Aditya Vipradas
1209435588

MAE 598

Homework 3.

Design optimization.

1. (1) Least squares problem.

The training data is provided.

$x = \{x_1, x_2\}$

$y = p$.

There are 11 data points for x & y.
The problem is to fit (x, y) in the function

$$p = x_1 \exp\left(A_{12}\left(\frac{A_{21}x_2}{A_{12}x_1 + A_{21}x_2}\right)^2\right)p_{1\,sat}$$
$$+ x_2 \exp\left(A_{21}\left(\frac{A_{12}x_1}{A_{12}x_1 + A_{21}x_2}\right)^2\right)p_{2\,sat}$$

Here, p_1sat & p_2sat are evaluated from the given data of a_1, a_2 & a_3 and the Antoine equation

$$\log(p_{sat}) = a_1 - \frac{a_2}{T + a_3}$$

$T = 20°C$

The function can be generalized as

$$y = X\beta + \epsilon$$

$\epsilon \rightarrow$ error & $\beta = \{A_{12}, A_{21}\}$.

The problem at hand is to estimate β such that the L2 norm of ϵ is minimized.

i.e. find β such that

$$\min \|y - X\beta\|^2,$$

the given function being a non-linear model.

Problem 1(2)
Gradient Descent Implementation

Aditya Vipradas
ASU ID: 1209435588

February 24, 2016

The gradient descent method is implemented with the stopping criterion of 1e-6 and without the Armijo line search. The initial guess value is (1,1). The convergence and contour plots obtained are as follows. As observed, the values of A_{12} and A_{21} are 1.9572 and 1.685 respectively which agree with the values obtained by the implementation of the lsqnonlin command in MATLAB. See the attached gradient descent codes.

