474246e-01
k=5  f=1.927425e-01
k=6  f=1.740140e-01
k=7  f=5.116383e-02
k=8  f=4.088937e-02
k=9  f=5.321486e-03
k=10  f=2.081231e-03
k=11  f=6.938428e-06
k=12  f=4.791664e-09
k=13  f=3.759931e-17
x=[1.000000, 1.000000]
```

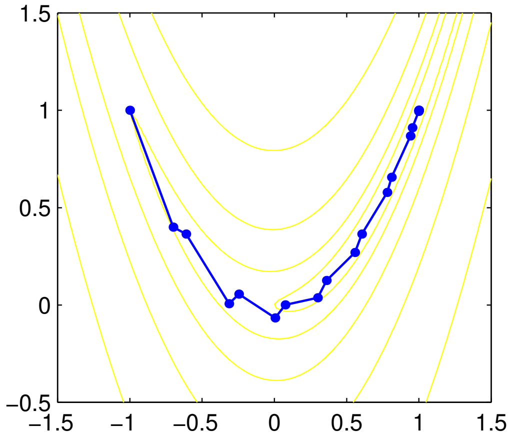
$x_0 = [2, 1]^T$

```
k=1  f=9.966750e-01
k=2  f=9.683910e-01
k=3  f=4.783168e-01
k=4  f=4.381545e-01
k=5  f=1.886043e-01
k=6  f=1.680944e-01
k=7  f=4.979714e-02
k=8  f=3.870206e-02
k=9  f=5.151410e-03
k=10  f=1.932815e-03
k=11  f=6.847587e-06
k=12  f=4.666417e-09
k=13  f=4.089497e-17
x=[1.000000, 1.000000]
```

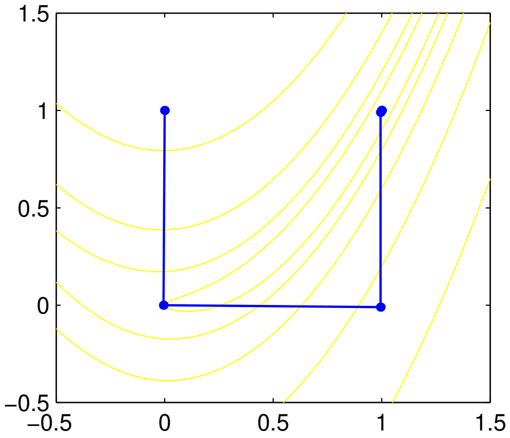
(b) Newton, x0=(−1,1)



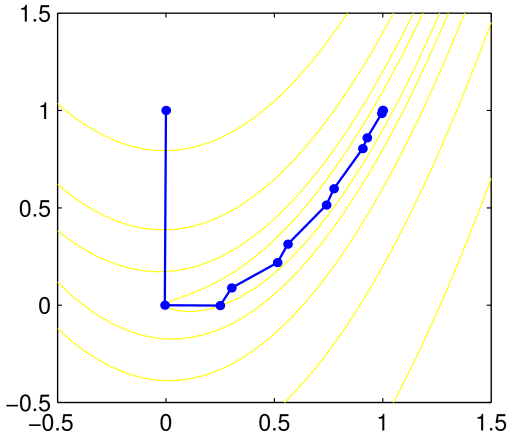
(c) Newton with backtracking, x0=(−1,1)



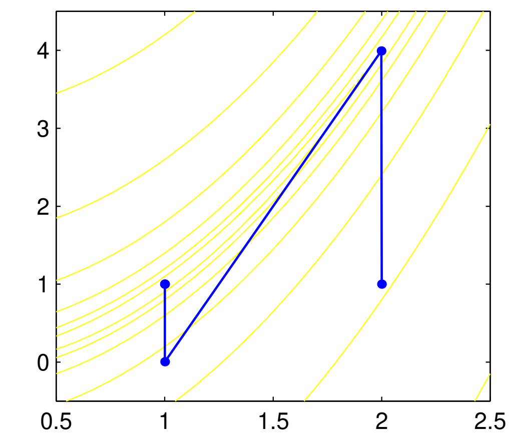
(b) Newton, x0=(0,1)



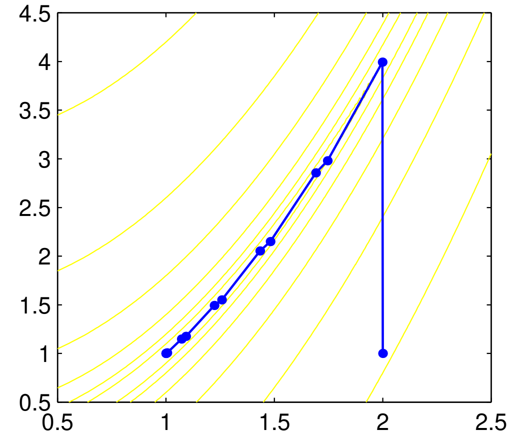
(c) Newton with backtracking, x0=(0,1)



(b) Newton, x0=(2,1)



(c) Newton with backtracking, x0=(2,1)



2. Let s_k be the solution to the trust region subproblem

$$\min_{||s||\leq\Delta_k} \nabla f(x_k)^T s + \frac{1}{2}s^T\nabla^2 f(x_k)s.$$

Since the quadratic $q(s) = \nabla f(x_k)^T s + \frac{1}{2}s^T\nabla^2 f(x_k)s$ is zero at the center of the trust region, $s = 0$, we have $q(s_k) \leq q(0) = 0$. Therefore

$$\rho_k = \frac{f(x_k + s_k) - f(x_k)}{\nabla f(x_k)^T s_k + \frac{1}{2}s_k^T\nabla^2 f(x_k)s_k} < 0$$

implies that $f(x_k + s_k) - f(x_k) > 0$, because the numerator and denominator must have opposite signs.

