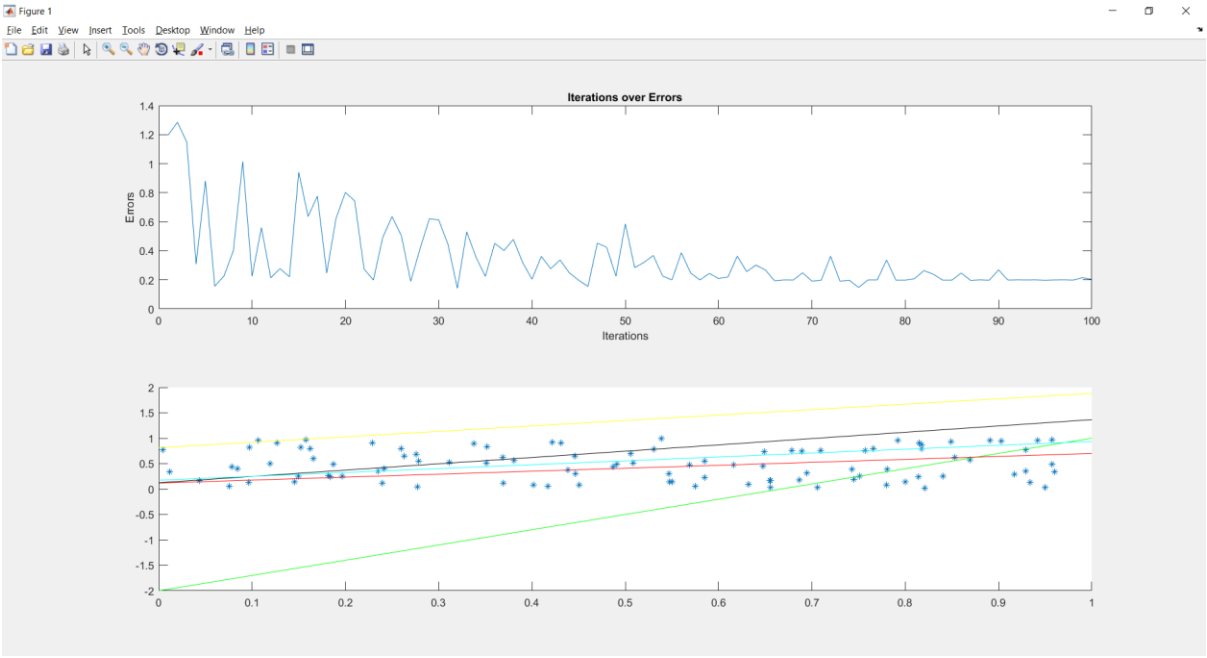
1. Implement the **delta training** rule for a two-input linear unit. Train it to fit the target concept $x_1 + 2x_2 - 2 > 0$. You must generate your own examples as follows: generate random pairs (x_1, x_2) and assign them to the positive class if $x_1 + 2x_2 - 2 > 0$; otherwise assign them to the negative class.

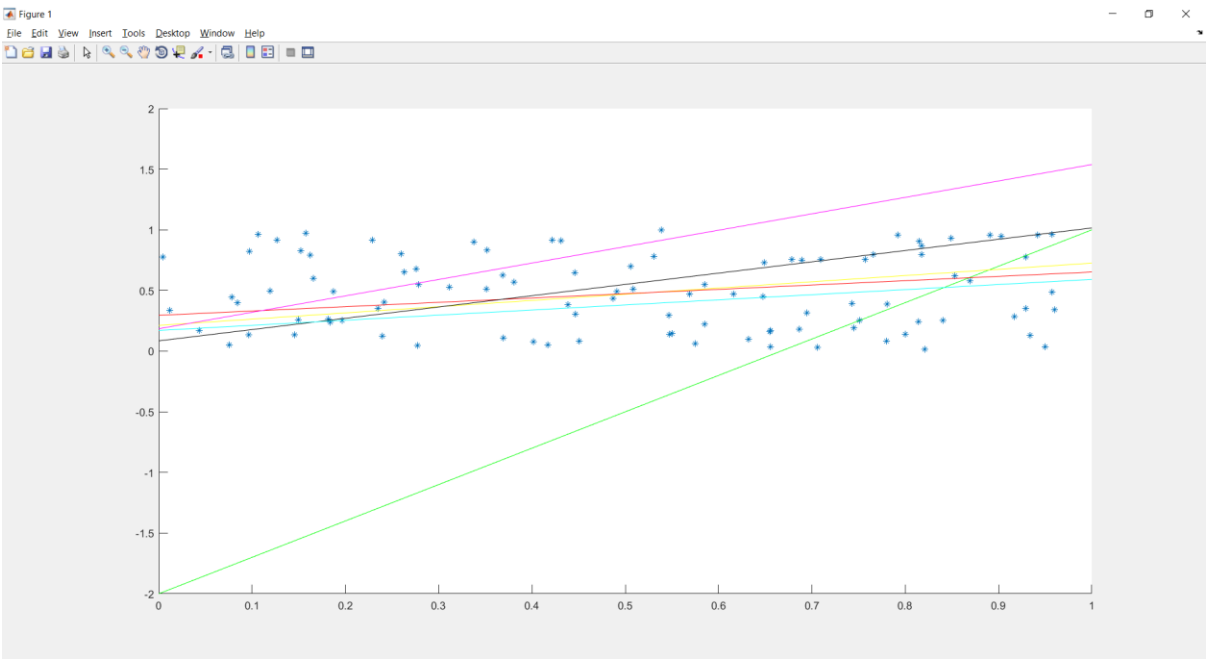
a. Plot the error E as a function of the number of training iterations/epochs.