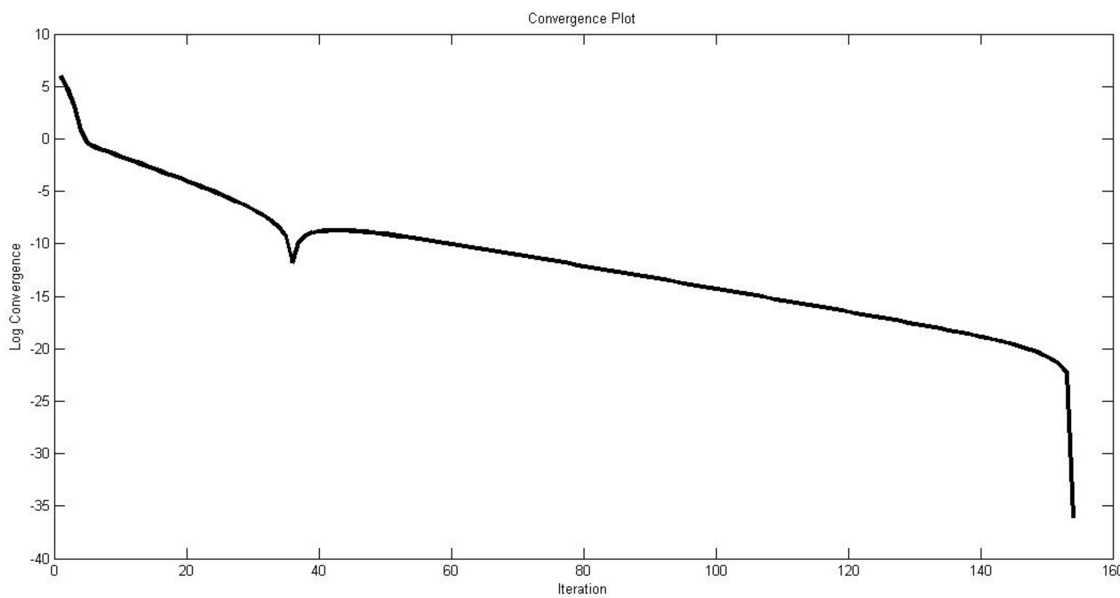

Figure 1: Convergence plot

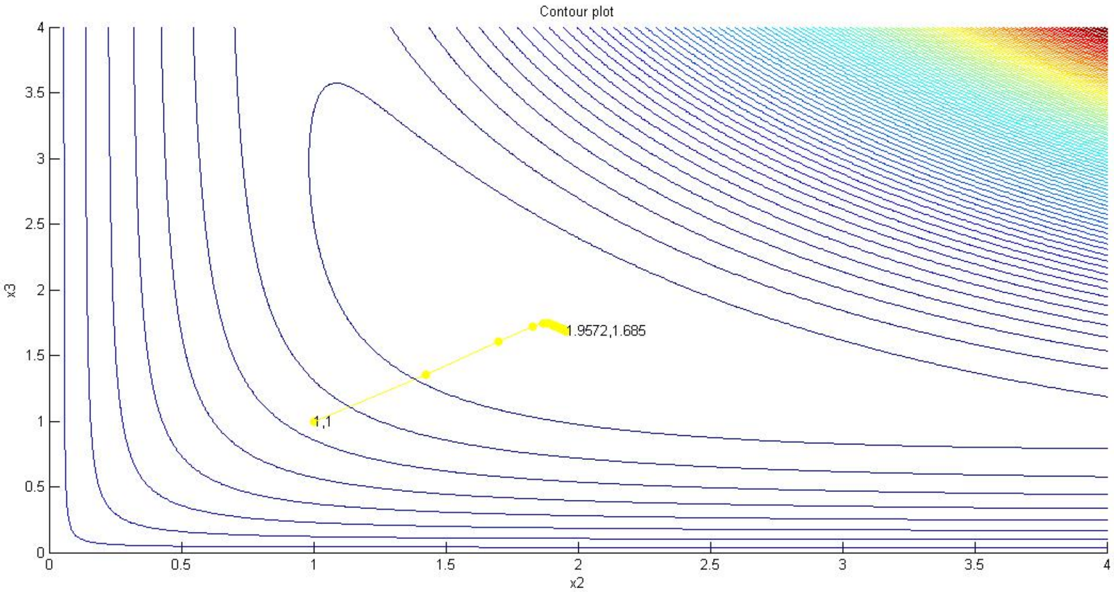

Figure 2: Contour plot with the values of A_{12} and A_{21}

```
function y = p1(x)

    %FUNCTION FILE

    %para consists of the a1, a2 and a3 values for water (row1) and
    %1,4 dioxane (row2)
    para = [8.07131, 1730.63, 233.426; 7.43155, 1554.679, 240.337];

    %define the temperature (deg celsius)
    T = 20;

    %evaluate the saturation pressures for water and 1,4 dioxane
    for i=1:1:2
        psat(i) = 10^(para(i,1) - para(i,2)/(T + para(i,3)));
    end

    %data
    xdata = [0.0:0.1:1];
    ydata = [28.1, 34.4, 36.7, 36.9, 36.8, 36.7, 36.5, 35.4, 32.9, ...
        27.7, 17.5];

    y = 0;

    %function
    for i = 1:1:length(xdata)
        x1 = xdata(i);
        x2 = 1 - x1;
        yval = ydata(i);

        y = y + (x1 * exp(x(1)*(x(2)*x2/(x(1)*x1 + ...
        x(2)*x2))^2) * psat(1) + x2 * exp(x(2)* ...
        (x(1)*x1/(x(1)*x1 + x(2)*x2))^2) * psat(2) - yval)^2;
    end
```

```
function g = p1gfd(x)

    %GRADIENT FILE

    g = [0;0];
    %function gradient
    g(1) = (p1([x(1)+0.01, x(2)])-p1(x))/0.01;
    g(2) = (p1([x(1), x(2)+0.01])-p1(x))/0.01;
```

```
%%%%%%%%%%%%%% Gradient Descent Implementation %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% By Max Yi Ren and Emrah Bayrak %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function solution = gradient(f,g,H,x0,opt)
    % Set initial conditions
     x = x0; % Set current solution to the initial guess
     iter = 0; % Set iteration counter to 0

    % Initialize a structure to record search process
     solution = [];

    % Calculate the norm of the gradient
     gnorm = norm(g(x),2); % this needs to be a scalar

    % Set the termination criterion:
    while gnorm>opt.eps % if not terminated
        iter = iter + 1

        % save current step
        solution.x([1,2],iter) = x;
        % solution.x is an array of solutions, i.e., a matrix
        % opt.linesearch switches line search on or off.
        % You can first set the variable "a" to different constant values and see how
it
        % affects the convergence.
        if opt.linesearch
