COL774

Assignment 1

### Devansh Dalal (2012CS10224)

*Q1.* Linear Regression

1. **Batch Gradient Descent**:

The Batch gradient descent is implemented in function *p1(η,ε)* which takes the learning rate *η* and stopping criteria *ε* as arguments for optimizing *J(θ)*. The value of variation of number of iterations as the function of *η* for *ε=0.000001* is shown in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| *η* | *Θ0* | *Θ1* | no. of iterations |
| 0.001 | 5.8391 | 4.6169 | 13538 |
| 0.01 | 5.8391 | 4.6169 | 1349 |
| 0.1 | 5.8391 | 4.6169 | 130 |
| 0.5 | 5.8391 | 4.6169 | 21 |
| 1 | 5.8391 | 4.6169 | 4 |
| 1.6 | 5.8391 | 4.6169 | 28 |
| 1.9 | 5.8391 | 4.6169 | 127 |
| 1.99 | 5.8391 | 4.6169 | 1320 |
| 2.1 |  |  | doesn't converge |

Variation of convergence on learning rate

**Observations**:

* + The optimal values of *η,* were observed as follow.

*Θ0=*5.8391

*Θ1=4.6169*

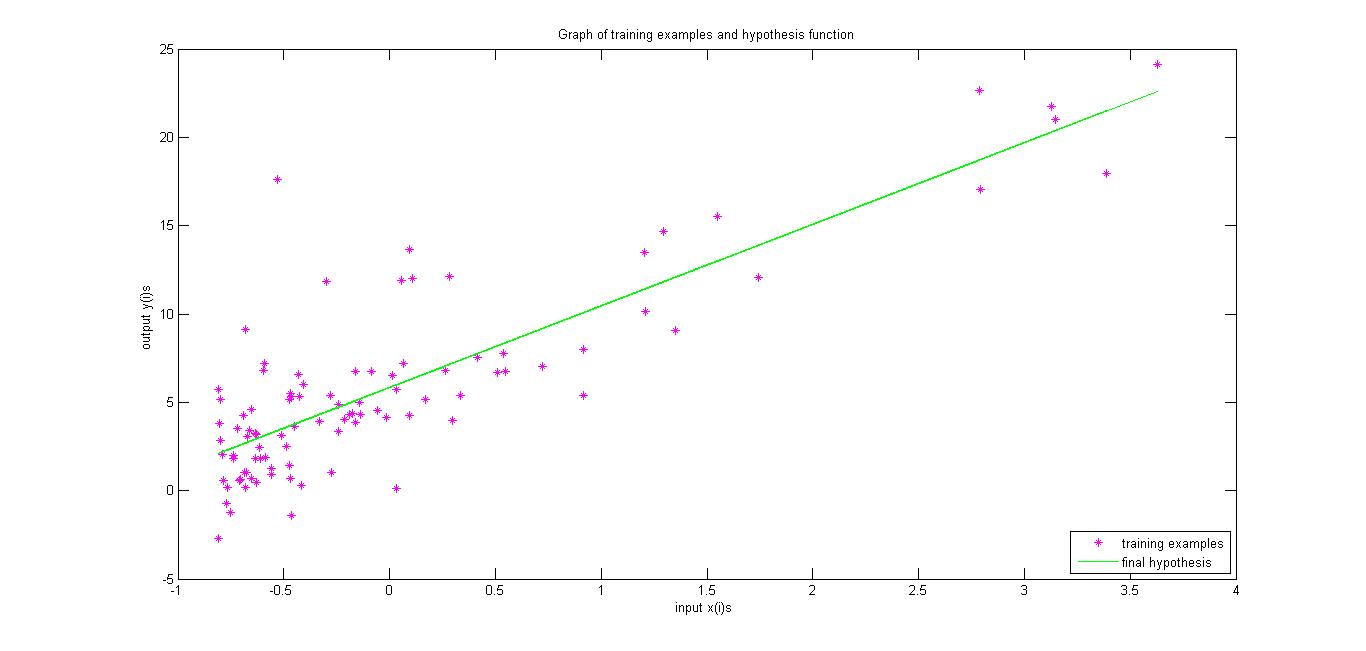
*η =1*

*ε=0.000001*

* + At higher values of learning rate the values of jumps over the optimal value in successive iterations and at about *η* >= 2.0, the batch gradient doesn't converge.
  + One method to solve these overshooting of theta is that to change *η* dynamically, i.e. decrease *η* with the increasing number of iterations.
  + On increasing stopping criteria the no. of iterations reduced but the accuracy of solution was also decreased.

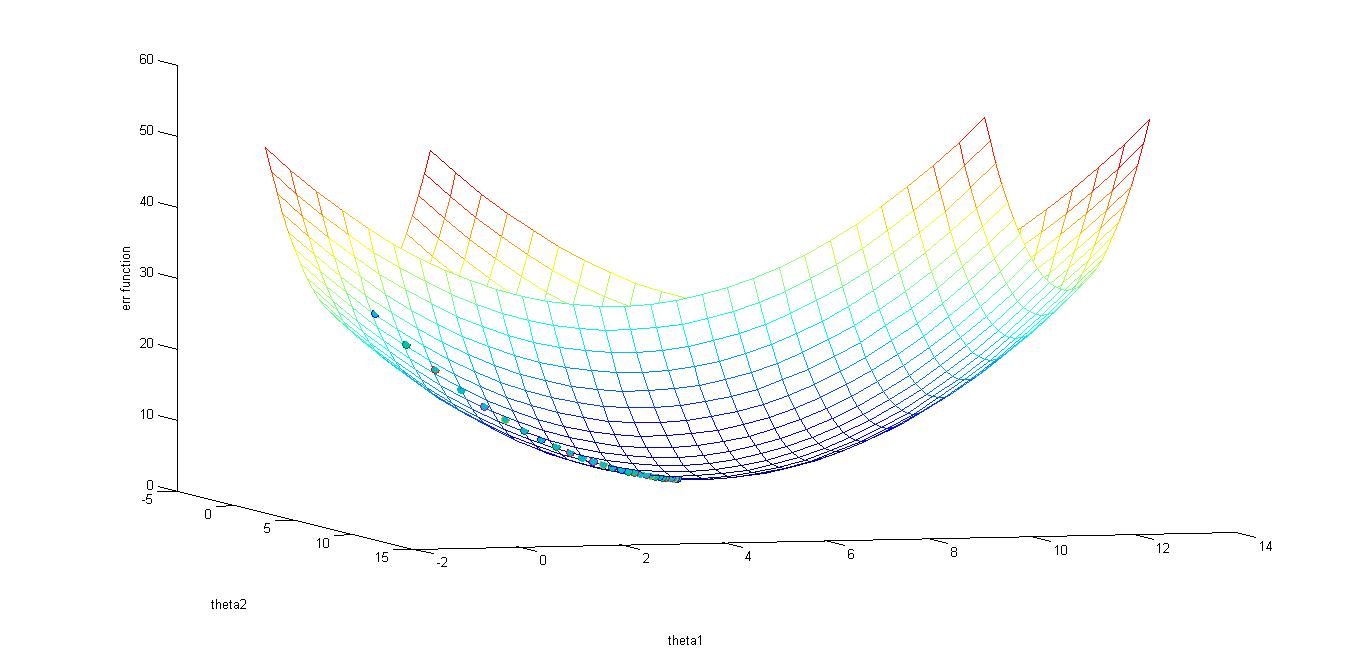
1. **Batch Gradient Descent**:

