Charvi Virani

UIN: 662581137

DCS PROGRAMMING PROJECT REPORT

Experiment Part 1

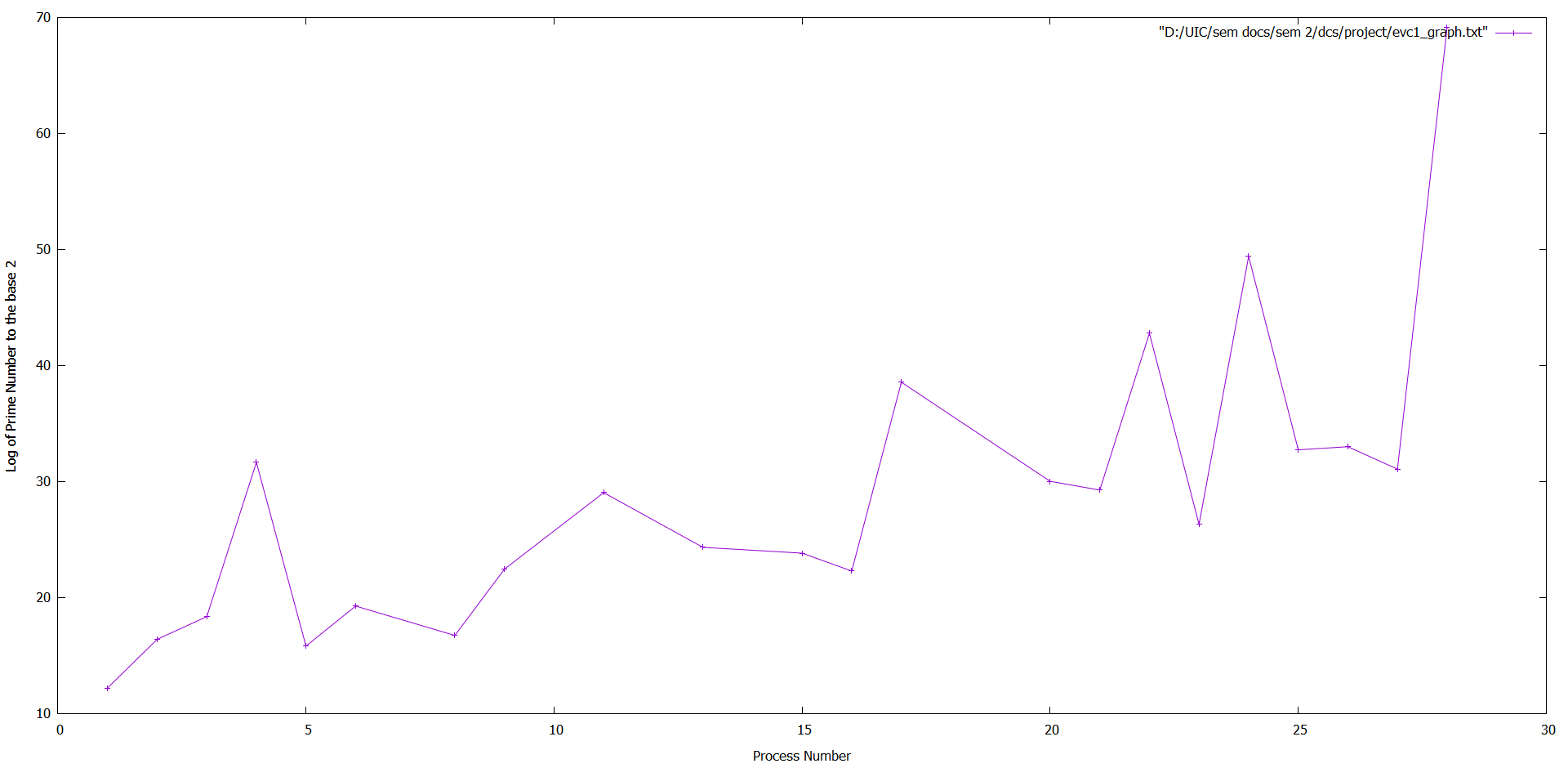
A goal of the project is to identify how fast the EVC grows, as a function of the number of events executed by a process, and the number of events executed by all the processes collectively. n is an input parameter. With n processes and assuming 32-bit integer, how many events it takes for the size of the EVC to occupy a number equal to 32n bits long. Once it equals 32n bits long, we can do a system-wide EVC reset. (Repeat the experiment assuming 64-bit integer instead of 32-bit.)

The below table helps in plotting the size of EVC in bits as a function of the number of events executed in the system. ‘n’ is an input parameter. With n = 30 processes, the below table shows the ‘Process Number’ (N) in the first column, the ‘Prime Number’ associated with the corresponding process number in the second column and the ‘Log of the prime number to the base 2’ of the corresponding process number in the third column.

|  |  |  |
| --- | --- | --- |
| Process Number | Prime number | Log base 2 |
| 1 | 4761 | 12.21704891355634 |
| 2 | 87025 | 16.40914228849841 |
| 3 | 337561 | 18.36478870680906 |
| 4 | 3390125123 | 31.65869137543058 |
| 5 | 57967 | 15.822944202454996 |
| 6 | 634933 | 19.276244836923958 |
| 8 | 109503 | 16.740610869613352 |
| 9 | 5768419 | 22.4597445305859 |
| 11 | 555497761 | 29.0492058557705 |
| 13 | 21169201 | 24.335463482206492 |
| 15 | 14969161 | 23.835490026886436 |
| 16 | 5134475 | 22.29178533786025 |
| 17 | 407821454881 | 38.569146719155256 |
| 20 | 1092748753 | 30.02531458576645 |
| 21 | 647959343 | 29.27132805138712 |
| 22 | 7597270400643 | 42.78861830894097 |
| 23 | 84033889 | 26.324467916299234 |
| 24 | 750562130239321 | 49.41496482902072 |
| 25 | 7191388769 | 32.743623258402785 |
| 26 | 8655490567 | 33.010968443482454 |
| 27 | 2265104507 | 31.07693046860764 |
| 28 | 650972292864256214473 | 69.1411580374546 |

By plotting a line graph of the above tabulated data, we can identify how fast the EVC grows, as a function of the number of events executed by a process, and the number of events executed by all the processes collectively.

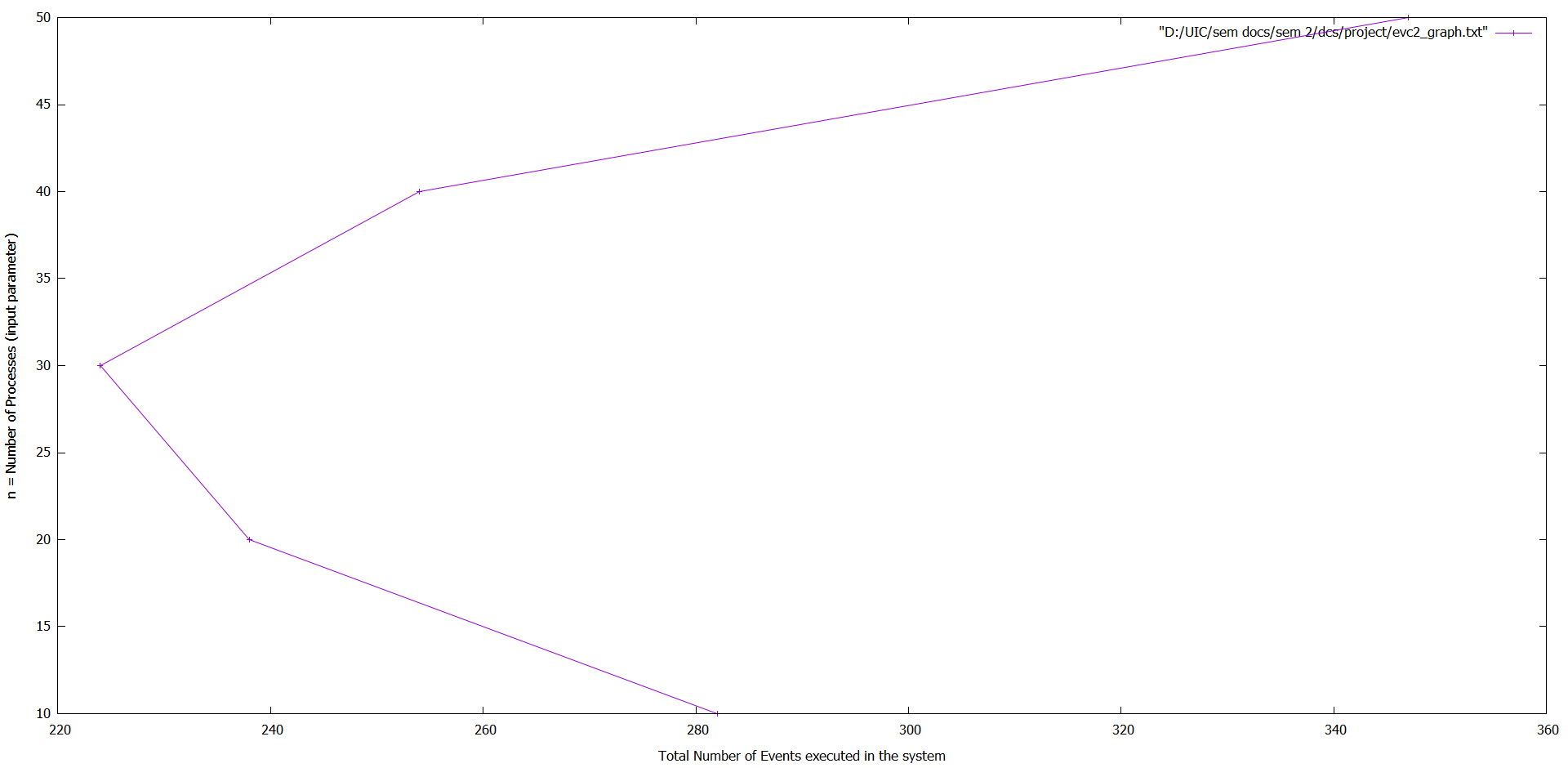
This graph is as shown below:



ANALYSIS: As seen from above graph, as the number of processes in the system increases, the logarithmic value (Y-axis) also increases. Thus, we can infer that the increase of number of processes also increases the EVC size

Now, we tabulate and then plot the (minimum) number of events executed per process, and the total number of events executed in the system, until the EVC size reaches 32n bits long, as a function of n = number of processes in the system. Again, ‘n’ is input parameter.

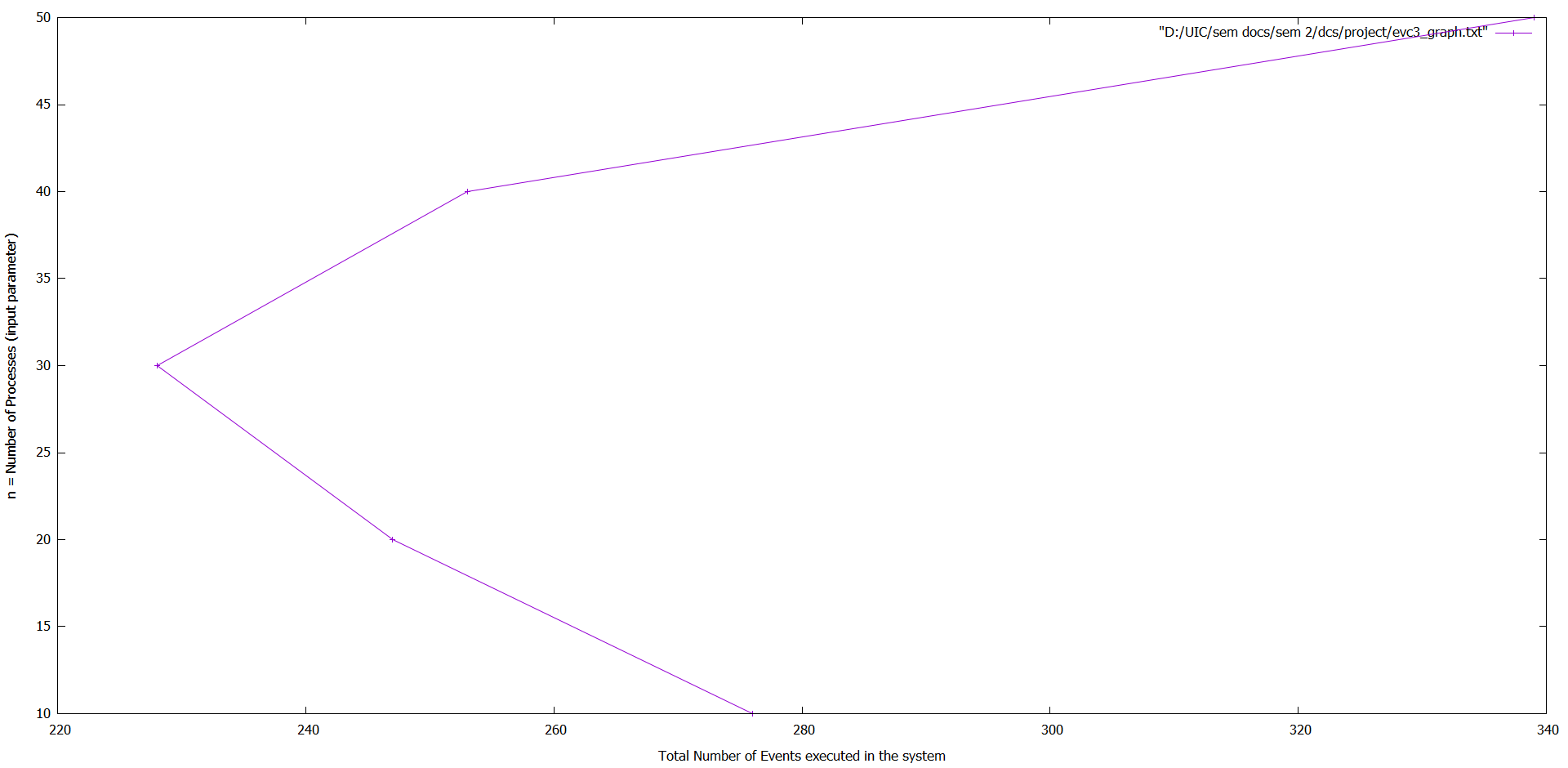
|  |  |  |
| --- | --- | --- |
| (min) number of events | Total number of events | n |
| 9 | 282 | 10 |
| 18 | 238 | 20 |
| 13 | 224 | 30 |
| 14 | 254 | 40 |
| 47 | 347 | 50 |



Repeat the above experiment assuming size is 64n bits long.

Thus, we tabulate and then plot the (minimum) number of events executed per process, and the total number of events executed in the system, until the EVC size reaches 64n bits long, as a function of n = number of processes in the system. Again, ‘n’ is input parameter.

|  |  |  |
| --- | --- | --- |
| (min) number of events | Total number of events | n |
| 11 | 256 | 10 |
| 7 | 237 | 20 |
| 15 | 268 | 30 |
| 18 | 243 | 40 |
| 21 | 339 | 50 |



ANALYSIS: By observing the above 2 graphs for 32-bit and 64-bit, we can conclude that as the number of processes (n) increases, the total number of events decreases at first and then increases again at approximately the same rate.

***Operation of EVC ti at process Pi***

* Initialize ti =1
* Before an internal event happens at process Pi, ti = ti ∗ pi (local tick).
* Before process Pi sends a message, it first executes ti = ti ∗ pi (local tick), then it sends the message piggybacked with ti .
* When process Pi receives a message piggybacked with timestamp s, it executes ti = LCM(s,ti) (merge); ti = ti ∗ pi (local tick) before delivering the message

**Internal v/s Communication events**

Internal v/s Communication events are represented by local ticks and shared ticks.

