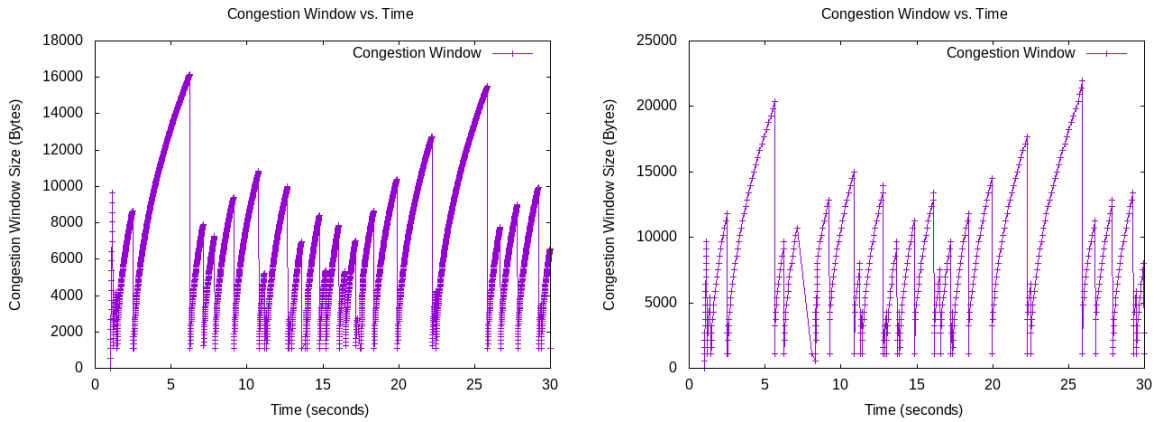
COL334 Assignment 3

Rayyan Shahid (2019CS10392)

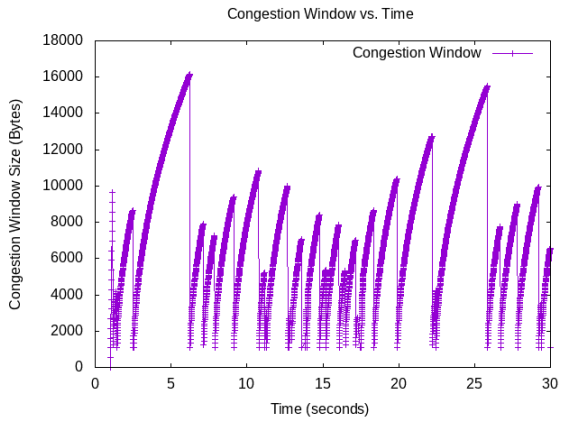
October 2021

1 Part 1

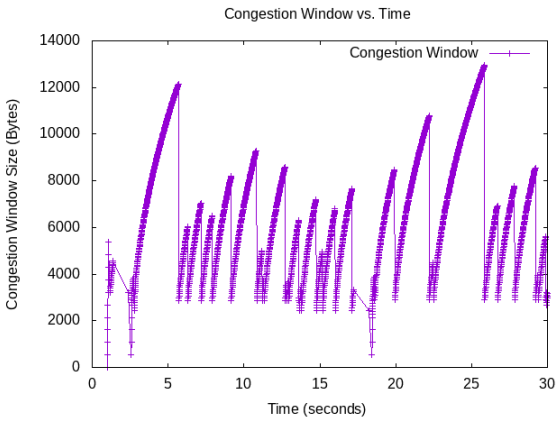


(a) Newreno

(b) Highspeed



(c) Veno



(d) Vegas

Figure 1: Congestion window size vs time plots for different TCP congestion control protocols

1.1

Protocol	Dropped Packets
Newreno	38
Highspeed	38
Veno	38
Vegas	39

1.2

For each of the protocols, the number of dropped packets is almost the same. Packets are dropped because of the `ReceiveErrorModel` set at the sink. The number of packets dropped are proportional to the number of packets transmitted. Because of same channel data rate, application data rate and propagation delay and similar performance of different protocols, the number of packets transmitted are almost the same in each case and as a result number of dropped packets is also the same.