The hypothesis function learned by my algorithm for *η*=1 and *ε=0.000001* is shown in the figure below.



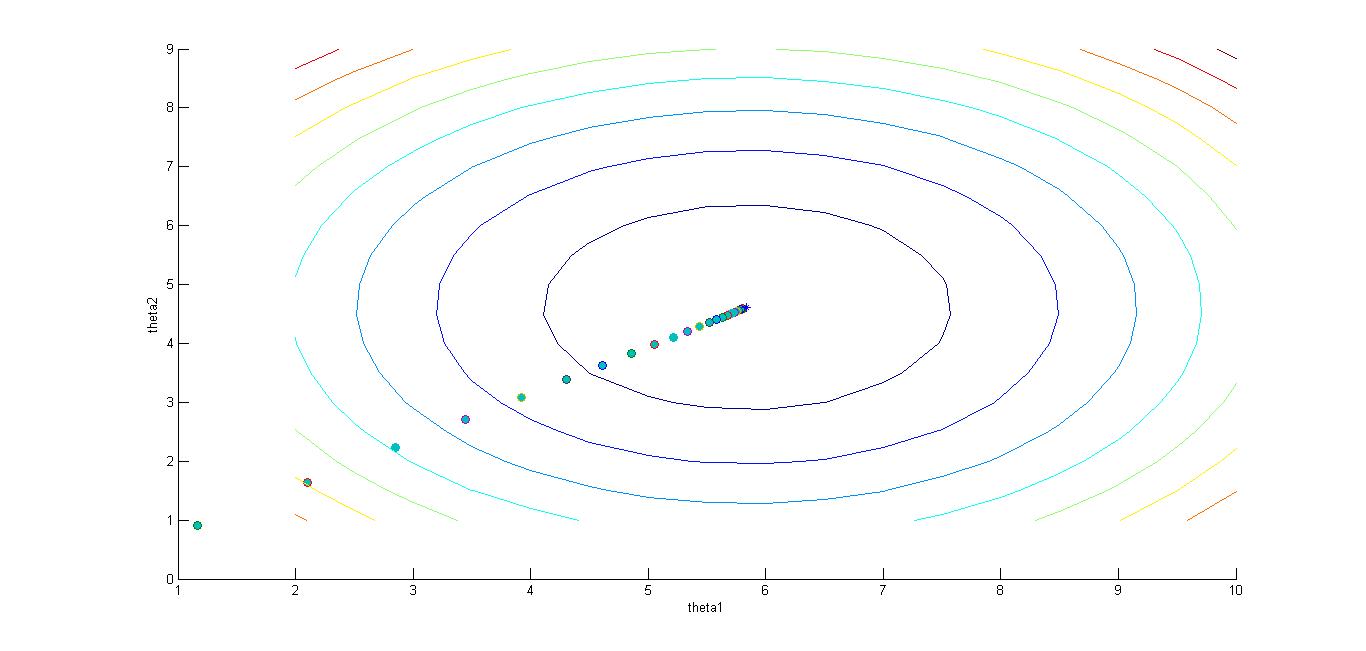
Plot of hypothesis vs training examples

1. I have plotted the for different values of using the meshgrid function in a 3D plot and the animation shows the actual convergence of gradient descent by displaying the error value using the current set of parameters at each iteration at a time gap of 1 second.



