Problem 1

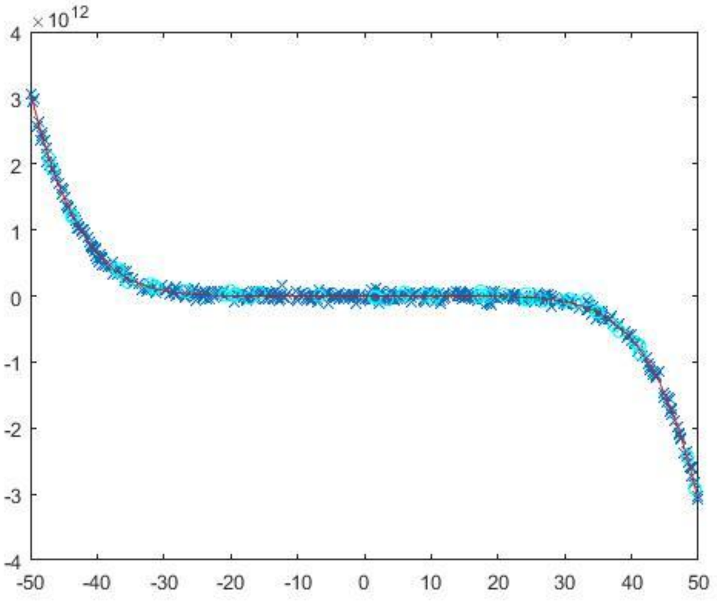
$$f(x;\theta)=\theta_0+\theta_1 x+\theta_2 x^2+\ldots+\theta_d x^d$$

$$R_{emp}(\theta)=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y_i-f(x;\theta))^2$$

Matlab code:

```
function [err,model,errT] = polyreg(x,y,D,xT,yT)
xx = zeros(length(x),D);
for i=1:D
  xx(:,i) = x.^(D-i);
end
model = pinv(xx)*y;
err   = (1/(2*length(x)))*sum((y-xx*model).^2);

if (nargin==5)
  xxT = zeros(length(xT),D);
  for i=1:D
    xxT(:,i) = xT.^(D-i);
  end
  errT  = (1/(2*length(xT)))*sum((yT-xxT*model).^2);
end
```



When D = 12,

error$_{\text{training}}$ = 1.1743e+21 , the minimum of error$_{\text{test}}$ = 8.0383e+20

Problem 2

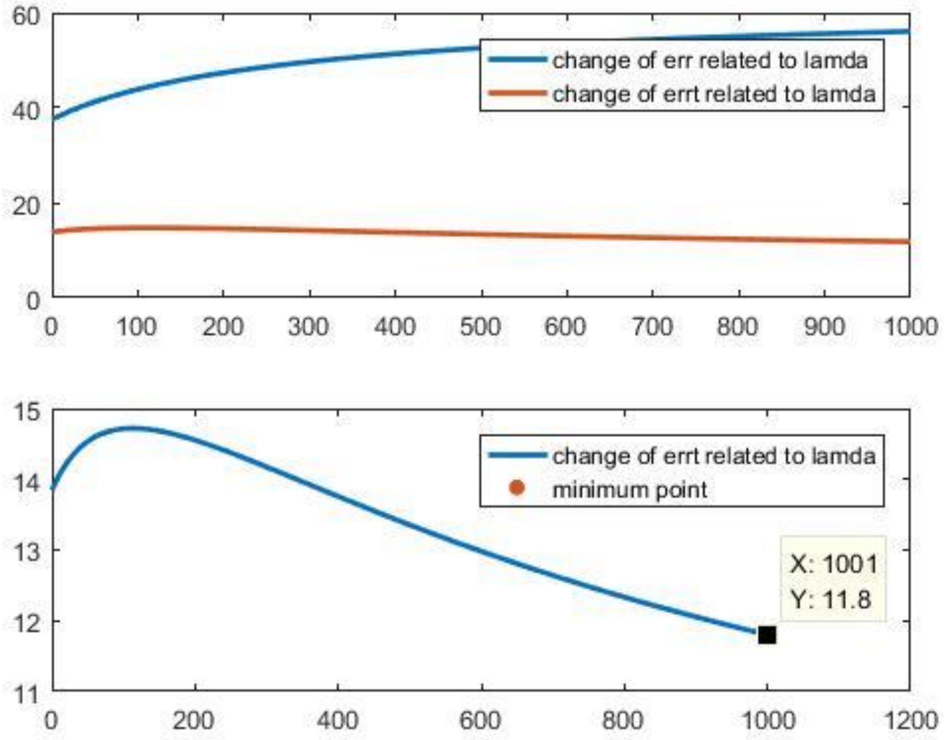
$$f(x;\theta)=\sum_{i=1}^{k}\theta_i x_i$$

$$R_{emp}(\theta)=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y_i-f(x;\theta))^2+\frac{\lambda}{2N}\|\theta\|^2$$