            a = lineSearch1(f,g,H,x,opt);
        else
            a = 0.001;
        end


        % Gradient descent:
        d = -1*g(x);
        x = x + a*d; % update x based on gradient info
        % Update termination criterion:
        gnorm = norm(g(x),2); % update the norm of gradient
    end
    disp(x);
```

```
%%%%%%%%%%%%%% Main Entrance %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% By Max Yi Ren and Emrah Bayrak %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Instruction: Please read through the code and fill in blanks
% (marked by ***). Note that you need to do so for every involved
% function, i.e., m files.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Optional overhead
clear; % Clear the workspace
% Note: for debugging purpose, do not use "clear all"
close all; % Close all windows
clc; %Clear screen

%% Optimization settings
% Here we specify the objective function by giving the function handle to a
% variable, for example:
f = @(x)p1(x); % replace rosenbrock with your objective function
% In the same way, we also provide the gradient and the Hessian of the
% objective:
g = @(x)p1gfd(x); % replace accordingly
H = @(x)p1H(x); % replace accordingly
% Note that explicit gradient and Hessian information is only optional.
% However, providing these information to the search algorithm will save
% computational cost from finite difference calculations for them.

% Specify algorithm
opt.alg = 'gradient';
%opt.alg = 'newton';

% Turn on or off line search. You could turn on line search once other
% parts of the program are debugged.
opt.linesearch = false; % or true

% Set the tolerance to be used as a termination criterion:
opt.eps = 1e-6; % this should be a small number like 1e-3

% Set the initial guess:
x0 = [1;1]; % this should be a p-dim vector where p is the size of the
% problem

%% Run optimization
% Run your implementation of the gradient descent and Newton's method. See
% gradient.m and newton.m.
if strcmp(opt.alg,'gradient')
    solution = gradient(f,g,H,x0,opt);
elseif strcmp(opt.alg,'newton')
    solution = newton(f,g,H,x0,opt);
end

%% Report
% Implement report.m to generate a report.
disp(solution);
report(solution,f);
```

