HW 4

Due Nov 5.

1: Gerschgorin and eigenvector localization

1. Suppose $w^* A = \lambda w^*$ for $A \in \mathbb{C}^{n \times n}$, and that

$$|a_{jj} - \lambda| > \sum_{i \neq j} |a_{ij}|.$$

   Argue that w_j cannot be the largest magnitude entry in w.

2. Let $G_j = \{z \in \mathbb{C} : |a_{jj} - z| \leq \sum_{i \neq j} |a_{ij}|\}$ be the jth Gerschgorin disk for a matrix A. Suppose $\cup_{j \in J} G_j$ is a connected component of the Gerschgorin set, i.e. $G_j \cap G_k = \emptyset$ for any $j \in J$ and $k \notin J$. Show that for any eigenvalue $\lambda \in \cup_{j \in J} G_j$, the corresponding row eigenvector must have its largest magnitude entry (or entries) at indices in J.

2: Reordering Schur Suppose that $T \in \mathbb{R}^{2 \times 2}$ is upper triangular, i.e.

$$T = \begin{bmatrix} \lambda_1 & \gamma \\ 0 & \lambda_2 \end{bmatrix}.$$

Write a routine to compute a Givens rotation

$$G = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}, \quad c^2 + s^2 = 1$$

such that

$$G^T T G = \begin{bmatrix} \lambda_2 & \eta \\ 0 & \lambda_1 \end{bmatrix}.$$

Your function should have the form

```
function [c,s] = p2reorder(T)
```

3: Solving Sylvester Consider the *Sylvester equation*

$$AX - XB = C$$

where X and C are m-by-n, A is m-by-m, and B is n-by-n. We are going to derive an efficient method for solving Sylvester equations (the *Bartels-Stewart* method).

1. Given the Schur decompositions of A and B, show how $AX - XB = C$ can be transformed into $A'Y - YB' = C'$ where A' and B' are triangular.

2. Describe how to solve the entries of Y one at a time by a process like back substitution. What condition on the eigenvalues of A and B guarantee the system is nonsingular?

3. Use the idea above to implement a Sylvester solver. Your function should have the form

```
function [X] = p3sylvester(A,B,C)
```

4: Converging QR Plot the diagonal of the iterates in the unshifted QR iteration for the following matrices

1. A 6-by-6 matrix of standard normals (`A = randn(6);`)

2. `B = randn(6); A = B*diag(1:6)/B;`

3. `A = [1, 10; -1, 1];`

In each case, take enough iterations that the curves converge or cycle. What do you observe?