Matlab code:

```
function [err,theta,errt] = polyreg2(x,y,xT,yT,N,NT)
xt = x'; lamda=0; xx = xt * x; m = size(xx);
err= [];errt= [];
for lamda=0:1:1000
    theta = pinv (xx + lamda * eye(m(1),m(2)))* xt * y;
    f = x * theta
    fT = xT * theta
    err = [err sum(((y-f).^2)/2/N) + lamda * (sum(theta.^2)/2/N)]
    errt = [errt sum(((yT-fT).^2)/2/N) + lamda * (sum(theta.^2)/2/N)]
end
```

The error$_{training}$ and error$_{test}$ plot looks like this:



When λ = 1000

The minimum of error$_{test}$ related to lamda = 11.8

Problem 3

$$g(z)=\frac{1}{1+e^{-z}}$$

$$g(-z)=\frac{1}{1+e^{z}}=\frac{e^{-z}}{e^{-z}+1}=\frac{e^{-z}+1-1}{e^{-z}+1}=1-\frac{1}{1+e^{-z}}=1-g(z)$$

$$g(y)=\frac{1}{1+e^{-y}}$$

$$1+e^{-y}=\frac{1}{g(y)}$$

$$e^{-y}=\frac{1}{g(y)}-1=\frac{1-g(y)}{g(y)}$$

$$\ln e^{-y}=\ln\frac{1-g(y)}{g(y)}$$

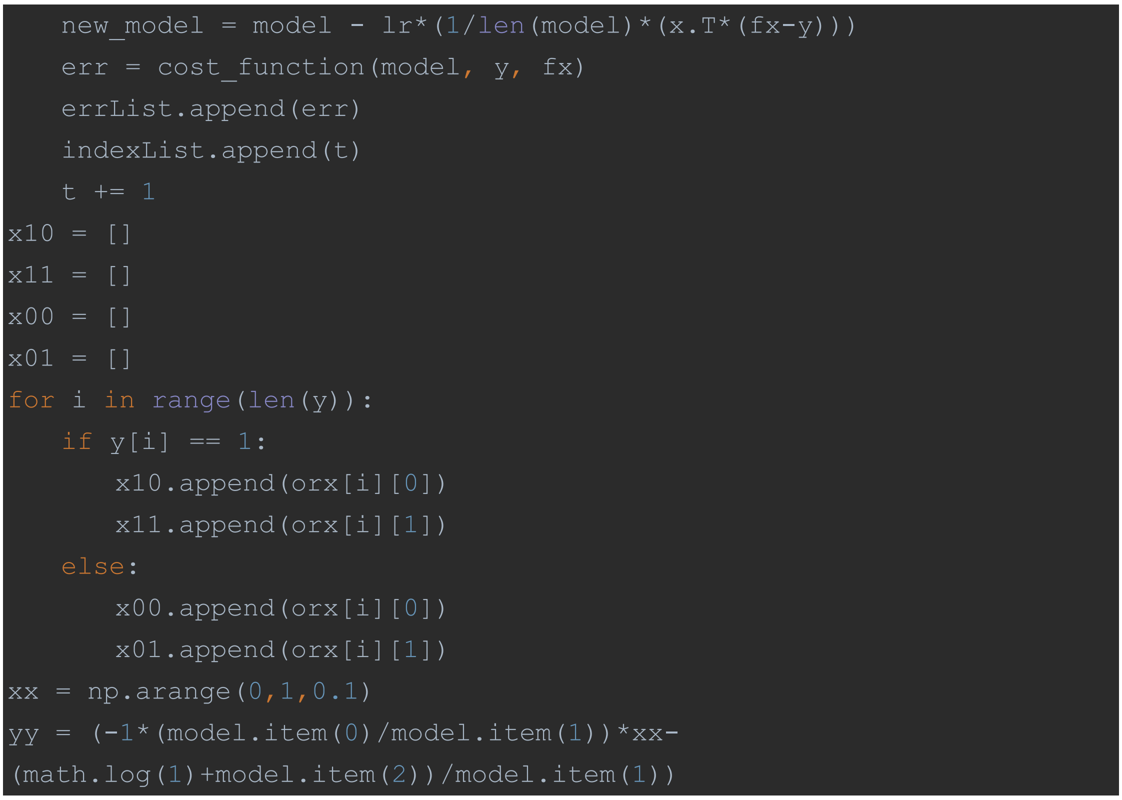
$$y=\ln\frac{g(y)}{1-g(y)}$$

$$g^{-1}(y)=\ln\frac{y}{1-y}$$

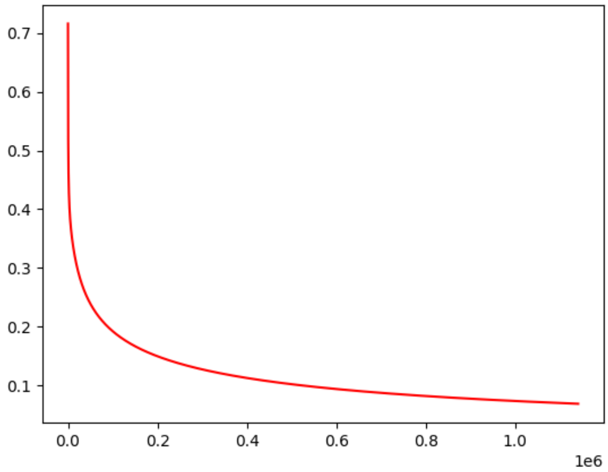
Problem 4

Python code:

```
def cost_function(theta, y, fx):
    loss = 1/y.shape[0]*sum((y-1).T*np.log(1.-fx)-y.T*np.log(fx))
    return loss.item(0)
def update_model(model, lr, tol, x, fx, y):
    new_model = model - lr*(1/len(model)*sum((fx-y)*x))
    return new_model