Local Tick: Whenever the logical time advances locally at Pi, the local component of the vector clock needs to tick. This increases the local component in the vector by 1: V [i] = V [i] + 1 While using EVC, this operation is equivalent to multiplying the EVC timestamp by the local prime number pi , Enc(V ) = Enc(V ) ∗ pi

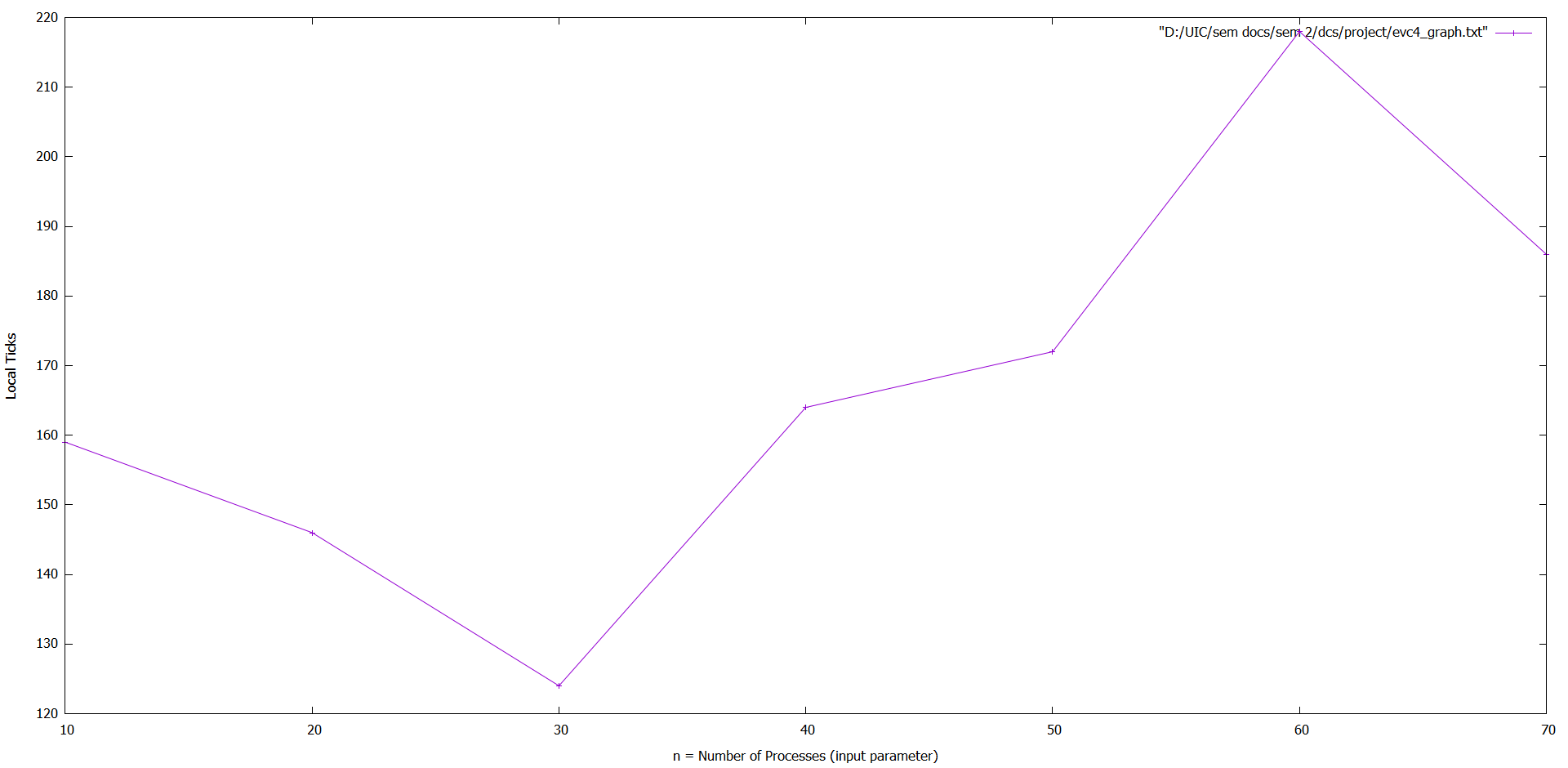
Now, we vary the above graphs by varying the mix percentage of internal events (baseline case is with no internal events) and communication events.

The data observed is tabulated below. ‘n’ is taken as input and it is the number of processes to be executed by the system.

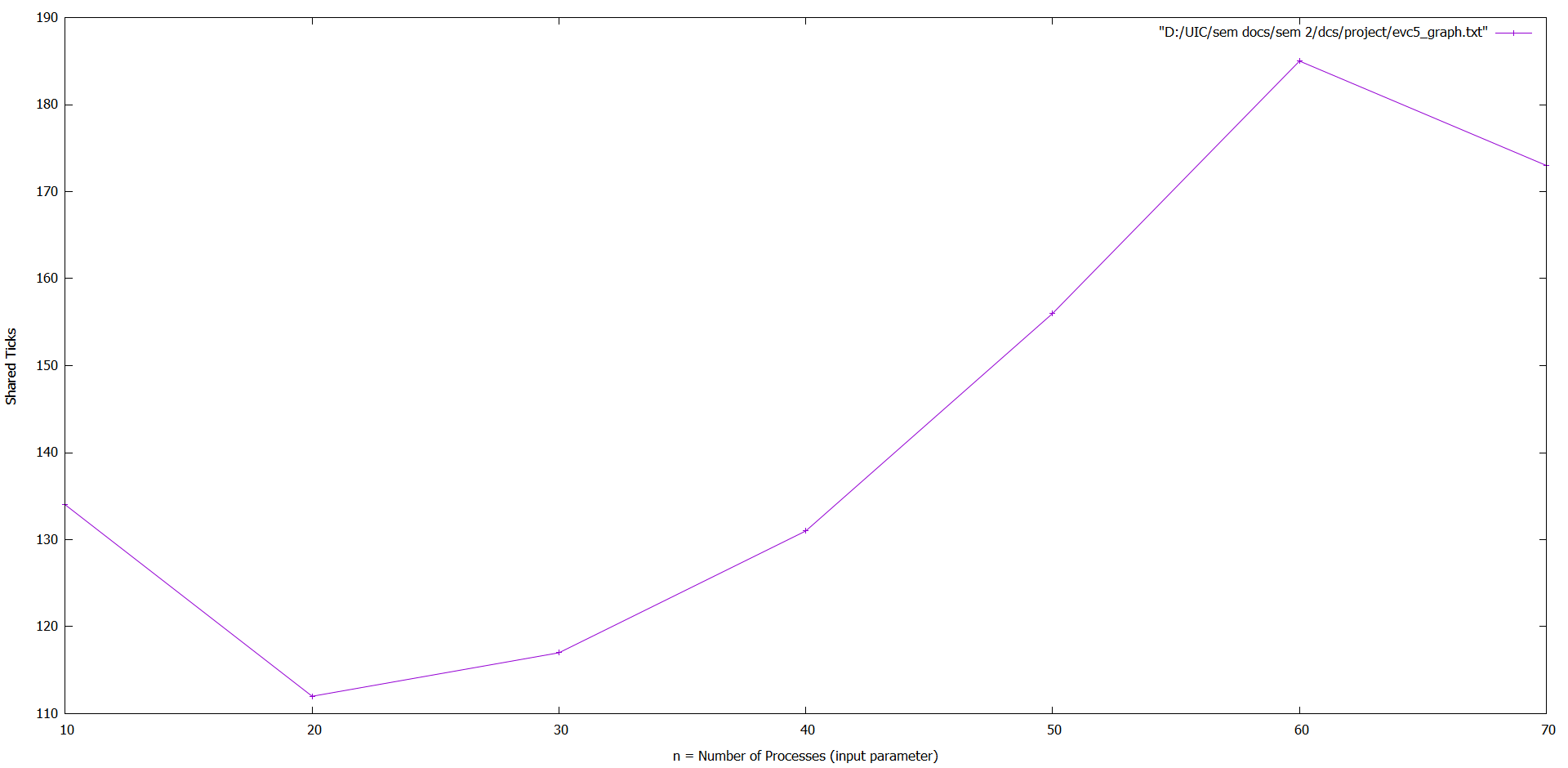
|  |  |  |
| --- | --- | --- |
| N | Local Tick | Shared Tick |
| 10 | 159 | 134 |
| 20 | 146 | 112 |
| 30 | 124 | 117 |
| 40 | 164 | 131 |
| 50 | 172 | 156 |
| 60 | 218 | 185 |
| 70 | 186 | 173 |

Thus, we plot and get 2 graphs

1. The ‘n = number of processes’ v/s ‘Local Ticks’ graph



1. The ‘n = number of processes’ v/s ‘Shared Ticks’ graph



ANALYSIS: The above two graphs show similar trends. As the ‘number of processes (n)’ increases, the number of shared and local ticks decrease at first and then increase again at approximately the same rate. This decrease and then increase in number of shared and local ticks seems to be a continuing pattern as the ‘number of processes (n)’ keeps increases.

Experiment Part 2

To understand the benefits and limitations of storing and transmitting the logarithms of the EVCs (instead of the EVCs themselves). Due to finite-precision arithmetic, round-off errors may get introduced. Such errors may cause inaccuracies in the comparison test between the EVCs of two events. (Recall the comparison test, for events e and f, e ◊ f if essentially EVC(f) mod EVC(e) = 0; now implement this using logarithms as shown in the paper, Section 4.4.) An important part of understanding the limitations is being able to quantify the level of accuracy (or its complement, the error rate introduced) in the use of logarithms and anti-logarithms.