b. Plot the decision surface after 5, 10, 50, 100 iterations.



c. Use different learning rates, analyze which works better and explain why.

The below graphs shows the surfaces obtained over different learning rates. We observed that if the learning rate is high, it converges to the lowest number of epochs and vice versa.



d. Now implement delta rule in an incremental fashion

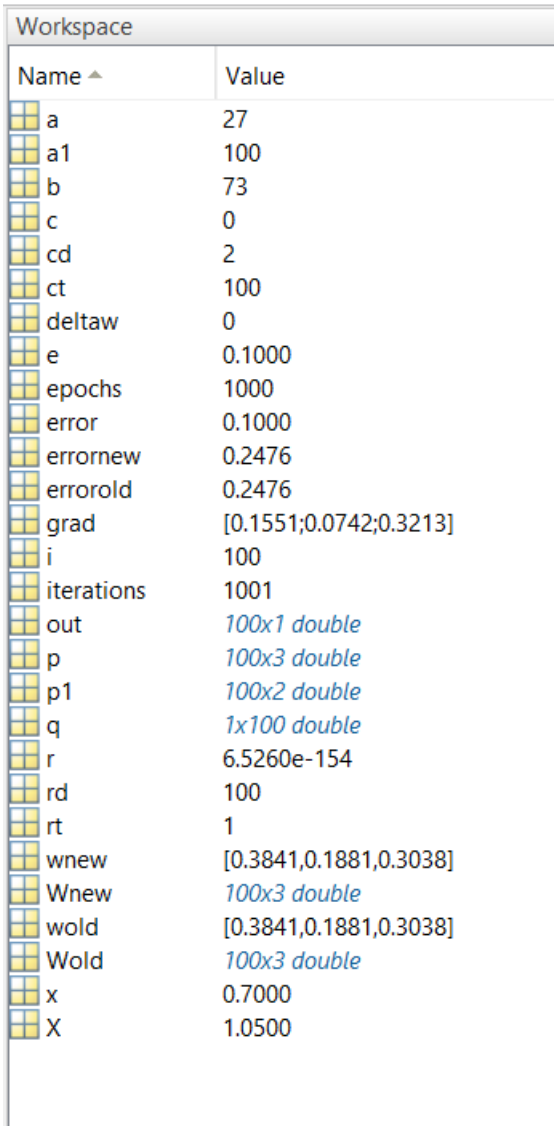
Command Window

```
>> Problem_1_d
Batch Mode:Elapsed time is 0.180478 seconds.
Number of times weights updated
iteration1 =

        9403

Incremental Mode:Elapsed time is 0.066105 seconds.
Number of times weights updated
iteration2 =

   111
```

Workspace

Name	Value
a	27
a1	100
b	73
c	0
cd	2
ct	100
e1	0.1000
e2	0.0999
epochs	10000
error	0.1000
i	100
iteration1	9403
iteration2	111
iterations	0
p	*100x2 double*
p1	*100x2 double*
q	*1x100 double*
r	1.0000e-03
rd	100
rt	1
w1	[0.3010,0.5504,-0.1287]
w2	[0.2396,0.5794,-0.1102]

Problem 2: Consider now exactly the same problem as above and implement variable learning rates



Problem 3: Derive a **gradient descent training rule** for a single unit with output o