1.3

1.3.1 Newreno

1. It is a loss based algorithm with additive-increase and multiplicative-decrease to control congestion window size.

2. During slow start, congestion window is incremented by at most SMSS(Sender Maximum Segment Size) bytes for each ACK received that cumulatively acknowledges new data. Slow start ends when cwnd exceeds ssthresh(slow start threshold) or when congestion is observed.

   $$cwnd+ = min(N, SMSS)$$

   Here N is the number of new bytes acknowledged in the incoming ACK

3. During congestion avoidance, cwnd is incremented by roughly 1 full-sized segment per round-trip time (RTT), and for each congestion event, the slow start threshold is halved.

1.3.2 Highspeed

1. It is modification of TCP Reno algorithm designed for connections with large congestion windows.

2. Similar to Reno, it is a loss based algorithm with additive-increase and multiplicative-decrease phases.

3. Because it is designed for larger congestion windows, the change in size of congestion window is also higher at every step, as observed by the sparse points in the plot. It also reaches higher peaks of congestion window size.

4. Performance is similar to Newreno and same number of dropped packets are observed.

1.3.3 Veno

1. It builds on the Reno algorithm and improves the performance of TCP in wireless links.

2. It effectively deals with random packet loss in wireless access networks by distinguishes between congestive and non-congestive states.

3. As our topology was a point to point physical link with only a `ReceiveErrorModel` set at the sink, therefore no significant differences were observed between Veno and Newreno and same number of dropped packets were observed for both.

1.3.4 Vegas

1. This is different from Reno as this is a packet delay based algorithm which detects congestion in the connection based on increasing RTT values as opposed to packet drops. As a result, it is able to detect congestion earlier and adjusts the congestion window size accordingly as observed in the plot.

2. To avoid congestion, Vegas linearly increases/decreases its congestion window to ensure the diff value falls between the two predefined thresholds, alpha and beta. As a result, the range of values of congestion window size was observed to be smaller for Vegas. It has lower peaks and higher lows as compared to the other protocols.

3. Its performance was more or less similar to other protocols with one higher count of observed dropped packets.

2 Part 2

2.1

1. The plots obtained for all the 5 channel data rates appears very similar. However, on close inspection, it is observed that plots for higher channel data rates are slightly compressed in the time axis.

2. As the channel data rate increases from 2Mpbs to 50Mbps, the number of packets dropped increases. This increase reduces for larger channel data rates.

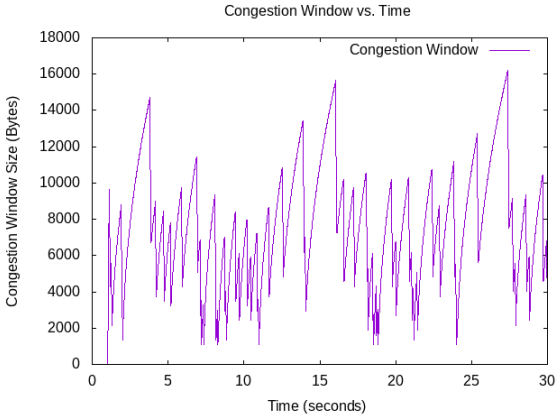
3. For all the cases, the channel data rate is larger than application data rate and therefore the dependence of the observed RTT on the channel data rate is small. This dependence becomes even smaller as the channel data rate increases (because of inverse dependence of channel data rate on RTT)

4. Because of smaller RTT, the frequency of updates on congestion window are higher which explains the compression observed in time axis, and as the result number of packets transmitted are also larger (higher effective data transmission rate). Since number of packets dropped in proportional to transmitted packets, we see a slight increase in dropped packets with increasing channel date rate.

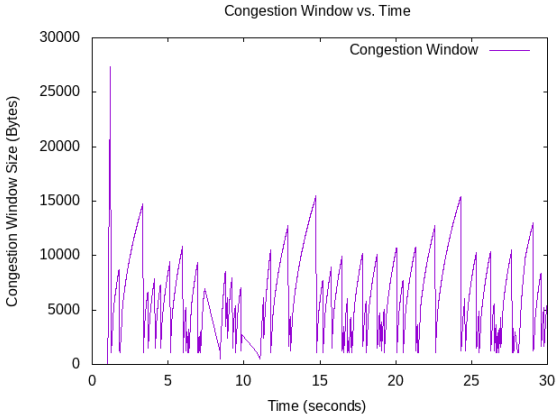
2.2

1. As the application data rate is increased from 0.5Mbps till 4mbps, we observer a large increase in number of dropped packets. This observation can be attributed to the fact that for smaller values of application data rate (as compared to channel data rate), the effective data rate is nearly proportional to the application data rate. As a result, the number of dropped packets also becomes nearly linearly dependent on application data rate.