We need to quantify the errors introduced by the rounding-offs in using finite-precision mantissas in the logarithms of the EVCs.

Below, the tables present data about the number and percentages of false positives and false negatives

Distributed executions for n = 10, 20, 30, 40, 50 have been generated.

v=50, base value (b) = 2, m = 32, 64, 128, 512, 1024 are set.

The percentage rate of errors (false positives and false negatives), for various n and m is calculated.

***Error Rate of FP : FP / (FP + TN)***

***Error Rate of FN : FN / (FN + TP)***

M= 32

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | 10 | 20 | 30 | 40 | 50 |
| FP | 1549 | 7048 | 19352 | 35391 | 74764 |
| FP Error Rate | 1.3519174 | 1.5776161 | 1.9029845 | 1.9803394 | 2.696253 |
| FN | 1715 | 13503 | 21932 | 33266 | 56412 |
| FN Error Rate | 22.310394 | 32.291466 | 29.833773 | 23.07863 | 22.492374 |

M= 64

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | 10 | 20 | 30 | 40 | 50 |
| FP | 1215 | 4725 | 9329 | 30053 | 45637 |
| FP Error Rate | 1.070796 | 1.041033 | 0.92599857 | 1.7103361 | 1.6418189 |
| FN | 1896 | 7371 | 17184 | 19334 | 40952 |
| FN Error Rate | 24.27346 | 21.871105 | 22.208149 | 10.580059 | 16.648983 |

M= 128

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | 10 | 20 | 30 | 40 | 50 |
| FP | 1410 | 5885 | 9221 | 19930 | 31327 |
| FP Error Rate | 1.2315809 | 1.2930144 | 0.9104535 | 1.1211324 | 1.146076 |
| FN | 1410 | 7285 | 10873 | 26272 | 27577 |
| FN Error Rate | 17.045454 | 23.911118 | 13.5592165 | 18.57755 | 9.452627 |

M= 512

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | 10 | 20 | 30 | 40 | 50 |
| FP | 2004 | 6583 | 8172 | 10870 | 15396 |
| FP Error Rate | 1.7326796 | 1.4346018 | 0.8064437 | 0.6146417 | 0.555635 |
| FN | 2638 | 7312 | 8172 | 13477 | 20544 |
| FN Error Rate | 34.825085 | 24.625332 | 10.454807 | 8.119995 | 7.934528 |

M= 1024

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | 10 | 20 | 30 | 40 | 50 |
| FP | 1564 | 4475 | 7538 | 7416 | 6.911157 |
| FP Error Rate | 1.3505461 | 0.98548305 | 0.75096285 | 0.41448507 | 0.5380928 |
| FN | 1390 | 4868 | 10005 | 18007 | 21606 |
| FN Error Rate | 20.182953 | 13.726982 | 11.215991 | 11.893188 | 6.9111557 |

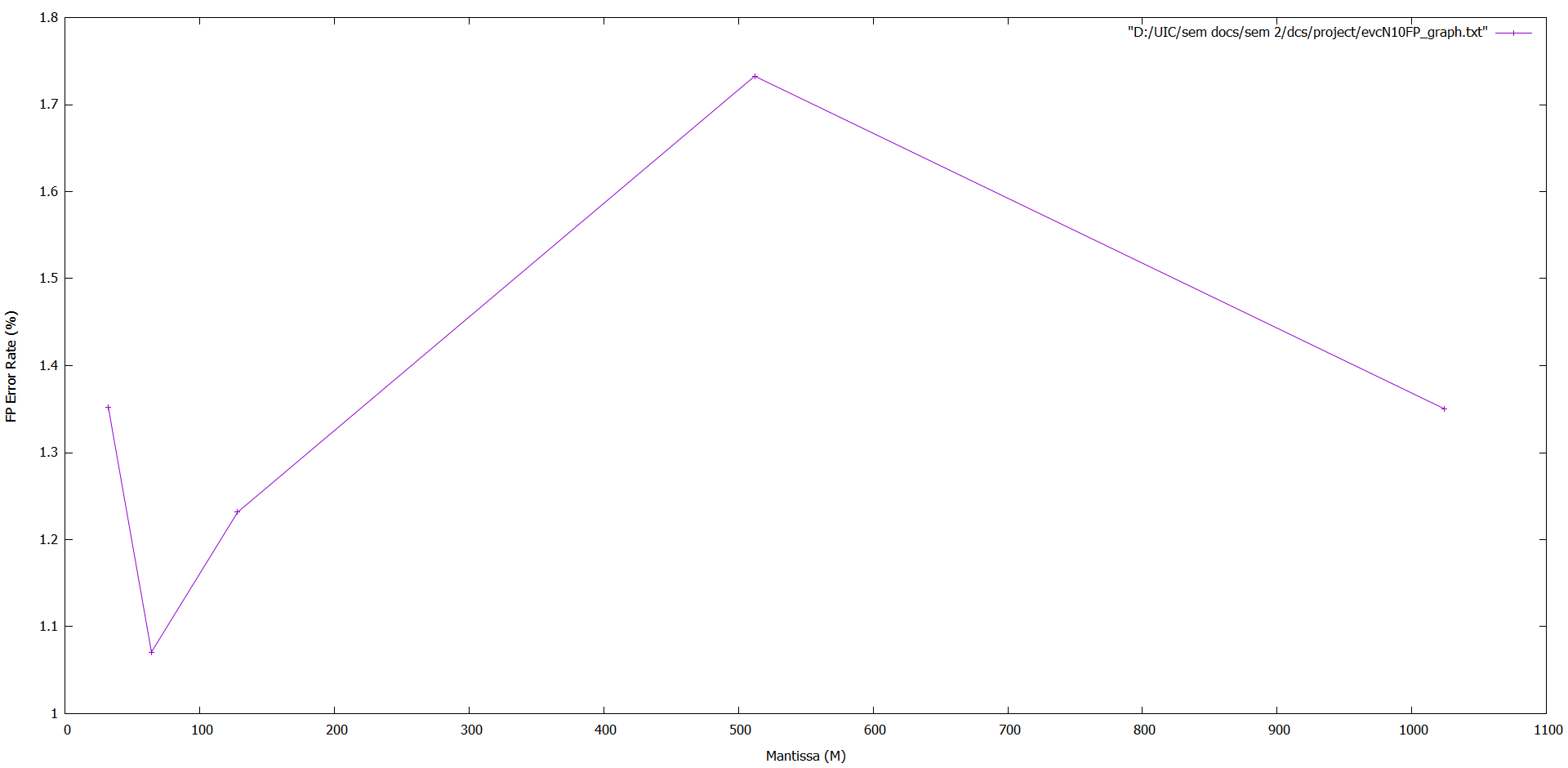
An important part of understanding the limitations is being able to quantify the level of accuracy (or its complement, the error rate introduced) in the use of logarithms and anti-logarithms.

Now we need to plot the percentage error rate (or degree of accuracy) of false negatives and false positives of the causality relation, for various n and m

