Homework #4, Problem #1 of 2 Due 11/7/2019, 11:59pm, (upload your files to Blackboard)

Problem #1

Implement the standard CG algorithm, and use it to solve linear systems describing the "Helical Coordinate Preconditioner for the Laplacian" in 1D, 2D, and 3D:

Matlab-centric problem matrices; push n until you run out of memory (or patience!)

```
d = ones(n,1); A = spdiags([d -2*d d], [-1 0 1], n, n);
d = ones(n^2,1); A = spdiags([d d -4*d d d], [-n -1 0 1 n], n^2, n^2);
d = ones(n^3,1); A = spdiags([d d d -6*d d d d], [-n^2 -n -1 0 1 n n^2 ],
n^3, n^3);
```

Ponder matrix size; number of iterations to drive the initial residual (given the initial guess of all zeros $\vec{x}_0 =$ `zeros(size(d))`, with a right-hand-side of all ones $\vec{b} =$ `ones(size(d))`) to a residual of size *tol* $\times$ *initial residual*; execution time; condition numbers; non-zero matrix elements; total # of matrix elements, etc...

Homework #4, Problem #2 of 2

Problem #2

A modified version of NW^{1st}-5.1: Implement the standard CG algorithm, and use it to solve linear systems in which A is the Hilbert matrix, whose elements are $a_{ij} = 1/(i+j-1)$. Set the right-hand-side to be all ones $\vec{b} =$ `ones(n,1)`, and the initial point to be the origin $\vec{x}_0 =$ `zeros(n,1)`. For dimensions n = 5, 8, 12, 20, plot the norm of the residual as a function of the iteration; stop when the norm is less than 10^{-6}

Note: The Hilbert matrix shows up in the normal equations in least squares approximations, and is an example of a matrix with a nasty condition number.

Compute the condition number for your matrices, and plot the spread of the eigenvalues. From the formulas, estimate how many steepest descent iterations you would need to solve the problem to the same precision. (Can you get a meaningful estimate?)