a)

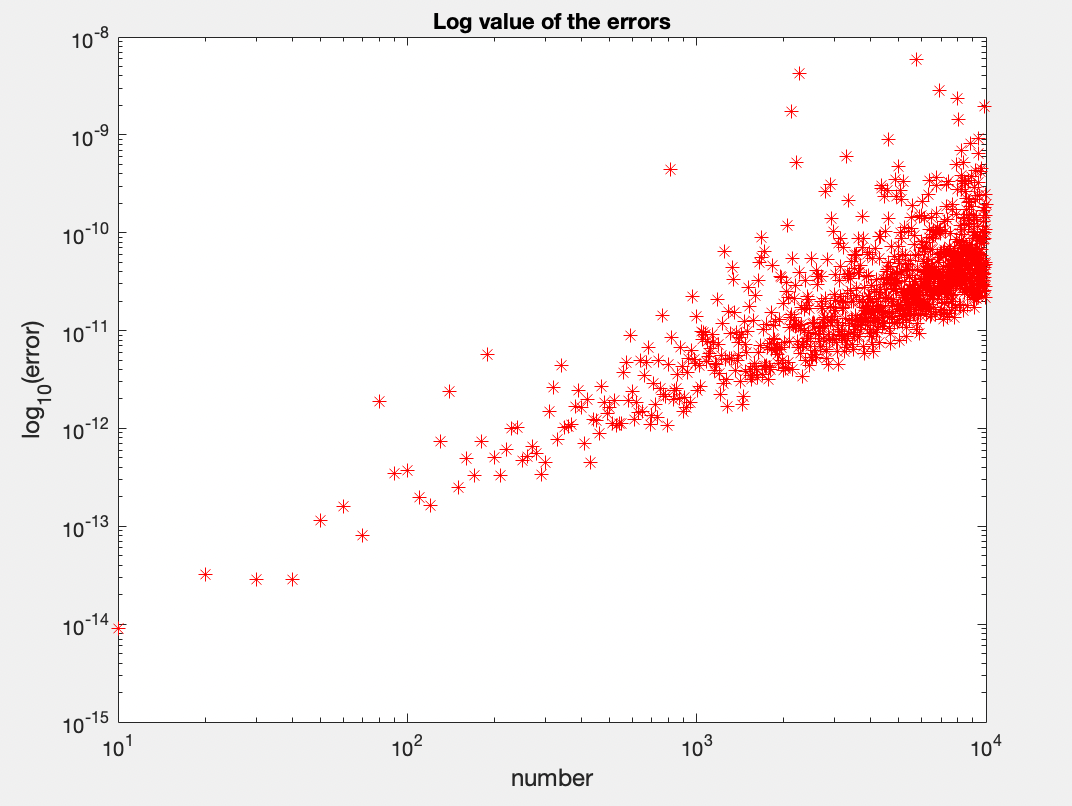
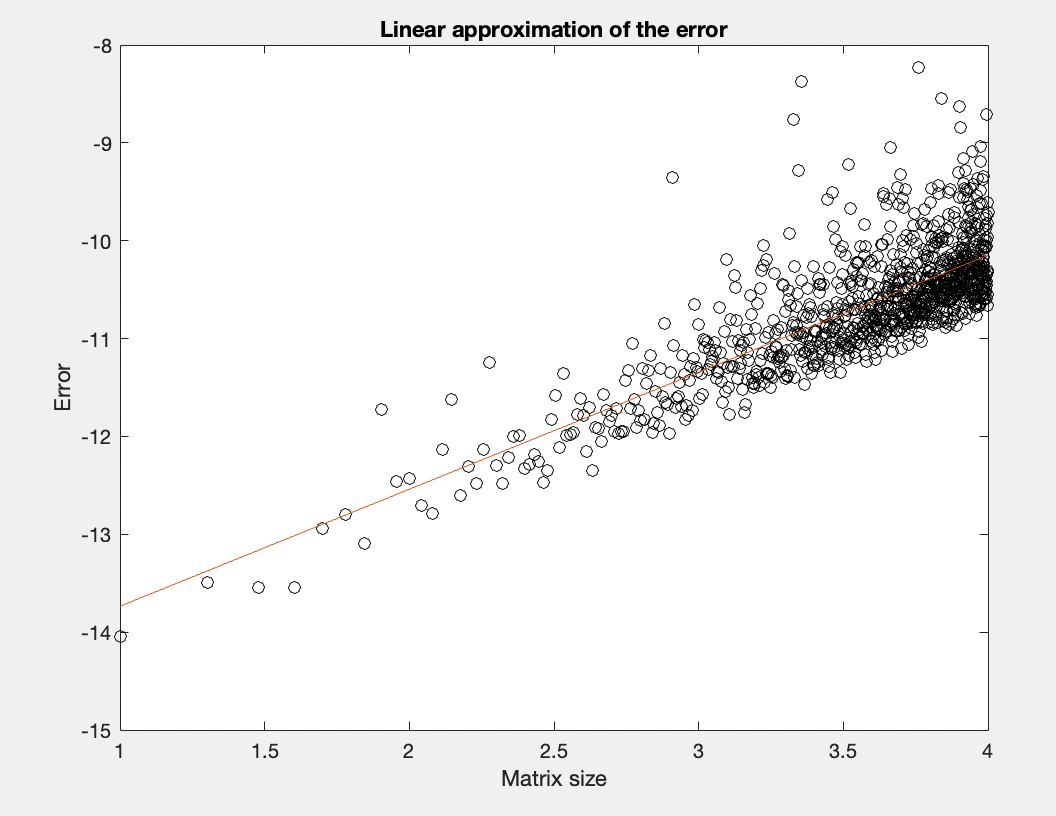


Figure 1 Figure 2

b) I chose N to be values from 10 to 10000 with an increment of 10 so that we have enough data points to have a more accurate linear approximation of the error. It also allowed me to have a big enough end value without making the computation extremely time consuming. The large number of points made it easy to observe the linear pattern in the error. I chose M to be 100 so that we have sufficient trials to improve the accuracy of the linear estimation. I also tried N values that went until 105 but the matrix size was too big for Matlab. A lower M (10) value gave me a weaker approximation of the slope and the y-intercept.

c) For different n values from 10 to 10000, random tridiagonal matrices are created and gaussian elimination is used by the backlash command to come up with the solution vector, M times for each n. The max error in each trial is stored and then later used to find the mean error for that n value. Log of the Mean errors of all the n values are then plotted against the respective n value. It can be observed from Figure 1 that the the log error follows a linear pattern. Then, polyfit is used to estimate the slope and the y-intercept of the line. The returned p1 (1.19563301207961)and p2(-14.9308145029054)are used in the equation: 1 = log N\* \* p1 + p2 ; to find the value of N\* where the error in the solution is 1.

D) Polyfit returns 1.19563301207961 and -14.9308145029054 as the slope and the y-intercept. Using,

y = m log x + c

1 = log N\* \* p1 + p2

1 = 19563301207961 log N\* - 14.9308145029054

1 = log N\* \* (1.19563301207961) -14.9308145029054

log N\* = 13.32419..

N\* = 1013.32419..

N\* = 2.1095..e13