An alternative approach, that many students used, is to use the fact that there exists $\lambda \geq 0$ such that $(\nabla^2 f(x_k)+\lambda I)s_k = -\nabla f(x_k)$ and $\nabla^2 f(x_k)+\lambda I$ is symmetric, positive semidefinite. Substituing in for $\nabla f(x_k)$ we get

$$q(s_k) = -\frac{1}{2}s_k^T(\nabla^2 f(x_k) + \lambda I)s_k - \frac{\lambda}{2}s_k^T s_k \leq 0.$$

But this is not as elementary as the first approach.

3. For convenience let's drop the subscript k and write A and A_+ instead of A_k and A_{k+1}, respectively. We have $A = JJ^T$, $y^T s > 0$ and $w = \alpha J^T s$ with

$$\alpha^2 = \frac{y^T s}{s^T JJ^T s}.$$

So the BFGS update is

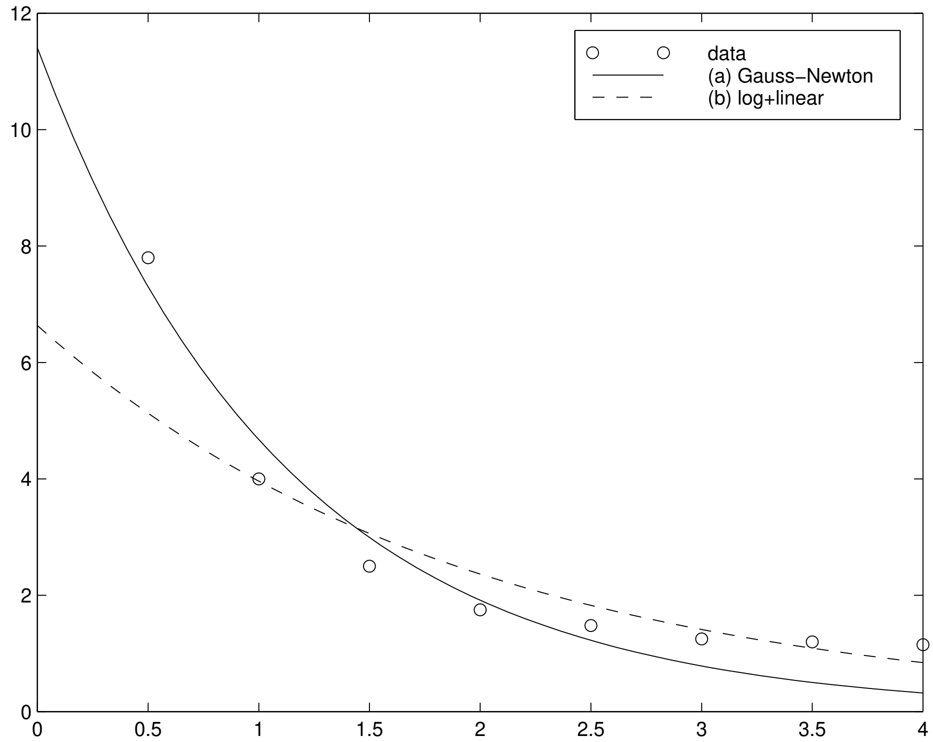
$$\begin{aligned}
A_+ &= J_+J_+^T \\
&= \left(J + \frac{(y-Jw)w^T}{w^T w}\right)\left(J^T + \frac{w(y-Jw)^T}{w^T w}\right) \\
&= JJ^T + \frac{Jw(y-Jw)^T}{w^T w} + \frac{(y-Jw)w^T J^T}{w^T w} + \frac{(y-Jw)w^T w(y-Jw)^T}{(w^T w)^2} \\
&= JJ^T + \frac{Jw(y^T - w^T J^T)}{w^T w} + \frac{(y-Jw)w^T J^T}{w^T w} + \frac{(y-Jw)(y^T - w^T J^T)}{w^T w} \\
&= JJ^T + \frac{Jwy^T}{w^T w} - \frac{Jww^T J^T}{w^T w} + \frac{yw^T J^T}{w^T w} - \frac{Jww^T J}{w^T w} + \frac{yy^T}{w^T w} - \frac{yw^T J^T}{w^T w} - \frac{Jwy^T}{w^T w} + \frac{Jww^T J^T}{w^T w} \\
&= JJ^T + \frac{yy^T}{w^T w} - \frac{Jww^T J^T}{w^T w} \\
&= JJ^T + \frac{yy^T}{\alpha^2 s^T JJ^T s} - \frac{\alpha^2 JJ^T ss^T JJ^T}{\alpha^2 s^T JJ^T s} \\
&= A + \frac{yy^T}{s^T y} - \frac{Ass^T A}{s^T As}.
\end{aligned}$$

4.
```
% PROBLEM 4