Convergence of Gradient Descent

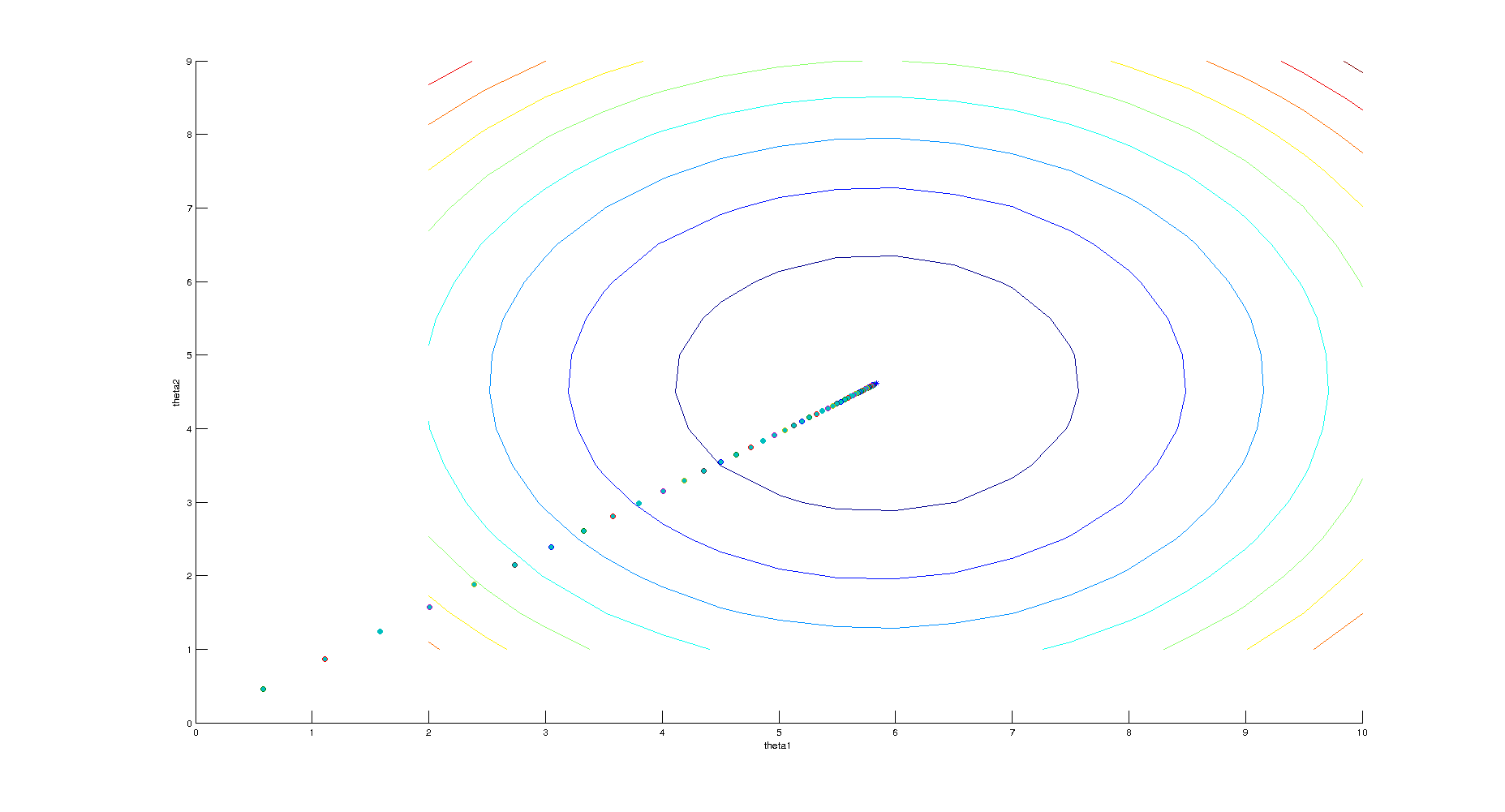
1. **Contour of the error function**



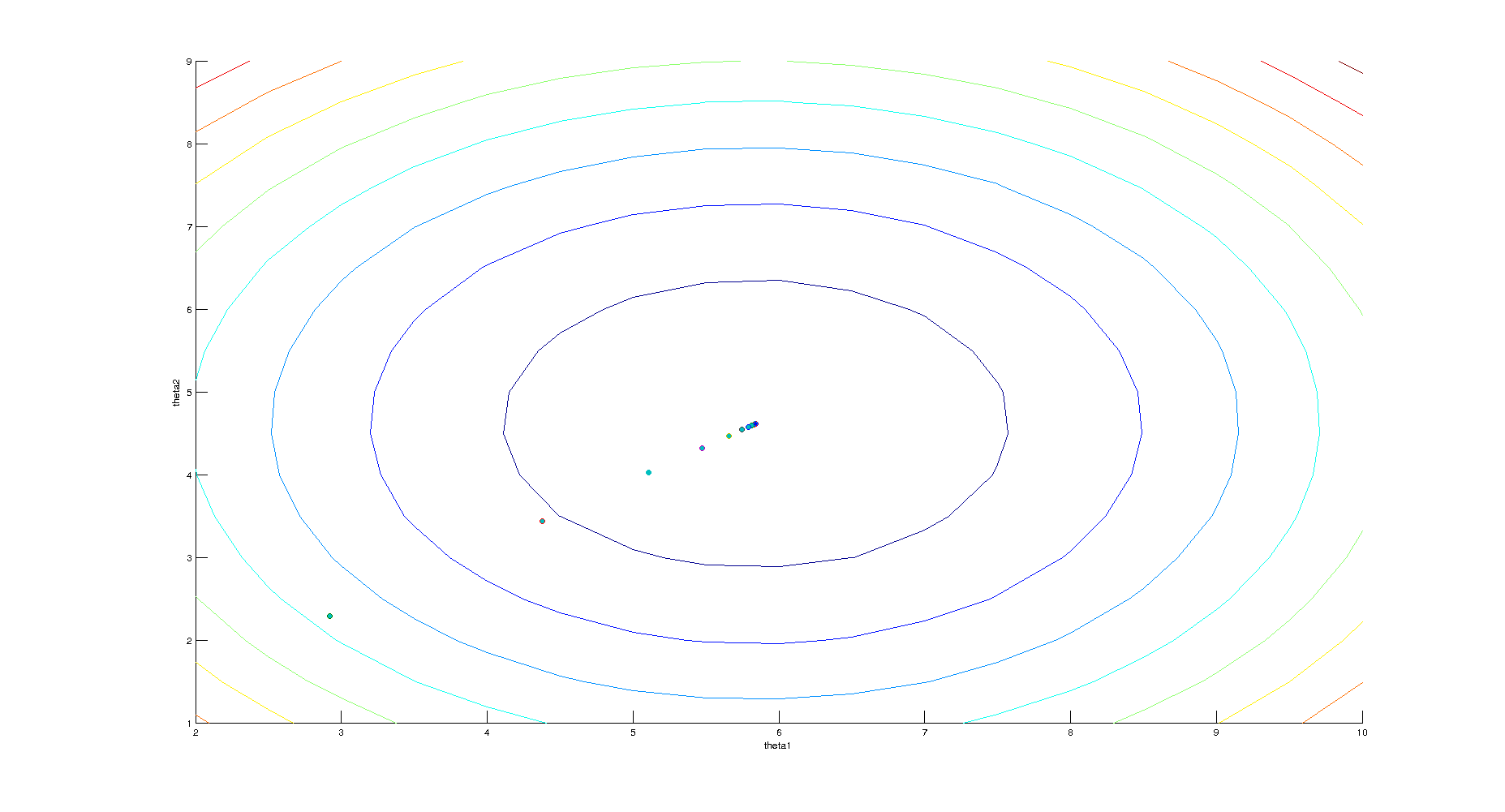
Convergence on contour plot