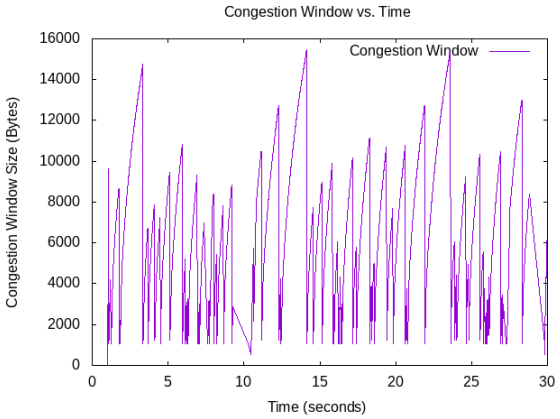
2. For larger values of application data rate, the channel data rate also significantly affects the observed RTT value (and as a result effective data rate) and for even larger values of application data rate, RTT and effective data rate depends almost entirely on channel data rate. This is reason we see similar plots and same number of dropped packets for application data rate 4Mbps and 10Mbps.



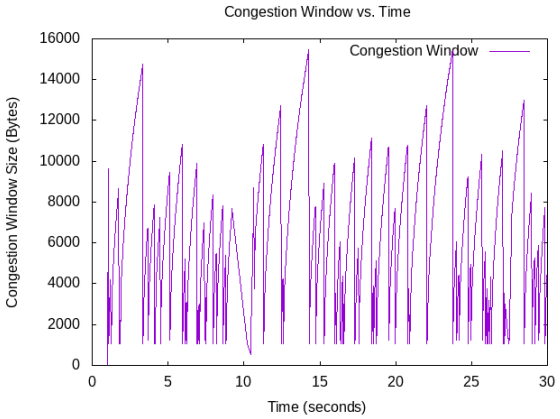
(a) 2Mbps Channel Data rate



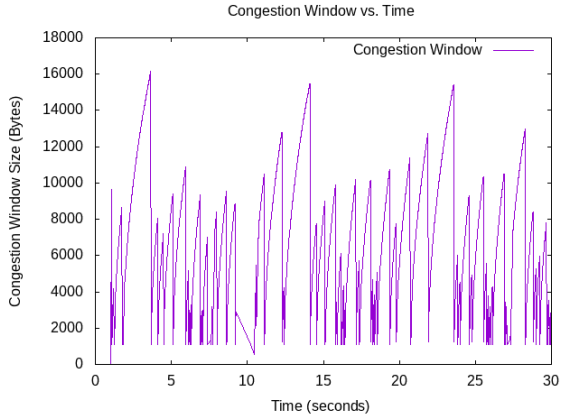
(b) 4Mbps Channel Data rate



(c) 10Mbps Channel Data rate



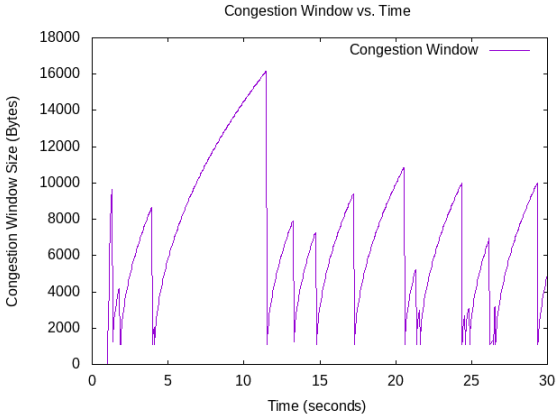
(d) 20Mbps Channel Data rate