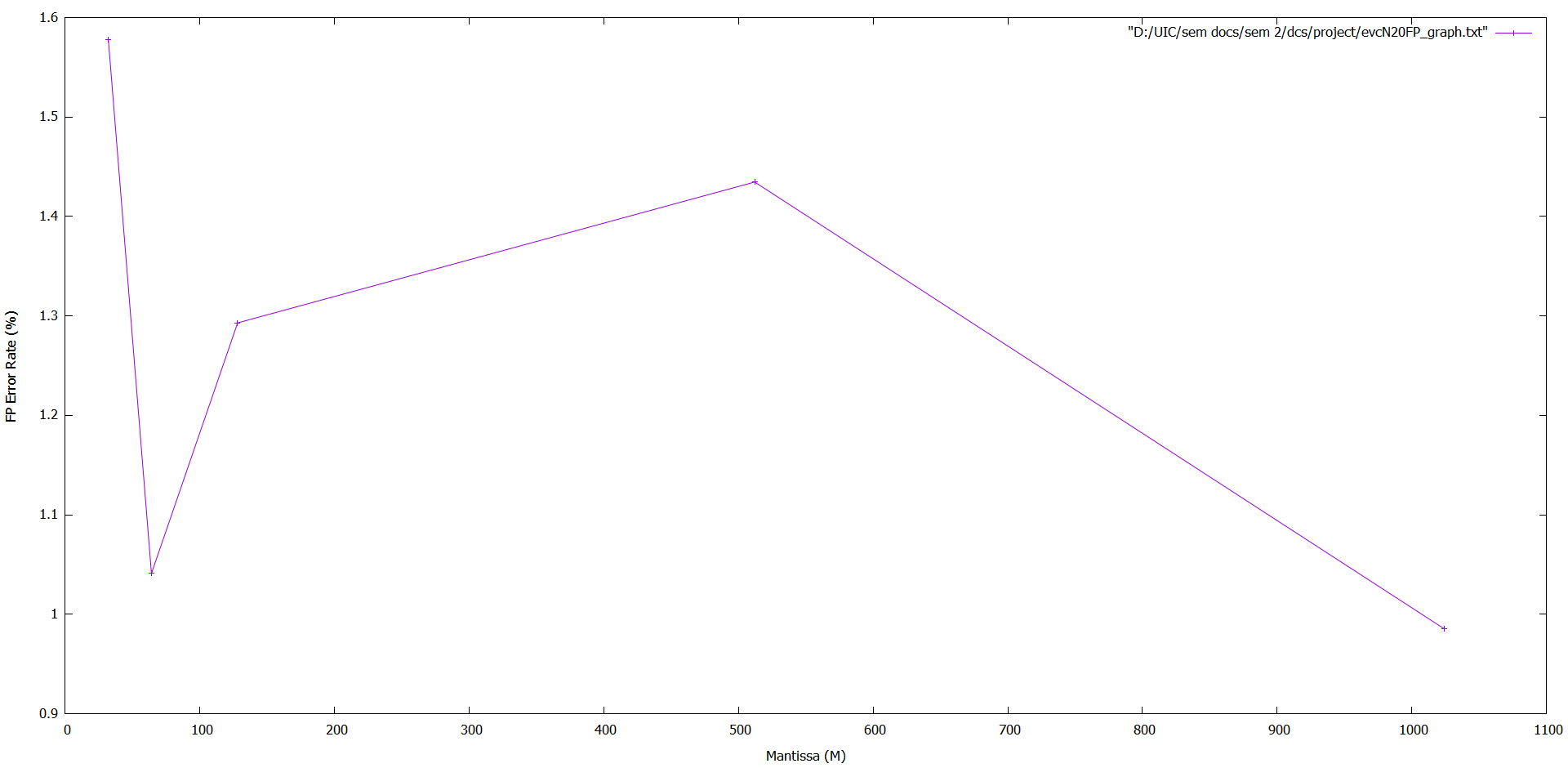
Using the above tables, we get generate the following table and plot its line graphs:

**Mantissa (M) v/s FP%**

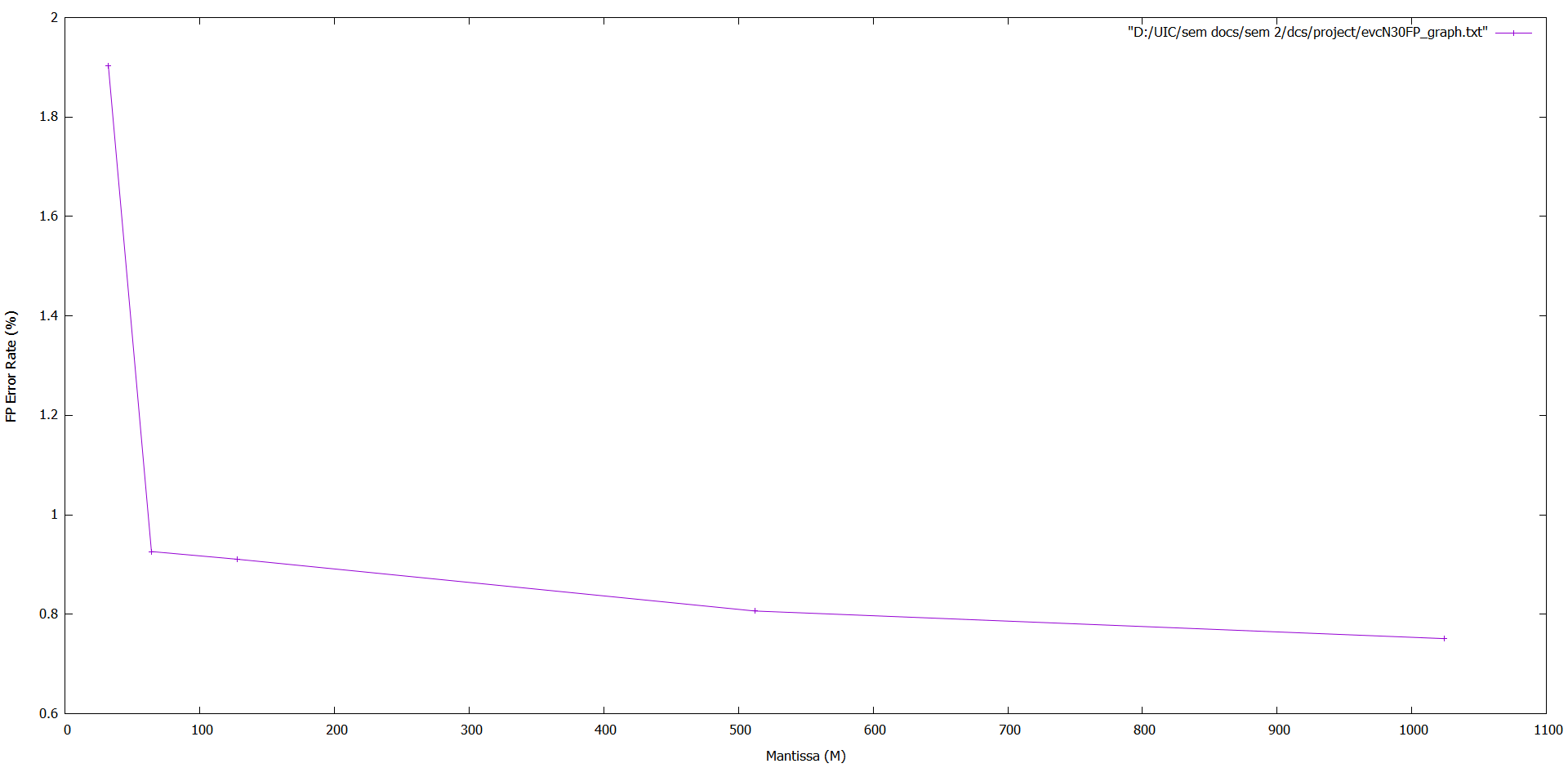
For N = 10



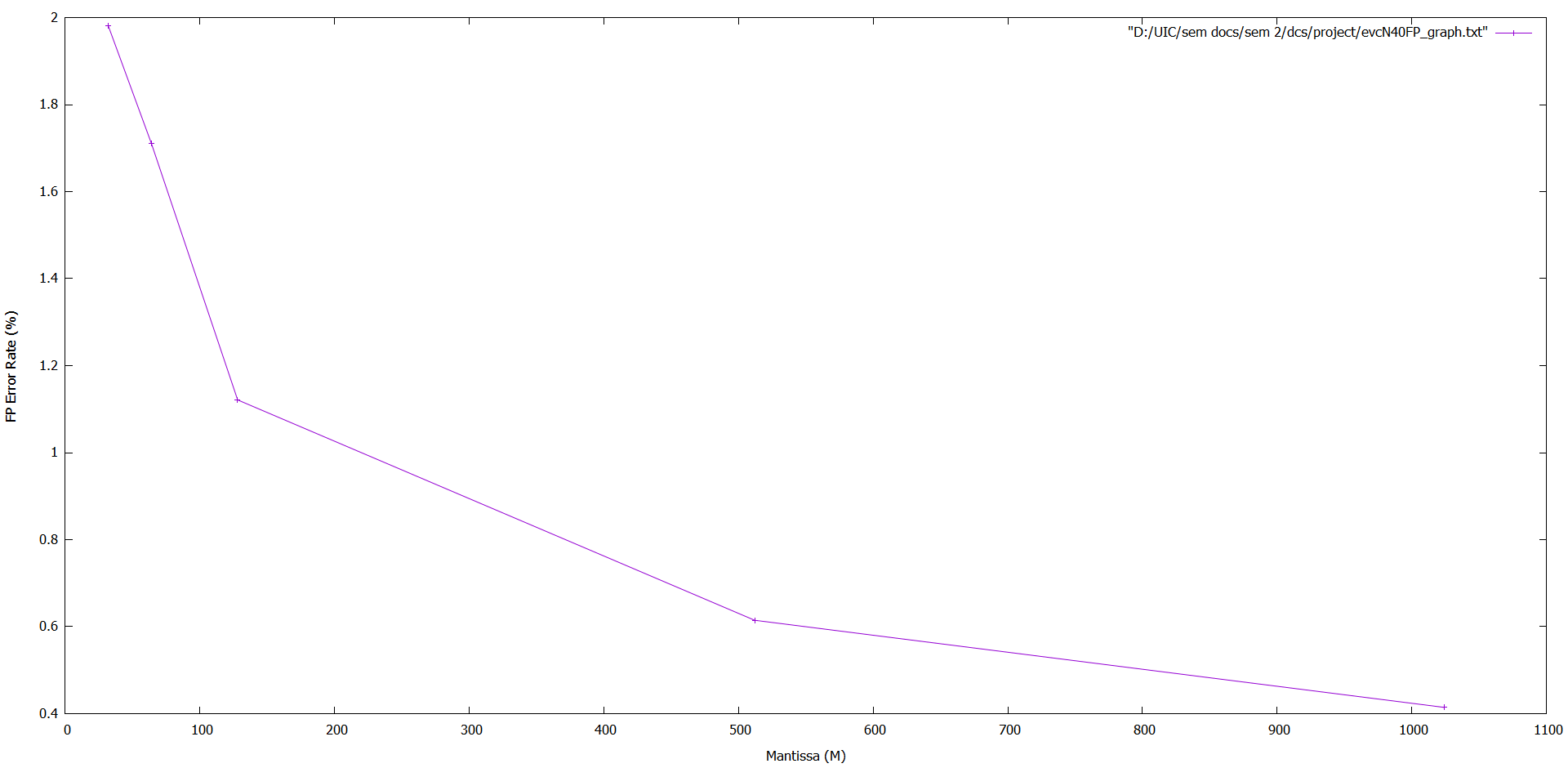
For N = 20



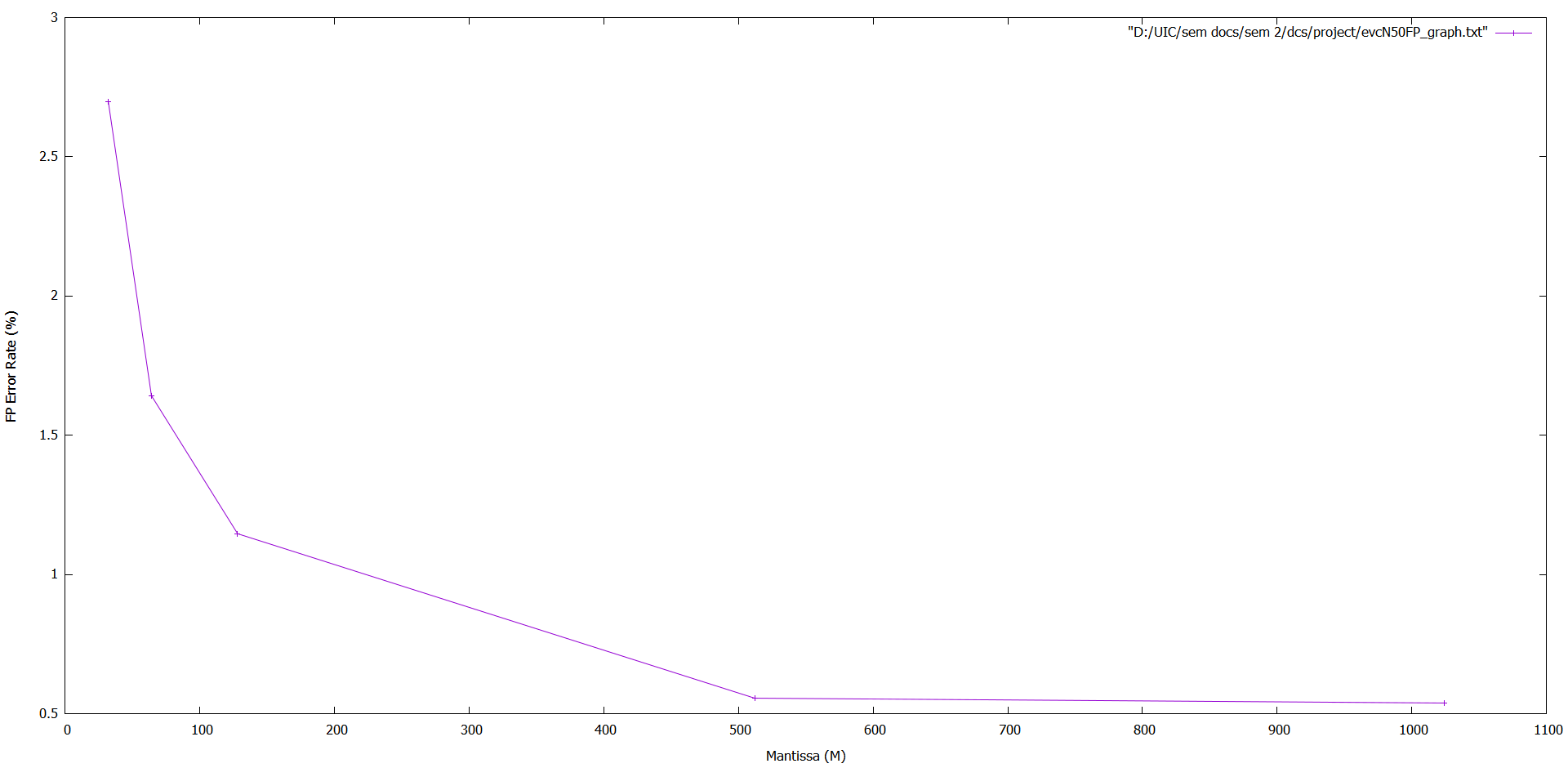
For N = 30



For N = 40



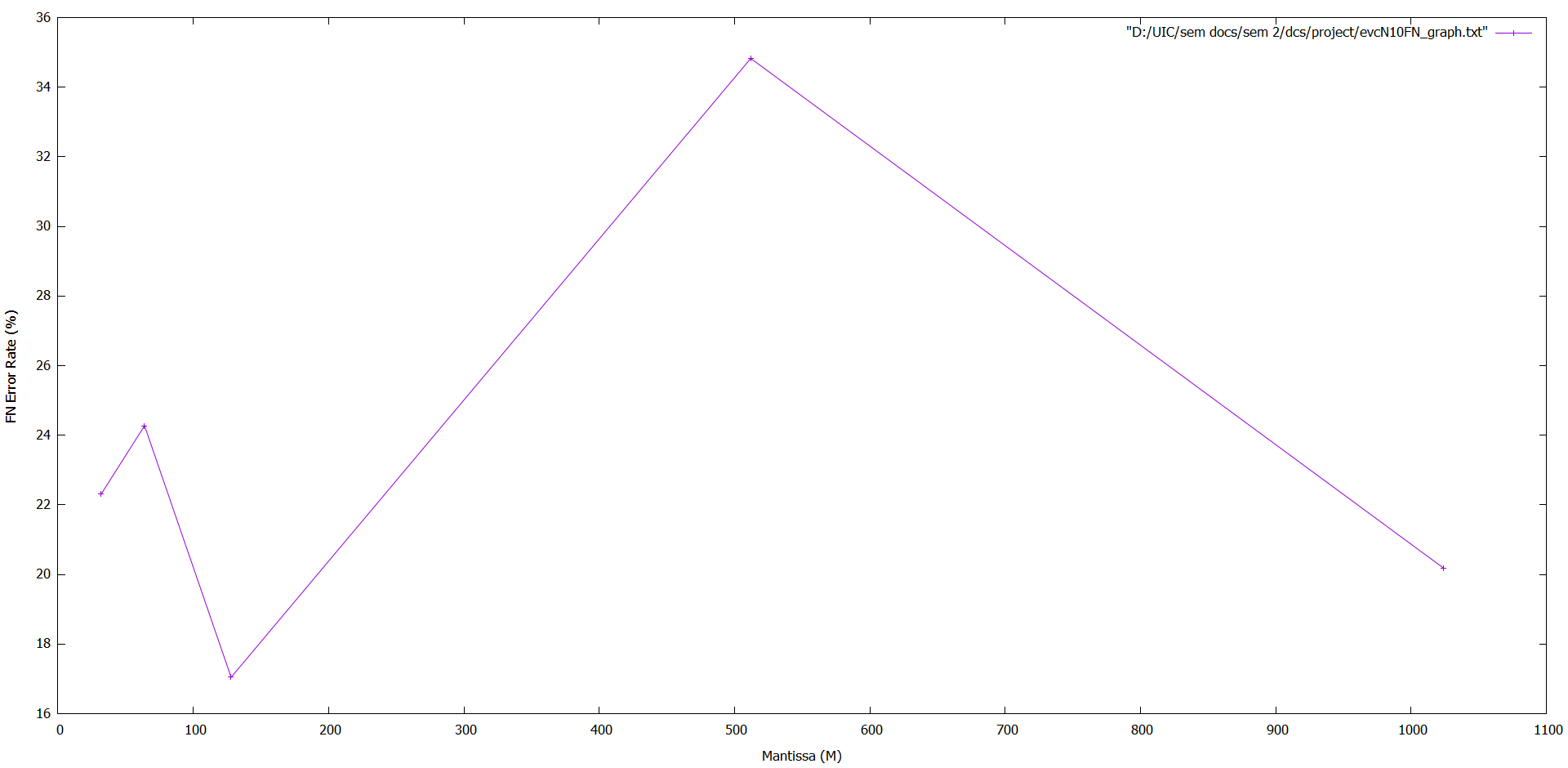
For N = 50



