Observations and comparison of Algorithms:

To start with let's take a look at the iterations and time taken

	STEEPEST DESCENT	PRECONDITIONED SD	CG	PCG
ITERATIONS	18055	68	233	69
TIME TAKEN	22 sec	Less than 1 sec	Less than 1 sec	Less than 1 sec

Note: These numbers are approximations, we are using a random starting point so the numbers of iterations could vary, the same goes for time take, which could range from machine to machine.

The overall theme seems pretty clear from the above table, steepest descent is the least efficient, and preconditioned Steepest Descent and preconditioned Conjugate Descent are the most efficient.

The reason Steepest descent doesn't perform well here is because of the condition number of matrix A. Matlab's `cond(A)` method gives the conditioned number to be **1053.** This means the contours along the larger side are stretched 1053 times the smaller side. That would mean the direction of the Steepest descent would not always point toward the lowest point and we would zig-zak around for a while instead of straight descent.

For the same reason when we precondition the matrix, the same descent algorithm performs way better. We used the `ichol` method to precondition our matrix A, and the resultant matrix had a condition number of **351.** This is a huge improvement and the iteration results show that. However, we paid the cost of incomplete Coleskey and then we did an upper traiangle solve and a lower triangle solve each iteration.

The key idea behind CG is that each search direction is conjugate to all previous search directions with respect to matrix A. This means that the search directions form an orthogonal basis for the vector space in which the solution approximation lies, and each new search direction minimizes the residual over this space. As a result, CG can converge much faster than other iterative methods for solving linear systems, especially when matrix A is large and sparse. In our case, we had a 2500x2500 sparse matrix, so converging within 233 steps is a very good result. In terms of cost, we incurred 3 dot products and a couple of matrix-vector multiplications.

This conjugate descent gets better when we preconditioned the matrix, again with our matrix A, we were able to converge in 69 iterations.