def logi_reg(x,y,lr,tol,xT=None,yT=None):
orx = x
ory = y
x = np.matrix(x, dtype=float)
y = np.matrix(y,dtype=float)
xT = np.matrix(xT,dtype=float)
yT = np.matrix(yT,dtype=float)
model = np.matrix(np.random.random(x.shape[1]),dtype=float).T
fx = 1/(1+np.exp(-1*x*model))
err = cost_function(model, y, fx)
new_model = model - lr*(1/len(model)*(x.T*(fx-y)))
indexList = []
errList = []
t = 1
while np.linalg.norm(new_model-model) >= tol:
    model = new_model
    fx = 1/(1+np.exp(-1*x*model))
```



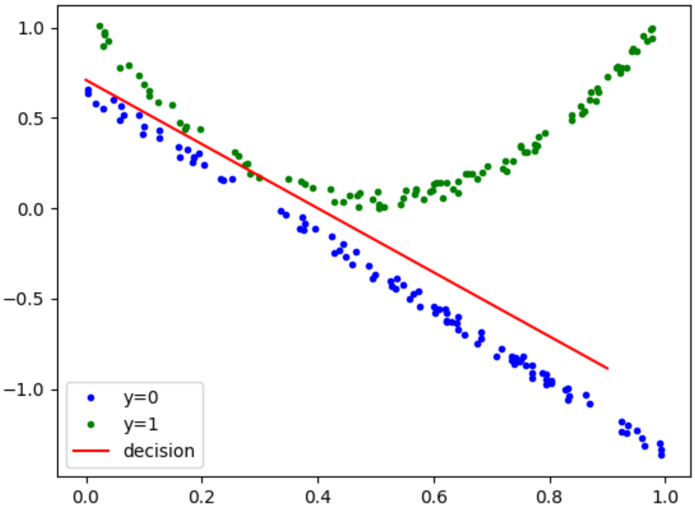
```
    new_model = model - lr*(1/len(model)*(x.T*(fx-y)))
    err = cost_function(model, y, fx)
    errList.append(err)
    indexList.append(t)
    t += 1
x10 = []
x11 = []
x00 = []
x01 = []
for i in range(len(y)):
    if y[i] == 1:
        x10.append(orx[i][0])
        x11.append(orx[i][1])
    else:
        x00.append(orx[i][0])
        x01.append(orx[i][1])
xx = np.arange(0,1,0.1)
yy = (-1*(model.item(0)/model.item(1))*xx-
(math.log(1)+model.item(2))/model.item(1))
```



Learning rate = 0.0001

Tolerance = 0.00001

Error declined while fitting



Decision is made based on the decision line (red line)

0 errors in this prediction

θ = [[39.30047909]

[22.15634712]

[-15.73055102]]