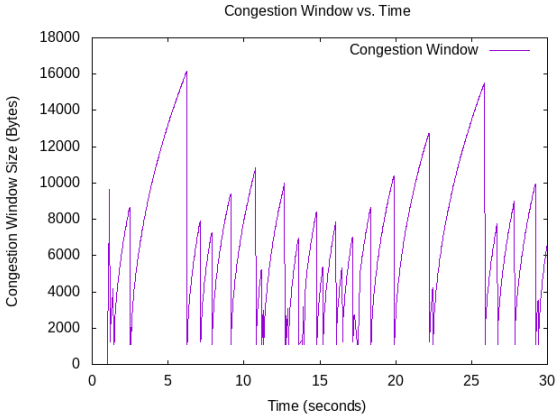
(e) 50Mbps Channel Data rate

Figure 2: Congestion window size vs time plots for varying channel data rates

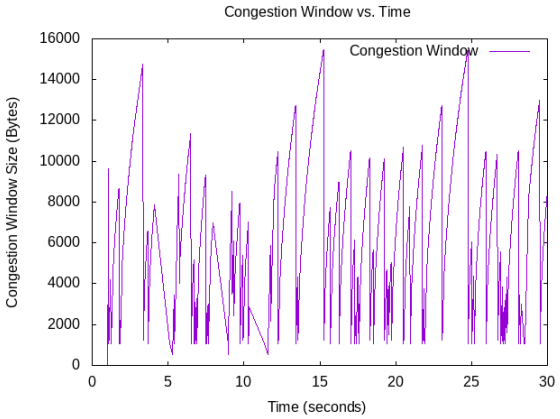
Channel Data Rate	Dropped Packets
2Mbps	62
4Mbps	72
10Mbps	73
20Mbps	74
50Mbps	75



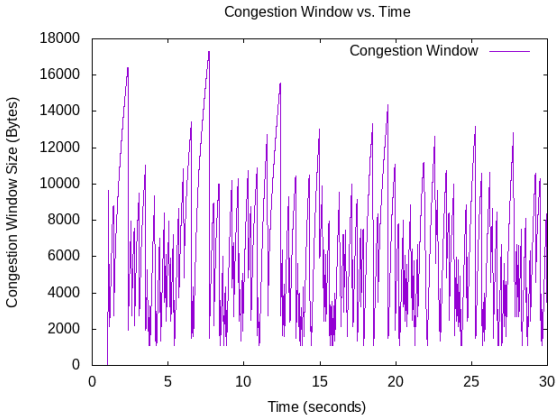
(a) 0.5Mbps Application Data rate



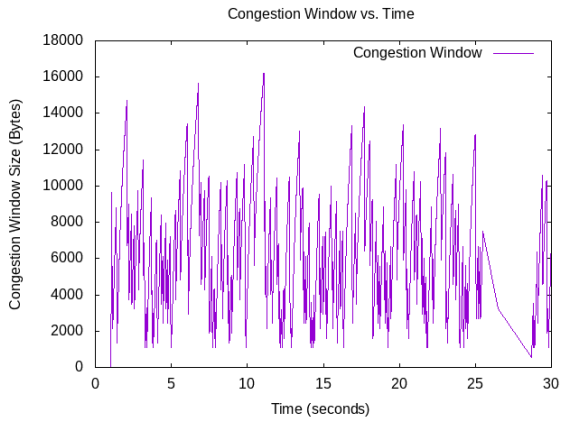
(b) 1Mbps Application Data rate



(c) 2Mbps Application Data rate



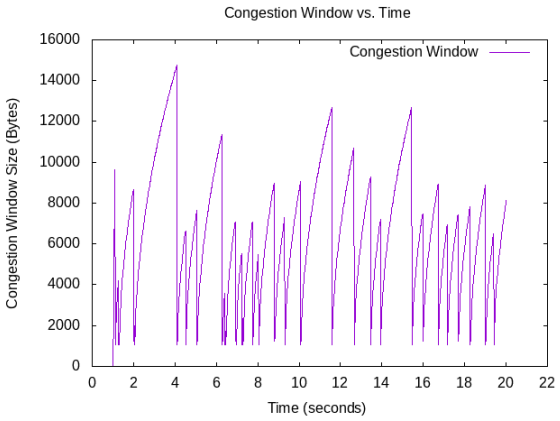
(d) 4Mbps Application Data rate



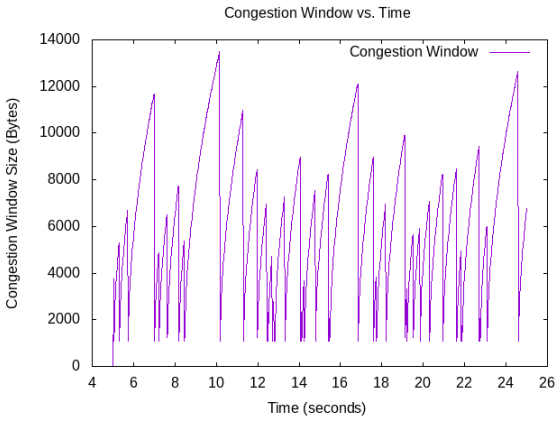
(e) 10Mbps Application Data rate

Figure 3: Congestion window size vs time plots for varying application data rates