Problem 1(3)
MATLAB lsqnonlin Implementation

Aditya Vipradas
ASU ID: 1209435588

February 24, 2016

Non-linear regression is performed on the given data to fit it to the given function using the *lsqnonlin* function in MATLAB. The default stopping criterion of 1e-6 is used. Initial guess value is (1,1). The values of A_{12} and A_{21} obtained after the implementation are 1.9584 and 1.6892 respectively which agree with the gradient descent implementation with slight discrepancy. Levenberg-Marquardt algorithm is put to use. Refer the plot below that displays this curve fit. Also refer the attached code.

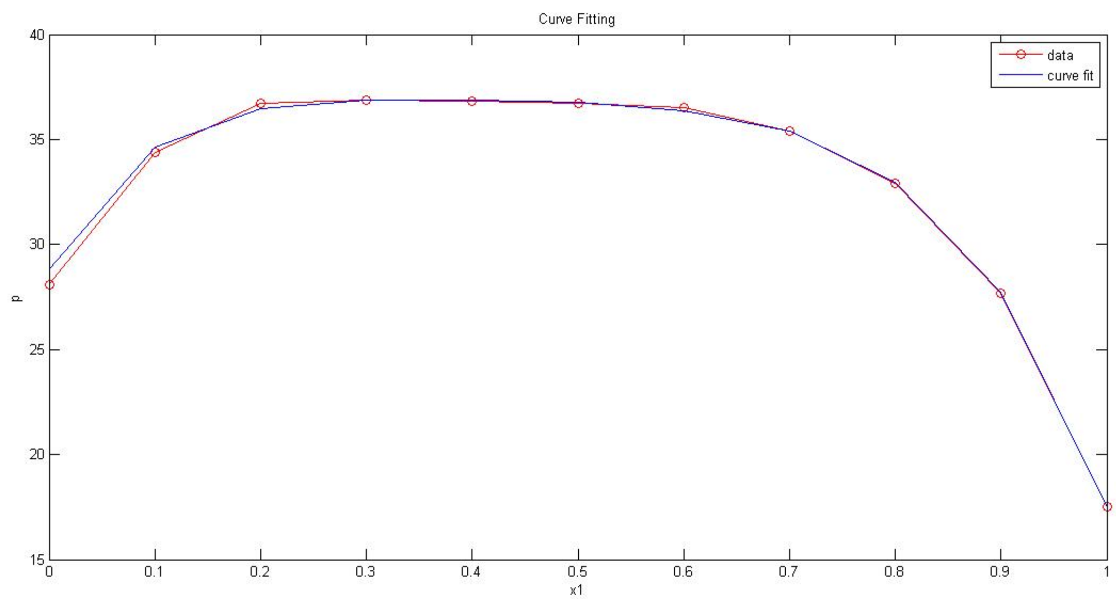

Figure 1: Curve fit

```
%clear screen
clc;
%para consists of the a1, a2 and a3 values for water (row1) and
%1,4 dioxane (row2)
para = [8.07131, 1730.63, 233.426; 7.43155, 1554.679, 240.337];

%define the temperature (deg celsius)
T = 20;

%evaluate the saturation pressures for water and 1,4 dioxane
for i=1:1:2
    psat(i) = 10^(para(i,1) - para(i,2)/(T + para(i,3)));
end

%given data
xdata = [0.0:0.1:1];
ydata = [28.1, 34.4, 36.7, 36.9, 36.8, 36.7, 36.5, 35.4, 32.9, 27.7, 17.5];

%define the function
fun = @(A) xdata.* exp(A(1)*(A(2)*(1-xdata)./(A(1)*xdata + ...
    A(2)*(1-xdata))).^2) * psat(1) + (1-xdata).* exp(A(2)* ...
    (A(1)*xdata./(A(1)*xdata + A(2)*(1-xdata))).^2) * psat(2) - ydata;

%evaluate the A parameters by fitting the data to the described function
%first guess
x0 = [5,5];
options.Algorithm = 'levenberg-marquardt';
A = lsqnonlin(fun,x0,[],[],options);

%display A
disp(A);

%plot data
plot(xdata, ydata, '-or', xdata, fun(A)+ydata, '-b');
title('Curve Fitting');
xlabel('x1');
ylabel('p');
legend('data','curve fit');
```

Problem 2
Neural Network Implementation

Aditya Vipradas
ASU ID: 1209435588

February 23, 2016

The given data of 250 samples is fed to the inbuilt neural network toolbox in MATLAB. Default 10 hidden layers are used. Samples used for training, test and validation are 70, 15 and 15 respectively. The neural network flow diagram is as follows.

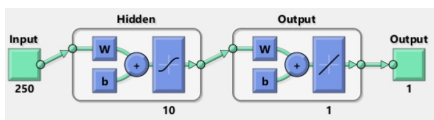

Figure 1: Neural network flow diagram

The performance charts obtained are as follows. As seen in the following plot, best validation is obtained at epoch 6.

