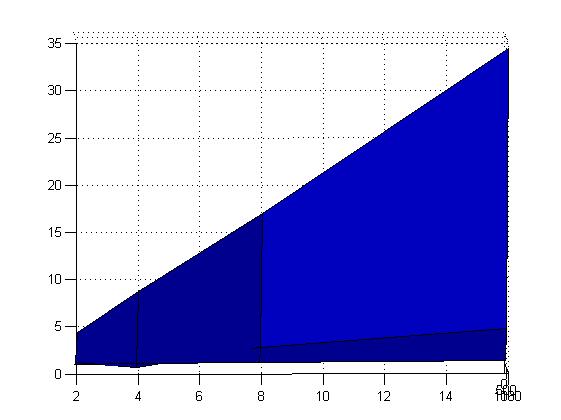
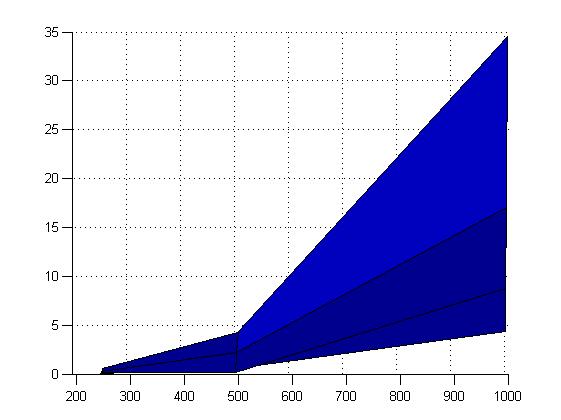
The graph is constructed by iterating through the desired number of nodes. The degree of the current node is found and connected to a random other node that isn’t K connected. If a node is already K connected then it is skipped over. The Constructor in the Random K regular graph and the getNotFullNodes method are involved in this. A very quick run time analysis showed that the time needed to generate the graph increases linearly with the K, the degree of each node in the graph and exponentially with N, the number of nodes in the graph. Both plots have time in seconds on the y axis.



Number of Edges per vertex



Number of Vertices

A breadth first search is implemented from node 0 to every other node in the graph to check whether it’s connected or not. If it is connected, an algorithm runs to check whether there is an Eulerian path which depended whether there was not exactly two nodes of odd degree and if every node was of even degree otherwise. If there is an Eulerian path, then every vertex must be of even degree so that every time a vertex was visited, there was a path available to leave it again. A stack was used to keep track of which edges had already been visited and an Eulerian path was calculated. This is essentially what my program implemented. Optimizations included using the triplet method of storing matrices which was modified to only store the upper diagonal as the matrix is symmetric. The BFS could have been written a lot more efficiently which means the program is slow to run on values greater than 1000 nodes. The constructor is also inefficient as it uses random\_shuffle to randomly shuffle an array of not yet K connected nodes for the current node to connect to rather than generating a random number. If I had more time this program could be made a lot more efficient, as the graph indicates, I would need to concentrate on optimising the program to deal with a graph with a large number of nodes. I could also replace the breadth first search to implement a more efficient algorithm to check if the graph is connected.