% Data points
t=[0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0]';
y=[7.80 4.00 2.50 1.75 1.48 1.25 1.20 1.15]';

% PART (a)

tol=1e-12;
kmax=40;
gamma=1e-4;
rho=0.9;

x=rand(2,1);
s=ones(2,1);
k=0;
while (norm(s)/max([norm(x),1])>tol) & (k<kmax)
    J=[-exp(x(2)*t), -x(1)*t.*exp(x(2)*t)];
    r=y-x(1)*exp(x(2)*t);
    fprintf('k=%d  x(1)=%f  x(2)=%f  norm(r)=%e\n',k,x,norm(r))
    s=-J\r;
    alpha=1;
    x_new=x+alpha*s;
    r_new=y-x_new(1)*exp(x_new(2)*t);
    while norm(r_new) > sqrt(1+2*gamma*alpha)*norm(r)
        alpha=rho*alpha;
        x_new=x+alpha*s;
        r_new=y-x_new(1)*exp(x_new(2)*t);
    end
    x=x+alpha*s;
    k=k+1;
end
xa=x
ra=y-xa(1)*exp(xa(2)*t);
norm_r=norm(ra)

% PART (b)

B=[ones(length(t),1) t];
z=B\log(y);
xb=[exp(z(1)) z(2)]'
rb=y-xb(1)*exp(xb(2)*t);
norm_r=norm(rb)


% Plot to see how curves fit
tt=linspace(0,4,50);
plot(t,y,'o',tt,xa(1)*exp(xa(2)*tt),'-',tt,xb(1)*exp(xb(2)*tt),'--')
legend('data','(a) Gauss-Newton','(b) log+linear')
```

The output:

```
k=0  x(1)=0.950129  x(2)=0.231139  norm(r)=7.581627e+00
k=1  x(1)=4.291837  x(2)=-0.925248  norm(r)=6.221069e+00
k=2  x(1)=11.488175  x(2)=-0.858782  norm(r)=1.602242e+00
k=3  x(1)=11.312314  x(2)=-0.881807  norm(r)=1.558164e+00
k=4  x(1)=11.377862  x(2)=-0.889292  norm(r)=1.557561e+00
k=5  x(1)=11.398117  x(2)=-0.891384  norm(r)=1.557515e+00
k=6  x(1)=11.403688  x(2)=-0.891962  norm(r)=1.557512e+00
k=7  x(1)=11.405218  x(2)=-0.892121  norm(r)=1.557511e+00
k=8  x(1)=11.405639  x(2)=-0.892164  norm(r)=1.557511e+00
k=9  x(1)=11.405754  x(2)=-0.892176  norm(r)=1.557511e+00
k=10  x(1)=11.405786  x(2)=-0.892180  norm(r)=1.557511e+00
k=11  x(1)=11.405795  x(2)=-0.892181  norm(r)=1.557511e+00
k=12  x(1)=11.405797  x(2)=-0.892181  norm(r)=1.557511e+00
k=13  x(1)=11.405798  x(2)=-0.892181  norm(r)=1.557511e+00
k=14  x(1)=11.405798  x(2)=-0.892181  norm(r)=1.557511e+00
k=15  x(1)=11.405798  x(2)=-0.892181  norm(r)=1.557511e+00
k=16  x(1)=11.405798  x(2)=-0.892181  norm(r)=1.557511e+00
k=17  x(1)=11.405798  x(2)=-0.892181  norm(r)=1.557511e+00
k=18  x(1)=11.405798  x(2)=-0.892181  norm(r)=1.557511e+00
k=19  x(1)=11.405798  x(2)=-0.892181  norm(r)=1.557511e+00
k=20  x(1)=11.405798  x(2)=-0.892181  norm(r)=1.557511e+00
k=21  x(1)=11.405798  x(2)=-0.892181  norm(r)=1.557511e+00

xa =

   11.4058
   -0.8922


norm_r =

    1.5575


xb =

    6.6356
   -0.5159


norm_r =

    2.8446
```

The results are quite different (see explanation on next page). The plot shows how the computed curves fit the data.



The reason why the answers are different is that we are not solving the same problem. In (a) we are solving

$$\min_x \sum_i \left(y_i - x_1 e^{x_2 t_i}\right)^2$$

whereas in (b) we are solving

$$\min_x \sum_i (\log(y_i) - \log(x_1) - x_2 t_i)^2 = \min_x \sum_i \left((\log(y_i) - \log(x_1 e^{x_2 t_i})\right)^2 .$$

The two objective functions are quite different. Taking logarithm like that does not preserve the location of minima.