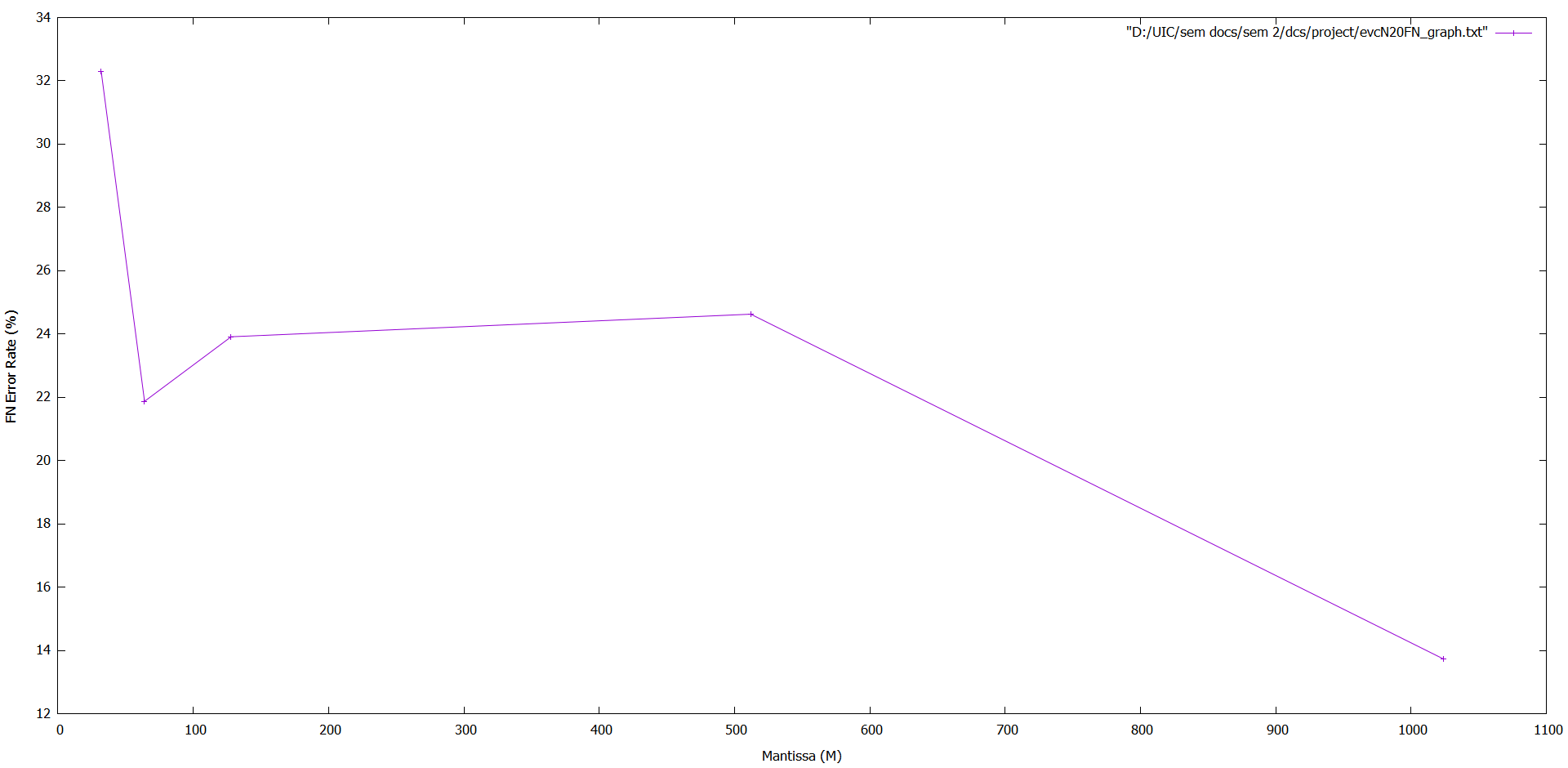
ANALYSIS: We observe the above 5 graphs and can conclude that as the mantissa (m) increases, the False Positive Error Rate (%) decreases. This is the case in all the graphs above for n = 10, 20, 30, 40, 50