1. **Contour plot at different learning rates.**

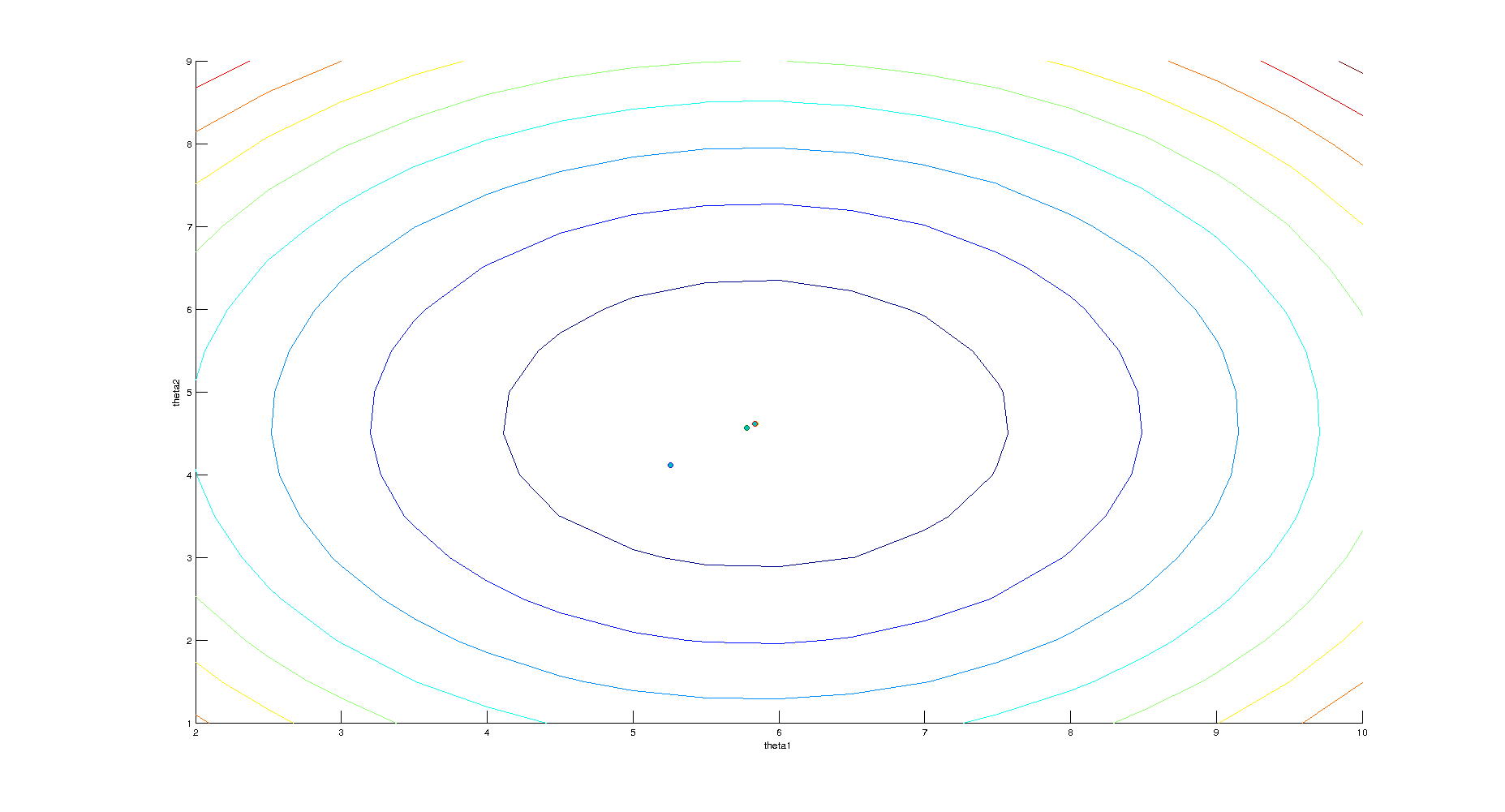
Below are the contour plot for different learning rates as asked in the question.



