1) There are many different iterative schemes that approximate the solution to the equation Ax=b. Among these techniques include the Conjugate Gradient (CG) and the Generalized Minimal Residual Method (GMRES). They are both Krylov subspace methods that are nonstationary iterative methods which do not have iterative matrices. They are instead aimed at minimizing the residual $r^{(k)} = b - Ax^{(k)}$ with respect to the vectors in the Krylov subspace. Although these two methods have some concepts in common, they also have many differences.

The CG method is designed to solve systems where the coefficient matrix A is symmetric positive definite. The minimization principle In this method is that x_k minimizes the A-norm of the error: $||x^* - x||_A = min_{x \in x0+K_k} ||x^* - x||_A$. CG is a very simple algorithm to implement and understand. In fact, it is simpler than Gaussian elimination with pivoting. It only deals with m vectors and not with individual entries of vectors or matrices. The only challenge is the choice of convergence criterion.

The GMRES method is designed to solve nonsymmetric systems. For this method, the kth iteration $x_k$minimizes the residual over $x_0 + K_k$: $\left||b - Ax_k|\right| = min_{x \in x0+K_k} ||b - Ax||$. This method does not require computation of a A^T on a vector, which is a very big advantage in a lot of cases. However, using residual polynomials is more complicated as decomposing matrix A is difficult without the spectral theorem. In addition, the method requires storing a basis for K_k and these storage requirements increase iteration progress.

Overall, CG and GMRES share similarities in that they have the same goal which is to approximate the closest value to the solution of Ax=b. They supersede the Jacobi, Gauss-Seidel, and SOR methods, but they also come with their challenges. They are efficient tools in numerical analysis.

[1] L. N. Trefethen and D. Bau (1997) Numerical Linear Algebra. SIAM: Philadelphia.
[2] C. T. Kelley (1995) Iterative Methods for Linear and Nonlinear Equations.

2a) For both CG and GMRES, the inputs are a nxn matrix A, nx1 column vector b. The output is the solution vector that we are trying to find and solve. The default tolerance level is 1e-6 and the default max iterations for CG is min(n,20) while for GMRES it is min(n,10). You can change these by entering in two more parameters in both of the functions. For example, you can enter A, b, tolerance, and max iterations into the function. For the GMRES function, the parameter 'restart' also needs to be entered. The norm that is used is (b-A*x)/norm(b). The p in pcg means preconditioned. A preconditioner can help make the system converge more quickly.

2b) BONUS. Please see my file name myCG.m to see the code for the conjugate gradient. Also, run the script called Bonus_script.m to see my solution.

I put in the following input:

```
A=[1 -0.5 0; -0.5 1 -0.5; 0 -0.5 1];
b=[0.5;0;0];
x0=[0;0;0];
tol=0.000001;
maxiter=25;

[x,iter]=myCG(A,b,x0,tol,maxiter)
```

And got the following output for solving the matrix when n=3 on the homework instructions:

```
>> Bonus_script

x =

    0.7500
    0.5000
    0.2500


iter =

     3
```

3a) In order to determine that the matrix when n=3 is symmetric positive definite, we must check if 1) $A^T = A$ and 2) $X^T AX > 0 \; \forall \; x \neq 0 \; \in \; R^n$ or in other words, all the eigenvalues are positive.

Condition 1:

$$\begin{bmatrix} 1 & -1/2 & 0 \\ -1/2 & 1 & -1/2 \\ 0 & -1/2 & 1 \end{bmatrix}^T = \begin{bmatrix} 1 & -1/2 & 0 \\ -1/2 & 1 & -1/2 \\ 0 & -1/2 & 1 \end{bmatrix}$$

=> Transpose of this matrix is equal to the original matrix so this condition is satisfied ✓

Condition 2:

Find eigenvalues where det(A-λI)=0

$$A-\lambda I = \begin{bmatrix} 1-\lambda & -1/2 & 0 \\ -1/2 & 1-\lambda & -1/2 \\ 0 & -1/2 & 1-\lambda \end{bmatrix}$$

$$\det = 0*\begin{vmatrix} -1/2 & 0 \\ 1-\lambda & -1/2 \end{vmatrix} - (-1/2)*\begin{vmatrix} 1-\lambda & 0 \\ -1/2 & -1/2 \end{vmatrix} + (1-\lambda)*\begin{vmatrix} 1-\lambda & -1/2 \\ -1/2 & 1-\lambda \end{vmatrix} = 0$$