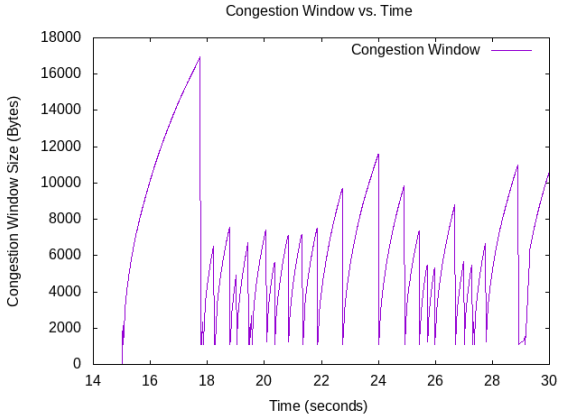
Application Data Rate	Dropped Packets
2Mbps	22
4Mbps	38
10Mbps	71
20Mbps	156
50Mbps	156



(a) Connection 1



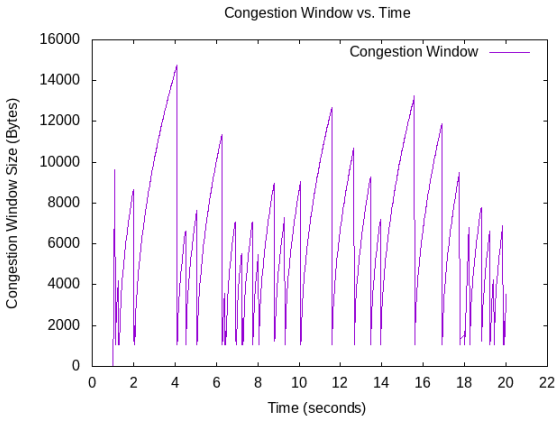
(b) Connection 2



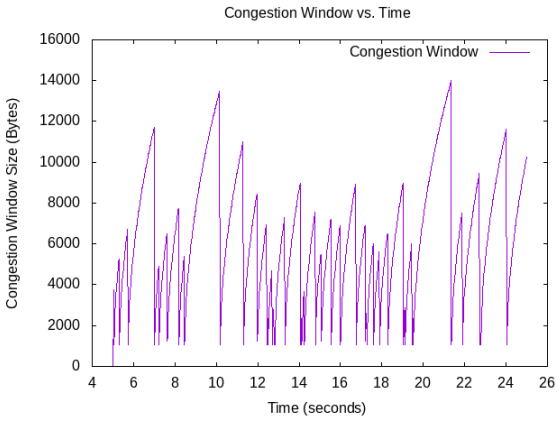
(c) Connection 3

Figure 4: Congestion window size vs time plots for configuration 1

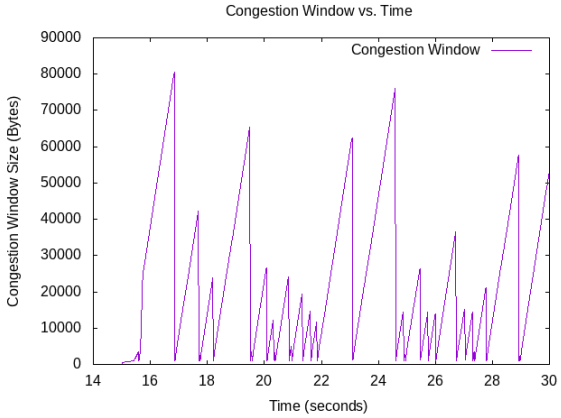
P2P link	Dropped Packets
N1-N3	76
N2-N3	30