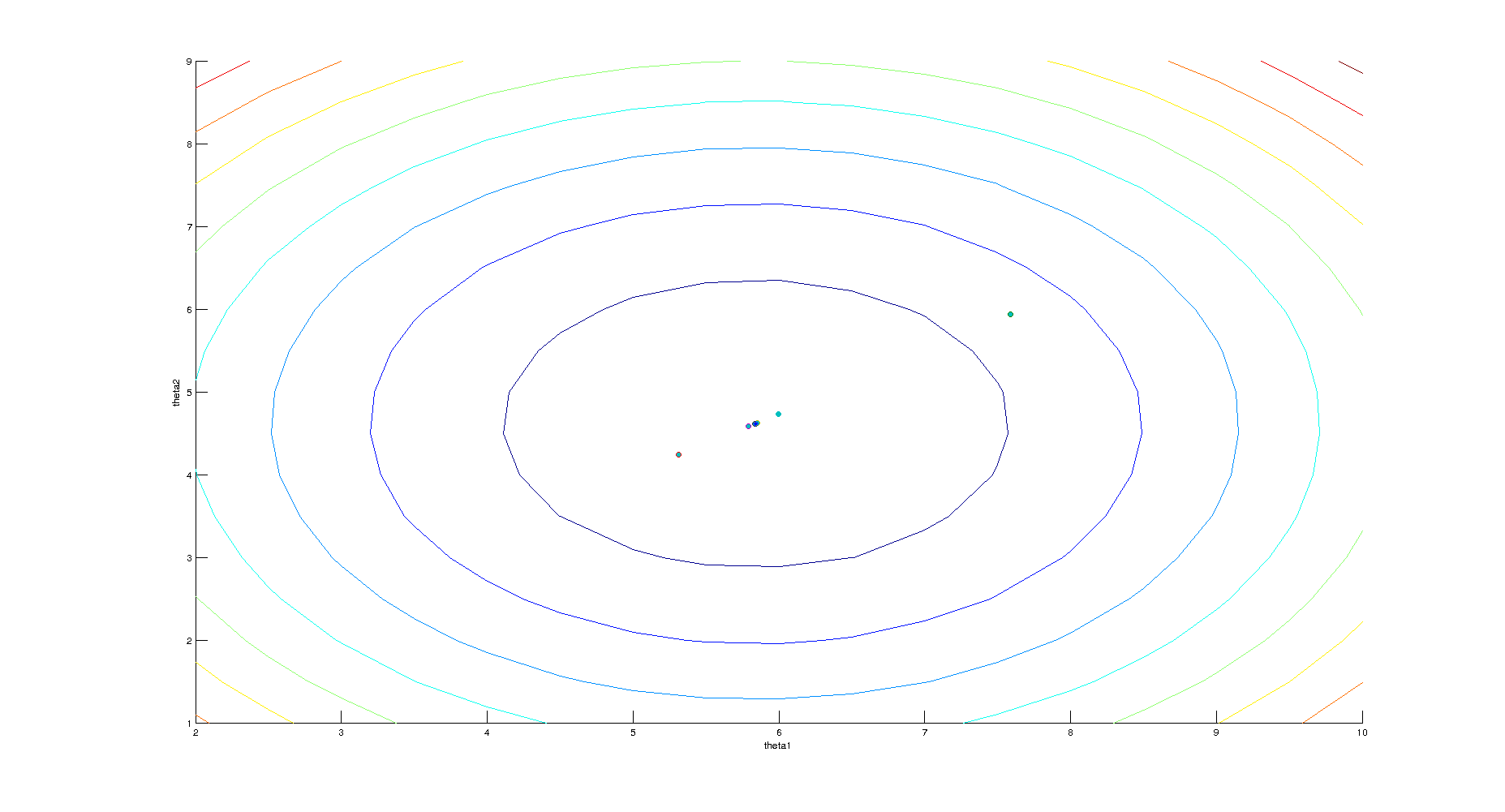
For Learning rate = 0.1



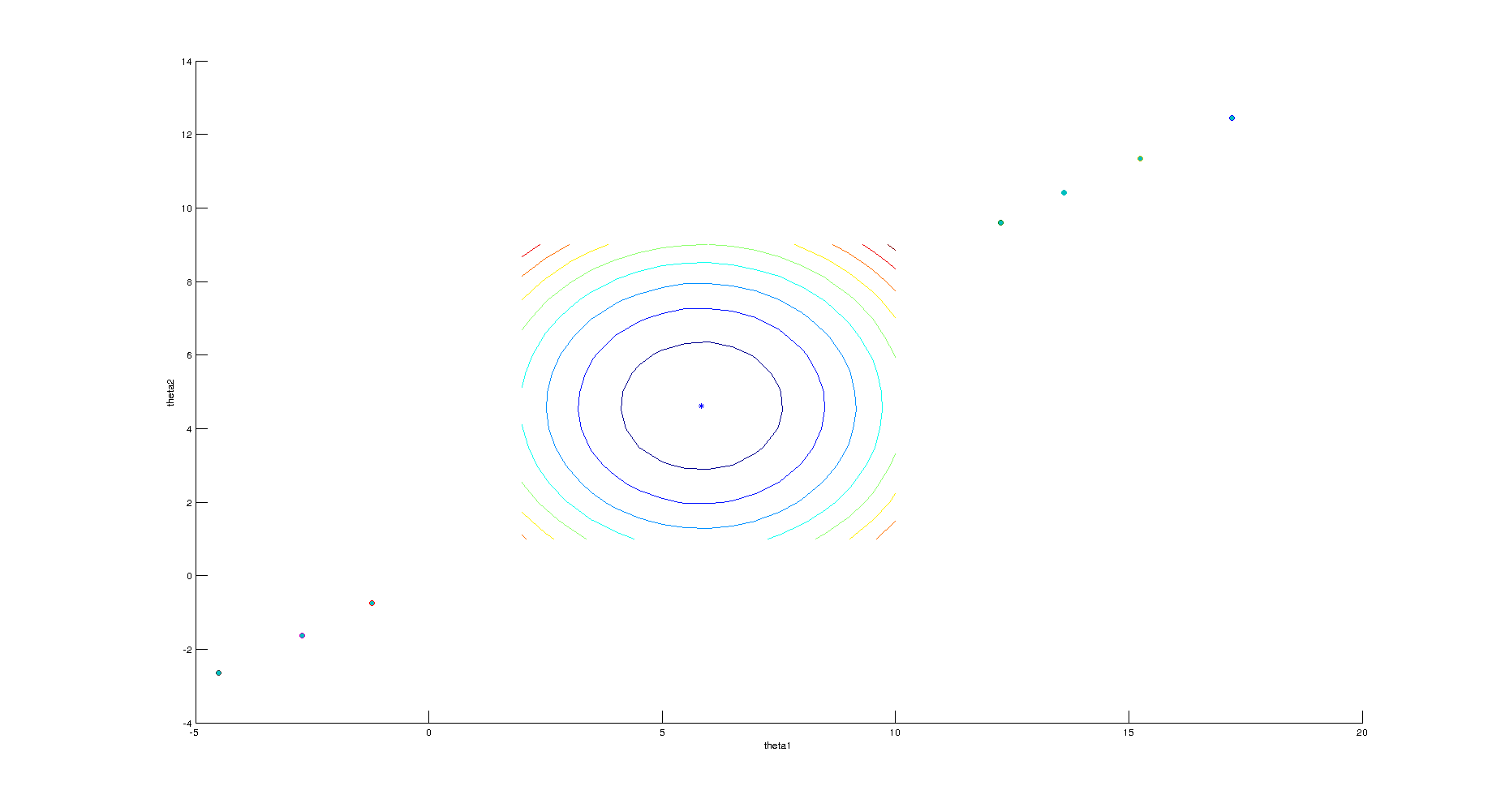
For learning rate = 0.5



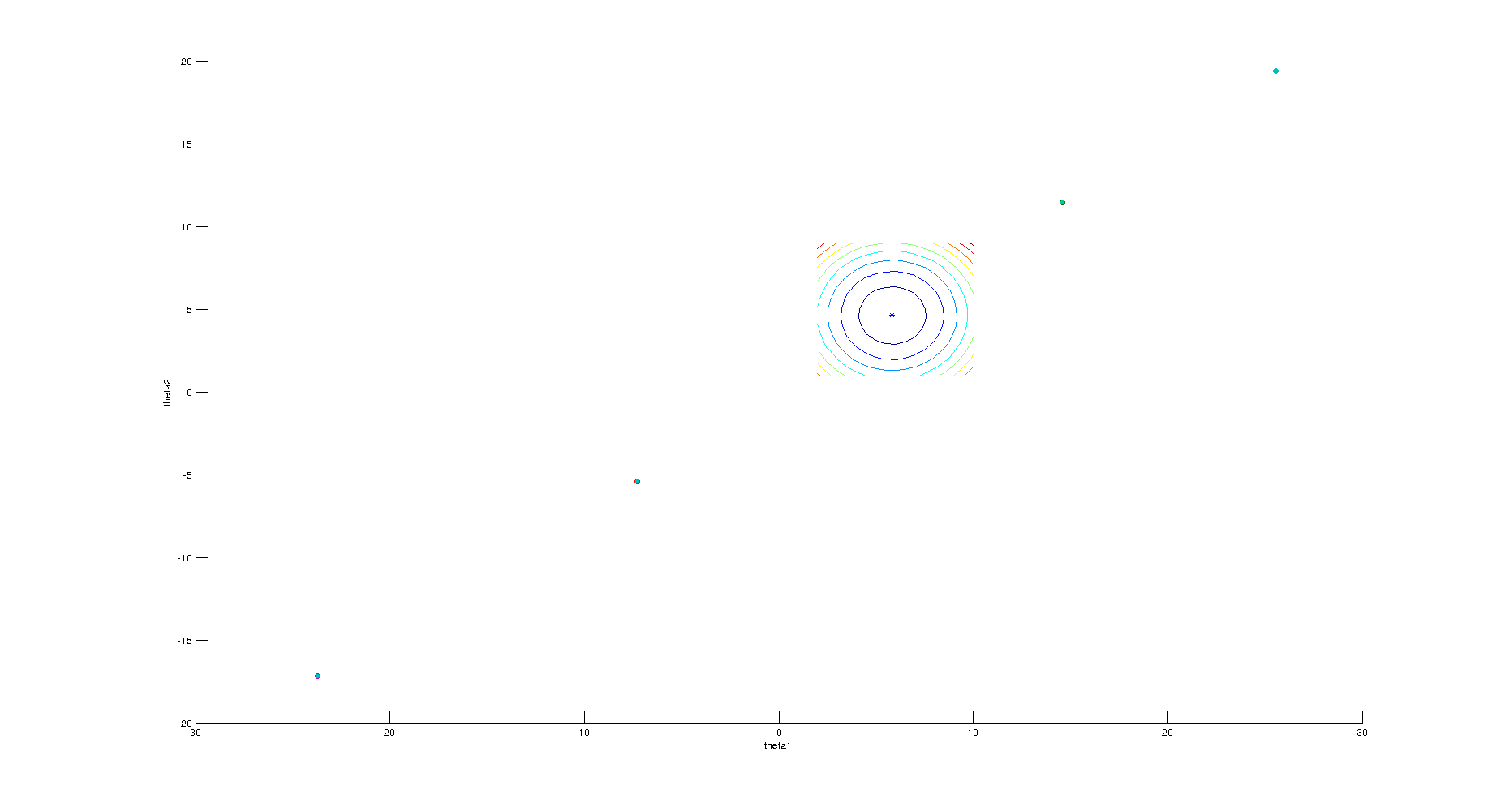
For learning rate = 0.9



For learning rate = 1.3



For learning rate = 2.1



For learning rate = 2.5

*Q2.* Linear Regression

1. **Batch Gradient Descent**:

The Batch gradient descent is implemented in function *p1(η,ε)* which takes the learning rate *η* and stopping criteria *ε* as arguments for optimizing *J(θ)*. The value of variation of number of iterations as the function of *η* for *ε=0.000001* is shown in the table below.

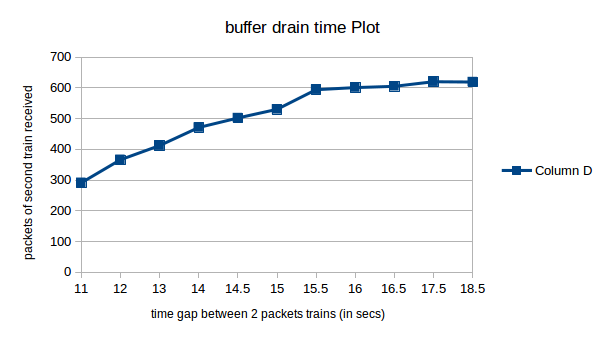
*Q 3.* Logistic Regression

1. **Newton’s method for optimizing**

The Newton’s method for optimizing is implemented in function *p1()*. The value of optimal was observed as:

*Question 2)*

1. **Methodology**:
   * To calculate the downlink buffer drain time, 2 consecutive packet trains were sent with a gap between them.
   * The gap was so adjusted that most of the packets of the second train reached back.
   * If the gap is too small then the starting packets of the second train will get lost.
   * If it happens, then the gap is increased by a small amount to find the minimum time in which most of the packets of the second train reach back.
2. **Observations**



*Figure 2: Drain estimation on IIDEA 3G cellular connection at 3:00 PM*

1. **Conclusion:**
   * Since we were able to find the buffer drain time only once ,we conclude that the average buffer drain time is around **16.5 sec**.

---------------------------------------------------------------------------------------------------------------------------

*Question 3)*

* **Throughput by buffer size and buffer drain time:**
  + The approximate throughput is calculated by

throughput = (buffer size) / (buffer drain time)

* + Throughput = 600 / 16.5 = 0.05Mbps [approximately on IDEA 3G cellular mobile]
* **UDP throghput (2)**
  + Throughput calculated by measuring the time between arrival of the first packet and the last packet in a train was calculated using the average of 20 such repetitions of such experiments and the average throughput was found to be **0.3048761 Mbps**.