**Mantissa (M) v/s FN%**

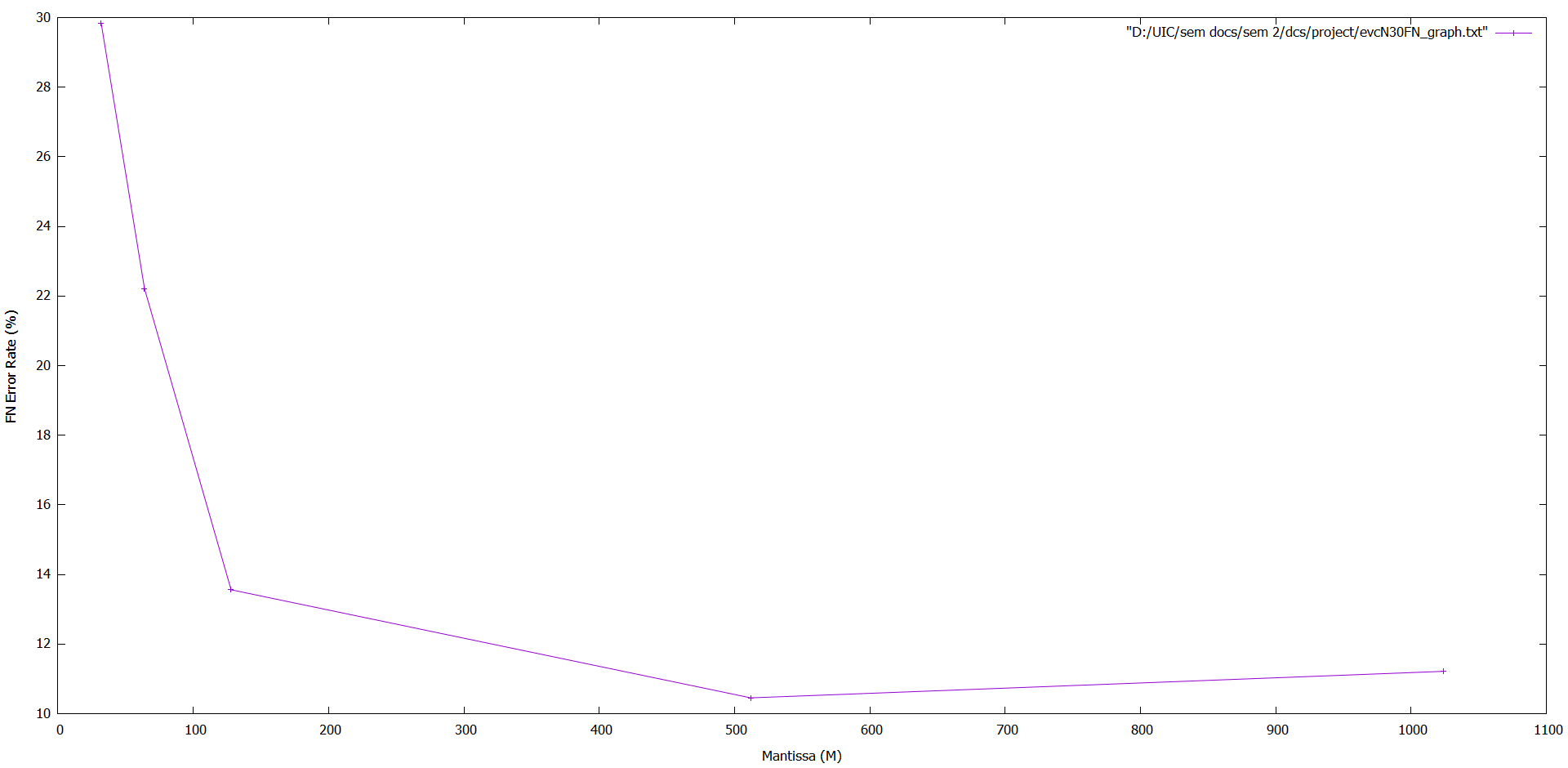
For N = 10



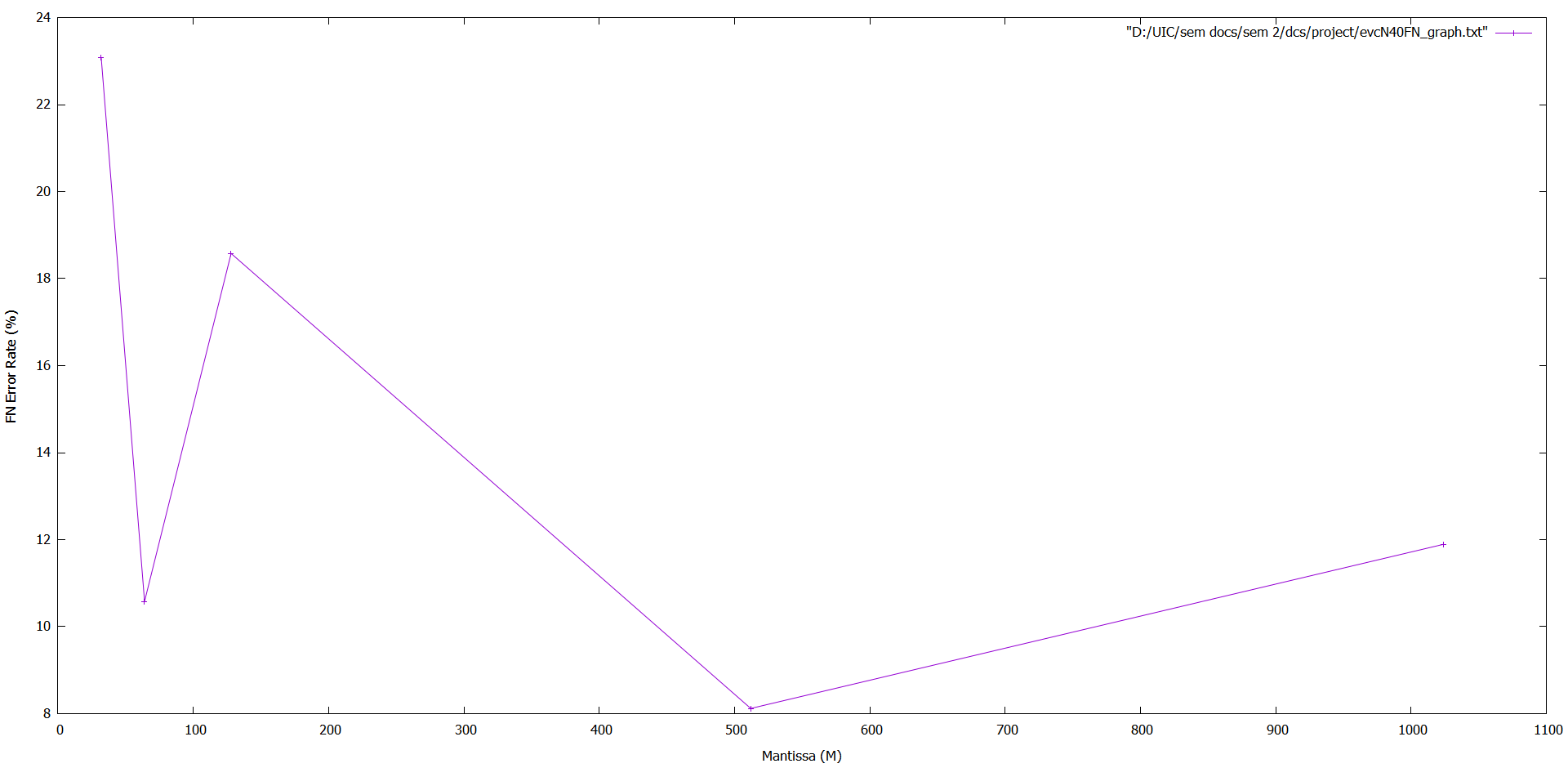
For N = 20



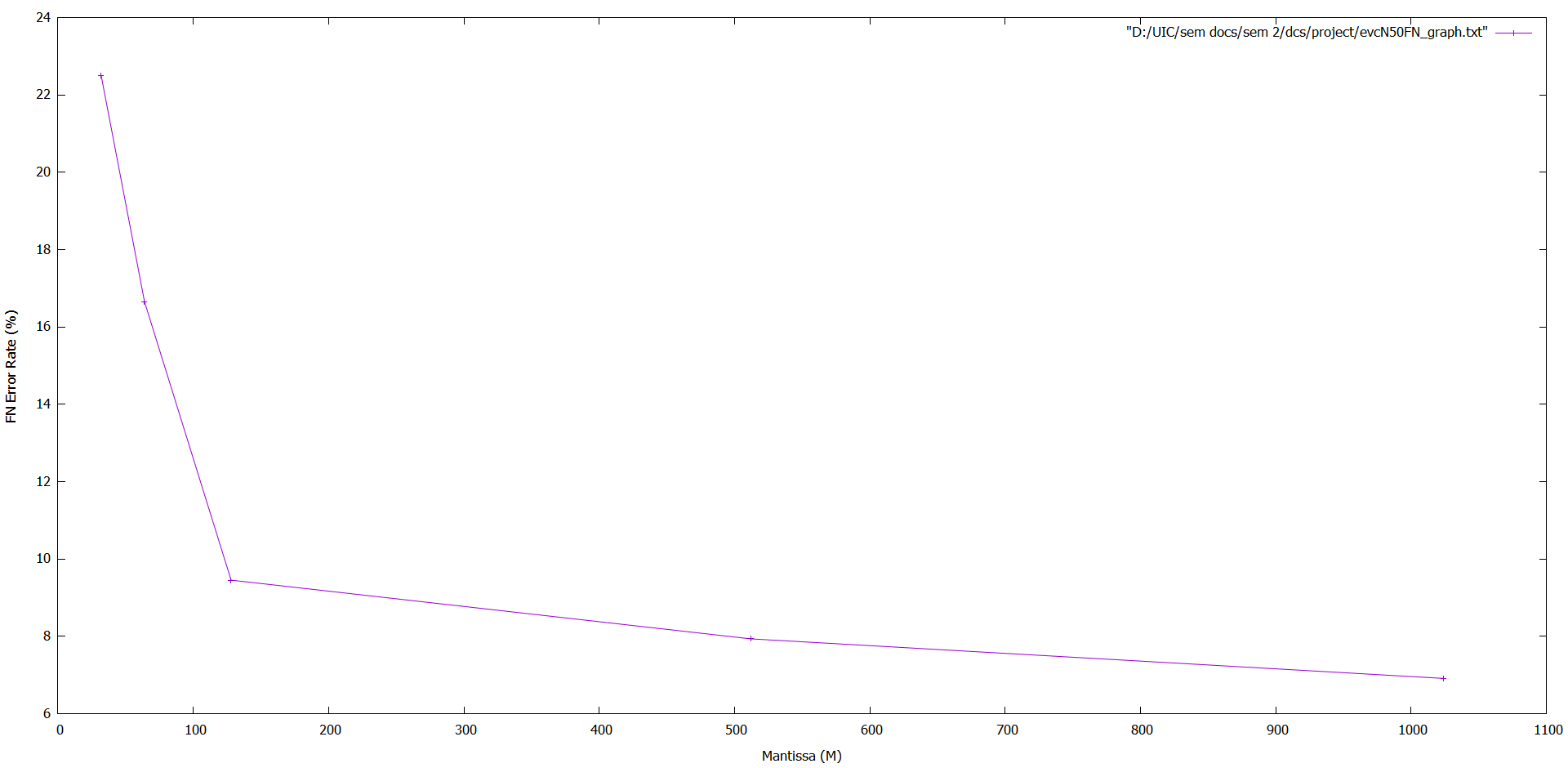
For N = 30



For N = 40



For N = 50



ANALYSIS: We observe the above 5 graphs and can conclude that as the mantissa (m) increases, the False Negative Error Rate (%) decreases. This is the case in all the graphs above for n = 10, 20, 30, 40, 50