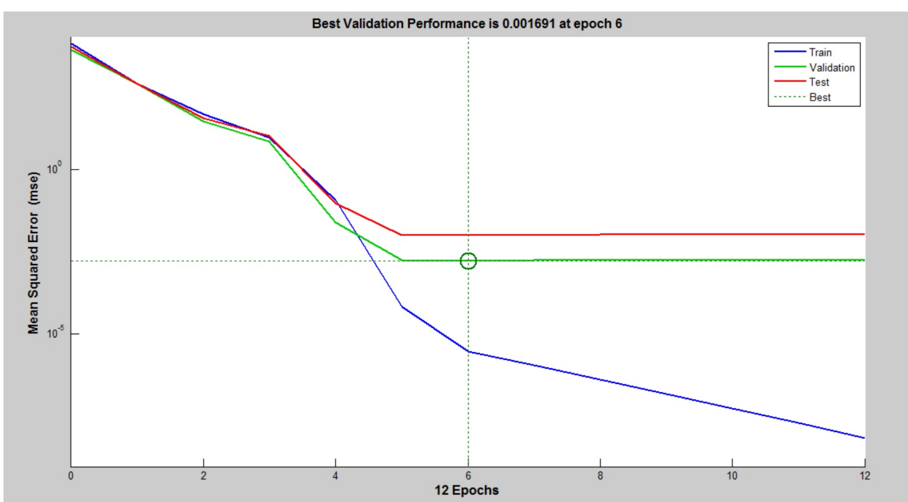

Figure 2: Performance plot

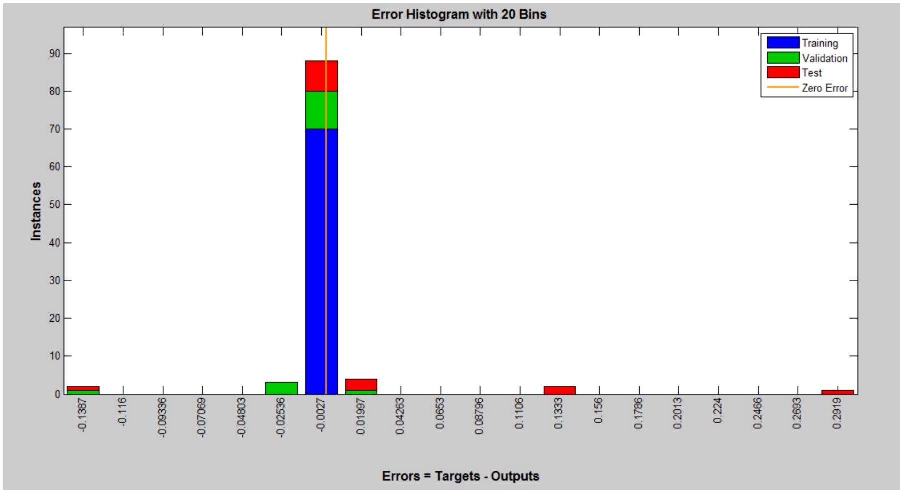

Figure 3: Error Histogram

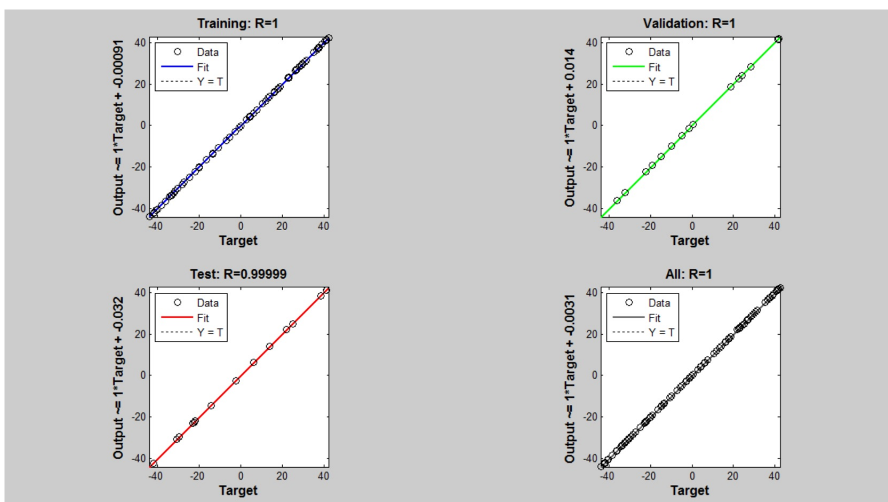

Figure 4: Regression Plot

As seen from the regression plot, the R value measure for all the outputs is 1. This suggests that the curve fit to the given data is acceptable.

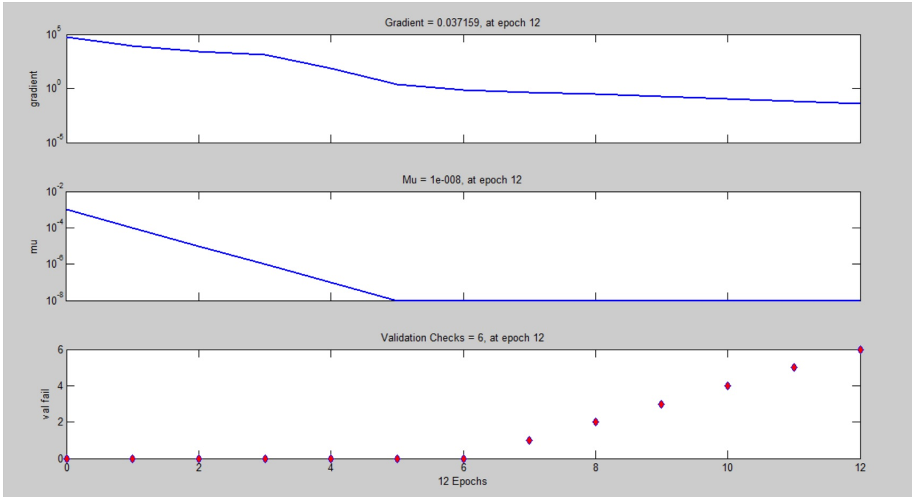

Figure 5: Training state plot

This is the test performance report for the neural network (Levenberg-Marquardt training) implementation on the given non-linear data set.