* **TCP throughput**
  + The given files were downloaded using TCP sockets in python using the IDEA 3G cellular network and the throughput was measured as.

**throughput = (size of file) / (time to download)**

|  |  |  |  |
| --- | --- | --- | --- |
| files | great.txt(1.5KB) | paper-reading.pdf(76.7KB) | giving-talks.pdf(5.3MB) |
| throughput  ( in Mbits/s ) | 0.00755 | 0.22837 | 0.2899156 |

*table showing throughput of various file on TCP*

* + The throughput of file *great.txt* was the least which is due to the slow start of the TCP and the connection establishment phase. Also Since the size of the file is very small (**1.5KB**) the overall goodput for this file is very small.
  + The throughput of file *paper-reading.pdf* was the much larger than first file. In this case the overall time in downloading the whole file was comparatively larger than the connection establishment phase and TCP window slow start phase and the average throughput reached **0.22837 Mbps**.
  + The throughput of file *giving-talks.pdf* was the maximum because its size is the largest. In this case the slow start phase of TCP has negligible effect and the file is downloaded mostly in TCP saturated state.
* **Comparison between throughput of TCP and UDP:**
  + The best achievable throughput for TCP(**0.2899156 Mbps**) was found be less than the throughput of UDP(**0.3048761 Mbps**). This is due to the TCP slow start phase and SYN, ACK acknowledge packets of the TCP.
  + On the other hand UDP is connection-less and unreliable protocol and so doesn;t uses SYN, ACK packets which are reducing the overall goodput of TCP.
* **Possible improvements over TCP**
  + From our observations it seems that the performance of TCP is not good for small files. ( **0.00755 Mbps << 0.2899156 Mbps**). The improved protocol can be such that it uses TCP like strategy for big files and UDP for small files. (**<1KB**) and to maintain reliability we can send multiple copies of those small files because sending multiple copies would not decrease the throughput to such low values as in TCP.