(a) Connection 1



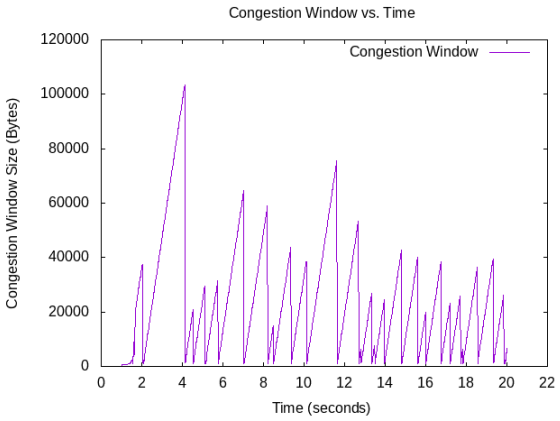
(b) Connection 2



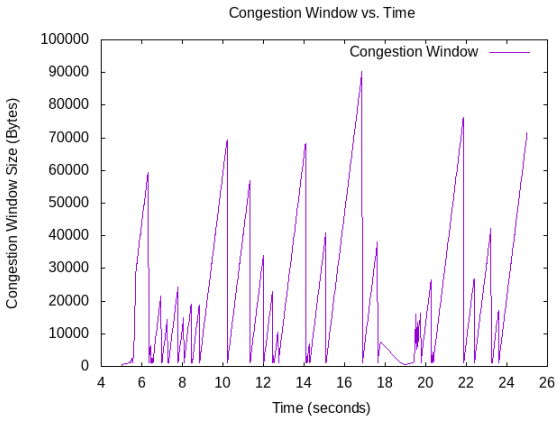
(c) Connection 3

Figure 5: Congestion window size vs time plots for configuration 2

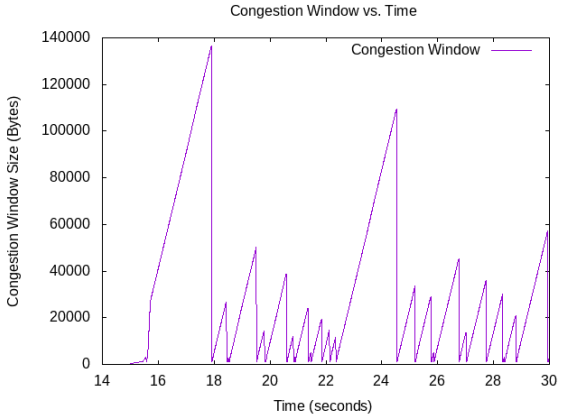
P2P link	Dropped Packets
N1-N3	80
N2-N3	30



(a) Connection 1



(b) Connection 2



(c) Connection 3

Figure 6: Congestion window size vs time plots for configuration 3

P2P link	Dropped Packets
N1-N3	78
N2-N3	27

3

3.1

It is observed that the number of packets dropped are almost the same for each link in all the configurations. The reason behind this could be that even though the variation in congestion window size is different in all the configurations, yet the effective data transmission rate for each link remains constant over the configurations and therefore the number of packets transmitted and as a result number of packets dropped remains more or less the same.

3.2

- For `TCPNewReno`, the congestion window grows logarithmically with time in the congestion avoidance phase (as increment in *cwnd* is inversely proportional to *cwnd*). As a result, the congestion window grows slowly with time in `TCPNewReno`. However, in `TCPNewRenoCSE`, the congestion window grows linearly with time in the congestion avoidance phase (as increment in *cwnd* is a constant). As a result, the congestion window increases much more rapidly as compared to `TCPNewReno`. As a result, very high peaks in congestion window size are observed.
- The connections 1, 2 and connection 3 are setup on different channels and hence we might expect the connections to be independent of each other. However, as they have the same sink node N3, so in those cases where the congestion window size is very large, it might cause large load at sink (probable reason could be limited size of buffer at sink) and as a result the observed RTT for the other connections might be much larger (might even result in timeout before ACK is received) resulting in smaller congestion window sizes for those connections.
- The above situation is observed in connection 2 in configuration 2, after the connection 3 is established at 15s, there is high peak for connection 3 (as a result of using `TCPNewRenoCSE`), and therefore it results in a smaller peak in connection 2, which was present in configuration 1.

4 Included Files

1. `Report.pdf`
2. `First.cc`
3. `Second.cc`
4. `Third.cc`
5. `TcpNewRenoCSE.cc`
6. `TcpNewRenoCSE.h`
7. `readme.md` - Contains the execution instructions for running the programs and generating plots.