½*((1-λ)(-1/2)-0)+(1-λ)((1-λ)(1-λ)-(1/4)) = 0

½*(-1/2+1/2λ)+(1-λ)(1-2λ+λ^2-1/4) = 0

-1/4 +1/4λ + (1-λ)(λ^2-2λ-3/4) = 0

-1/4 +1/4λ +λ^2-λ^3-2λ+2λ^2-3/4+3/4λ=0

$$\frac{-(\lambda-1)(2\lambda^2-4\lambda+1)}{2} = 0$$

λ = 0.292893, 1, 1.70711

=> All eigenvalues are positive so this condition is also satisfied ✓

So, matrix is symmetric positive definite because both conditions are true. This condition is true for all n because it will always be a square, tridiagonal matrix which means it is symmetric and is always positive definite. The theorem for positive definiteness says that a symmetric nxn real matrix M is positive definite if the scalar $z^T Mz$ is strictly positive for every non-zero column vector z of n real numbers.

$$M = \begin{bmatrix} a1 & b2 & 0 & \\ b2 & a2 & \ddots & 0 \\ 0 & \ddots & \ddots & bn \\ 0 & 0 & bn & an \end{bmatrix} \quad z = \begin{bmatrix} a \\ b \end{bmatrix}$$ so

$$[c_1 \quad c_2 \quad \dots \quad c_n] * \begin{bmatrix} a1 & b2 & 0 & \\ b2 & a2 & \ddots & 0 \\ 0 & \ddots & \ddots & bn \\ 0 & 0 & bn & an \end{bmatrix} * \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} = (a1*c1+b2)c1+(c1+a2*c2+c3)...$$ so this will be positive which means that for all n, the matrix will be symmetric positive definite.

3b) Please run the file named Project_script.m for the solution vectors since they are too large to paste on this document. In order to run Project_script, I wrote a function called Abmat.m to find the nxn matrices and also modified jacobi.m, gauss-seidel.m, and SOR.m. The following are tables for max iterations and p(T) for the iterative techniques for each n.

n=5

Methods	**Iterations**	**p(T)**
Jacobi	84	0.866025403784439
Gauss-Seidel	42	0.75

SOR	15	0.333333
PCG	5	N/A
GMRES	5	N/A

n=10

Methods	**Iterations**	**p(T)**
Jacobi	263	0.959492973614497
Gauss-Seidel	130	0.920626766415592
SOR	26	0.56039
PCG	10	N/A
GMRES	10	N/A

n=50

Methods	**Iterations**	**p(T)**
Jacobi	4104	0.998103328737045
Gauss-Seidel	2041	0.996210254835970
SOR	104	0.88402
PCG	50	N/A
GMRES	50	N/A

n=100

Methods	**Iterations**	**p(T)**
Jacobi	13276	0.999516282291987
Gauss-Seidel	6614	0.999032798566793
SOR	204	0.93968
PCG	100	N/A
GMRES	100	N/A

n=500

Methods	**Iterations**	**p(T)**
Jacobi	163789	0.999980449576215
Gauss-Seidel	81771	0.999960679538963
SOR	1004	0.98754
PCG	500	N/A
GMRES	500	N/A

Finding optimal w for SOR method:

$$w = \frac{2}{1 + \sqrt{1 - [p(T_J)]^2}}$$

For n=5

$$w = \frac{2}{1 + \sqrt{1 - [-0.866025403784439]^2}} = 1.333333$$

For n=10

$$w = \frac{2}{1 + \sqrt{1 - [\,-0.959492973614497]^2}} = 1.56039$$

For n=50

$$w = \frac{2}{1 + \sqrt{1 - [\,0.998103328737045]^2}} = 1.88402$$

For n=100

$$w = \frac{2}{1 + \sqrt{1 - [\,0.999516282291987]^2}} = 1.93968$$

For n=500

$$w = \frac{2}{1 + \sqrt{1 - [0.999960679538963]}} = 1.98754$$

The number of iterations and computational time gradually increase as n gets larger. In fact, there is a drastic difference in the number of iterations and time between n=5 and n=500. For SOR, the optimal choice of w does change as n changes because of the size of the matrix. The different sizes of the matrices gives different eigenvalues for Jacobi and Gauss-Seidel which in turn changes the calculation when computing w.