---------------------------------------------------------------------------------------------------------------------------

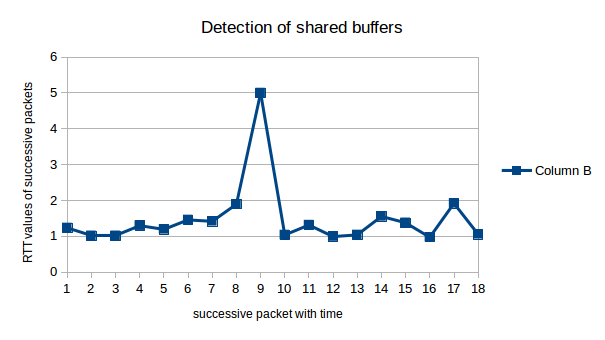
*Question 4)*

**Methodology:**

* To determine if the buffer is shared or per-source, we used the following:
  + A script was written that periodically requests the server to send a packet back and thus finds the RTT of the reply from the server.
  + This script was run from one computer and a flooding was done from another, with both the computers connected to 2 different phones and both the phones of the same network and kept in vicinity so that they are connected to the same GGSN.
  + If it is seen that the RTT increases during and just after the flooding, that would mean that the downlink buffer is shared.
  + If the RTT does not increase / change much, then that would mean the buffer is per-source.

**Observations:**

* It was observed that the RTT increased during the flooding and one packet even got timed out.
* Following was the RTT observed before, during and after the flooding.



*Figure 4 - RTT observed before, during and after the flooding (peak represents timeout)*

**Conclusion:**

* Since there is a increase in the RTT during the flooding, it can be concluded that the downlink bottleneck buffers are common.

---------------------------------------------------------------------------------------------------------------------------

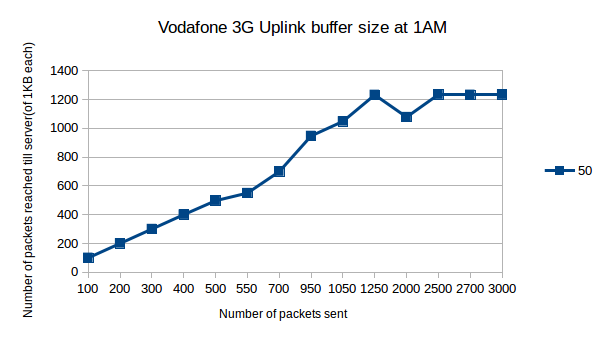
*Question 5)*

**Methodology**:

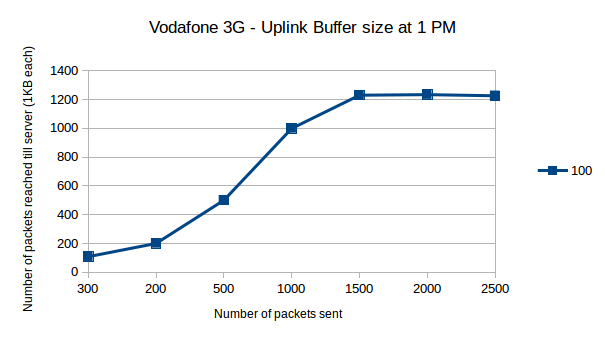
* To find where the uplink buffers exist
  + In order to find where the uplink buffers exist, a traceroute was done and correspondingly a flooding of the network was also performed.
  + If the bottleneck link bufffers actually exist at the client itself, meaning the link from my phone to the second hop is the slowest/bottleneck link,
    - Then the RTT to the first hop which is my phone will remain the same during and after the flooding.
    - The RTT to the second hop will increase as the packets will begin to queue up in the outgoing buffer of the phone itself
  + Instead, if the bottleneck link is not the link from my phone to second hop but is the link after the GGSN, then
    - the packets will not queue up in the phone, but instead they will queue up in the GGSN buffers.
    - Also, then the RTT to the second hop will remain the same and the RTT to the third hop will increase during and after the flooding.
* To find the uplink buffer size
  + In order to find the uplink buffer size, the uplink was flooded with ECHO packets of 1KB size [with the server requiring to send 0 size message].
  + This will effectively throttle the uplink.
  + Some time after sending the packets, STAT packets / messages are used to find which of the packets reached the server.
  + If through the STAT we get to know that the packets are effectively getting dropped after a certain number, then this would imply the buffer size.
* Drain time
  + Similar to finding the downlink drain time, the uplink drain time was found:
    - A train of packets was sent and another one after a certain α amount of time.
    - Each of the packets sent in the previous 2 trains were of 1KB size and required the server to not reply.
    - After sending the 2 trains, STAT messages were used to find which packets of the 2 trains actually reached the server.
    - The value of α was so adjusted all the packets of the second train reached the server.
* Shared or per-source
  + First we determine where the uplink buffers exist.
  + Then we do a traceroute to glenstorm and correspondingly flood the network with large packets from some other source i.e. mobile.
  + 2 cases are possible:
    - If the bottleneck buffer is at the client itself
      * Then if the buffer is shared, the traceroute during and after the flooding will show that the RTT to the second hop as increasing.
      * If the buffer is per-source, the traceroute during flooding will not show increase in RTT.
    - If the bottleneck is at the GGSN,
      * Then if the buffer is shared , the tracroute will show an increase in RTT for the third hop.
      * If the buffer is per-source, the traceroute during flooding will not show increase in RTT.

