Homework 7

Elizabeth Hunt

November 27, 2023

1 Question One

See `UTEST(eigen, dominant_eigenvalue)` in `test/eigen.t.c` and the entry `Eigen-Adjacent -> dominant_eigenvalue` in the LIZFCM API documentation.

2 Question Two

See `UTEST(eigen, leslie_matrix_dominant_eigenvalue)` in `test/eigen.t.c` and the entry `Eigen-Adjacent -> leslie_matrix` in the LIZFCM API documentation.

3 Question Three

See `UTEST(eigen, least_dominant_eigenvalue)` in `test/eigen.t.c` which finds the least dominant eigenvalue on the matrix:

$$\begin{bmatrix} 2 & 2 & 4 \\ 1 & 4 & 7 \\ 0 & 2 & 6 \end{bmatrix}$$

which has eigenvalues: $5 + \sqrt{17}, 2, 5 - \sqrt{17}$ and should thus produce $5 - \sqrt{17}$.
See also the entry `Eigen-Adjacent -> least_dominant_eigenvalue` in the LIZFCM API documentation.

4 Question Four

See `UTEST(eigen, shifted_eigenvalue)` in `test/eigen.t.c` which finds the least dominant eigenvalue on the matrix:

$$\begin{bmatrix} 2 & 2 & 4 \\ 1 & 4 & 7 \\ 0 & 2 & 6 \end{bmatrix}$$

which has eigenvalues: $5 + \sqrt{17}, 2, 5 - \sqrt{17}$ and should thus produce 2.0.
With the initial guess: $[0.5, 1.0, 0.75]$.
See also the entry `Eigen-Adjacent -> shift_inverse_power_eigenvalue` in the LIZFCM API documentation.

5 Question Five

See `UTEST(eigen, partition_find_eigenvalues)` in `test/eigen.t.c` which finds the eigenvalues in a partition of 10 on the matrix:

$$\begin{bmatrix} 2 & 2 & 4 \\ 1 & 4 & 7 \\ 0 & 2 & 6 \end{bmatrix}$$

which has eigenvalues: $5 + \sqrt{17}, 2, 5 - \sqrt{17}$, and should produce all three from the partitions when given the guesses $[0.5, 1.0, 0.75]$ from the questions above.
See also the entry `Eigen-Adjacent -> partition_find_eigenvalues` in the LIZFCM API documentation.

6 Question Six

Consider we have the results of two methods developed in this homework: `least_dominant_eigenvalue`, and `dominant_eigenvalue` into `lambda_0`, `lambda_n`, respectively. Also assume that we have the method implemented as we've introduced, `shift_inverse_power_eigenvalue`.
Then, we begin at the midpoint of `lambda_0` and `lambda_n`, and compute the `new_lambda = shift_inverse_power_eigenvalue` with a shift at the midpoint, and some given initial guess.

1. If the result is equal (or within some tolerance) to `lambda_n` then the closest eigenvalue to the midpoint is still the dominant eigenvalue, and thus the next most dominant will be on the left. Set `lambda_n` to the midpoint and reiterate.
2. If the result is greater or equal to `lambda_0` we know an eigenvalue of greater or equal magnitude exists on the right. So, we set `lambda_0` to this eigenvalue associated with the midpoint, and re-iterate.
3. Continue re-iterating until we hit some given maximum number of iterations. Finally we will return `new_lambda`.