**Observations And Conclusions:**

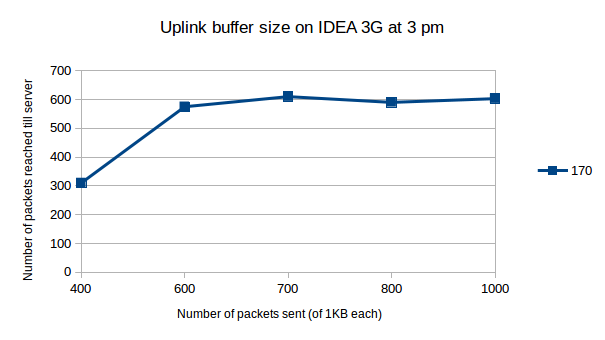
* To find where the uplink buffers exist:
  + It was seen in case of VODAFONE connection, that the uplink buffers exist at the phone itself.
    - When the test was being done, the connection was EDGE.
    - So, on doing traceroute the RTT of the packets got timed out on the second hop.
    - Also, the RTT to the first hop - the phone remained unchanged and the RTT to the second hop increased during flooding.
    - We can conclude that the packets were getting queued in the outgoing buffer of the phone and this is Ok since the connection then was of EDGE which gives a lower bandwidth, meaning the packets were not being able to be writtten on the wireless link as quickly as they were being sent.
  + Also, this explains why the uplink buffer size obtained below is around 1200 KB.
  + It is high because it is the buffer of the phone that is the bottleneck buffer and we can expect the phone to give a buffer size of around 1.2 MB to the connection.
* To find the uplink buffer size
  + The uplink buffer size obtained is described as in the graph below:



*Figure 5(a) -Uplink buffer size on VODAFONE 3G at 1 AM*



*Figure 5(b) -Uplink buffer size on VODAFONE 3G at 1 PM*



*Figure 5(c) -Uplink buffer size on IDEA 3G at 3 PM*

* From the above plots, it can be seen that the uplink buffer size on VODAFONE 3G connection is around 1200 KB and on IDEA 3G is around 600 KB.
* As already explained in the last subsection, the high value of buffer size in case of VODAFONE is because the uplink buffers exist at the client, i.e. the phone itself. The phone is not able is write out the packets as quickly on the link and so the packets are getting queued at the phone itself. The phone must be providing a maximum buffer size of around 1200 KB.

Shared or per-source

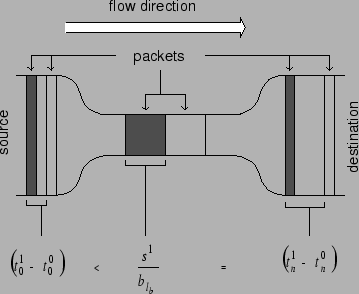
* It was found that already that the uplink buffer exists at the phone itself. So, the problem then reduces to finding whether the phone allocates common or per-source buffers to its connected devices.
* Intuitively since the phone is a limited memory device, so it must be allocating its memory cautiously and so the buffer should be shared.
* Indeed, on connecting 2 computers to the same phone through which the test was being done, it was found the RTT from one computer increased when the other one flooded the network.

---------------------------------------------------------------------------------------------------------------------------

*Question 6)*

**Methodology**:

* Packet pair techniques:
  + If we know the capacity of a path then using the inter-arrival time at the receiver and the inter-dispatch time at the sender, we can infer the available bandwidth as follow.



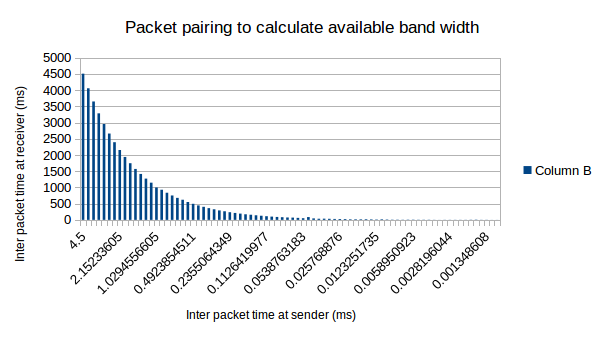
*Figure 6*

* + The relation between the inter-arrival time of packets at the receiver and the sender and the capacity of network is given by

*t*1n - *t*0n = max( *s*2 */* b , *t*10 - *t*00*)*

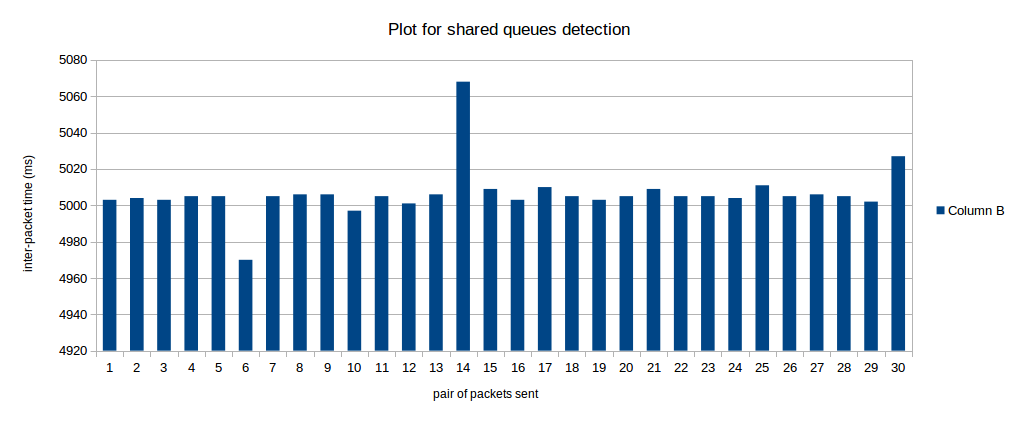
*where b is the capacity of the bottleneck buffer and s*2 *is the 2nd packet size*

* observations
  + The value of b was cross-checked using *ookla speed test website .*
  + So we just kept sending the two packets with decreasing separation in time, when inter-packet time at sender was large i.e *t*10 - *t*00 *> s*2 */* b then inter-packet time at receiver will vary linearly with inter-packet time at sender but for low values of inter-packet time at sender the inter-packet time at receiver became constant and this constant helped in calculating the available bandwidth and the value of capacity of the channel provides a lower bound for the inter-packet time by the sender.



bandwidth estimation by decreasing the inter-packet time

* + The limiting value of inter-packet time was found to be around 2-3 ms for IDEA 3G cellular network which gave the available bandwidth 3.72 MBps and with *ookla speed test* available bandwidth was around 4.47 MBps which is close to calculated value.
* Shared queues detection
  + It was observed that there was not much variation of inter-packet time at client for the same inter-packet time at receiver on IDEA 3G cellular connection . By this we may infer about the absence of shared queues .



variation of inter-packet time ( in ms)