ENGN 2020:

Homework #3

Brown University School of Engineering

Assigned: February 11, 2019, Due: February 23, 2019

Readings: Chapter 8, 20.

Problem 1

Part a. A point (x_1, x_2) on the Cartesian plane can be interpreted as coefficients that multiply the standard (*i.e.*, Cartesian) basis vectors; e.g., the vector (1.5, 2.2) would be expressed as

$$1.5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2.2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

More generally, a vector $\underline{\mathbf{v}}$ is expressed in a basis B as $\sum_i c_i \underline{\mathbf{u}}_i$, where $\underline{\mathbf{u}}_i$ are the basis vectors and c_i are the coefficients that describe this point in space. In eigenvalue problems, it is often desirable to use the eigenvectors as a basis set.

Write a script that converts an n-dimensional vector `vC` from standard (Cartesian) coordinates into a new coordinate system specified by its basis vectors. As an example, if you are given the Cartesian point (1.5, 2.2) with basis vectors $\begin{bmatrix} 1 & 2 \end{bmatrix}^{\mathrm{T}}$ and $\begin{bmatrix} 2 & -1 \end{bmatrix}^{\mathrm{T}}$, your vector expressed in the new basis should be $\begin{bmatrix} 1.18 & 0.16 \end{bmatrix}^{\mathrm{T}}$.

Submission

Label: hw3_1a
Points: 1
Input variables: `vC` (numpy array of shape $(n, 1)$), `B` (numpy array of shape (n, n) containing the basis vectors of the new coordinate system on the columns)
Output: numpy array of shape $(n, 1)$ containing the transformed coordinates

Part b. Now generalize the above approach. You now have two complete basis sets, $B = \{\underline{\mathbf{u}}_1, \underline{\mathbf{u}}_2, \cdots, \underline{\mathbf{u}}_n\}$ and $B' = \{\underline{\mathbf{u}}'_1, \underline{\mathbf{u}}'_2, \cdots, \underline{\mathbf{u}}'_n\}$ where $\underline{\mathbf{u}}_i$ and $\underline{\mathbf{u}}'_i$ are the basis vectors of each set, for n-dimensional vectors. Write a function that takes in the vector `vB` expressed in base B:

$$\underline{\mathbf{v}}_B = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix}^{\mathrm{T}}$$

and converts it to base B'.

Submission

Label: hw3_1b
Points: 1
Input variables: `vB` (numpy array of shape $(n, 1)$), `B` (numpy array of shape (n, n) containing the basis vectors of set B on the columns), `Bprime` (same as `B`, but for B')
Output: numpy array of shape $(n, 1)$ containing the transformed coordinates

Problem 2

K8-4-5. You may work this either by hand or on a computer, but in either case show your intermediate steps. Note that there are two typos in the book. The problem instructions should read:

> Similar matrices have equal eigenvalues. Verify this for $\underline{\underline{\mathbf{A}}}$ and $\underline{\underline{\hat{\mathbf{A}}}} = \underline{\underline{\mathbf{P}}}^{-1}\underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{P}}}$. If $\underline{\mathbf{y}}$ is an eigenvector of $\underline{\underline{\hat{\mathbf{A}}}}$, show that $\underline{\mathbf{x}} = \underline{\underline{\mathbf{P}}}\,\underline{\mathbf{y}}$ are eigenvectors of $\underline{\underline{\mathbf{A}}}$. Show the details of your work.

Paper Submission

(Turn in a PDF to Canvas.)
Points: 1

Problem 3

Write a code that uses the power method to estimate the dominant eigenvector of a matrix $\underline{\underline{\mathbf{A}}}$. Specifically, it should scale each eigenvector guess by the l_∞ norm, as in K-20.8 Example 1, and return the estimate of the eigenvector after n iterations. You can verify your code works on the textbook example.

Submission

Label: hw3_3
Points: 1
Input variables: `A` (numpy array of shape (m, m)), `x0` (initial eigenvector guess; numpy array of shape $(m, 1)$, `n` (integer)
Output: numpy array of shape $(n, 1)$ containing the eigenvalue estimate after n iterations

Problem 4

For this problem, you will plot the Gerschgorin-disc estimates of the eigenvalues of an arbitrary matrix $\underline{\underline{\mathbf{A}}}$, along with the actual eigenvalues as solved for with numpy.

Part a. First, create a function that gives the center and radius of the Gerschborin disks for a given matrix $\underline{\underline{\mathbf{A}}}$; you should return them in the order corresponding to the rows of the matrix. Your function should also return the true eigenvalues of $\underline{\underline{\mathbf{A}}}$ as calculated with `np.linalg.eig`. You can check your code against Example 1 of K20.7.

Submission

Label: hw3_4
Points: 1
Input variables: `A` (numpy array of shape (n, n))
Output: a dictionary containing three items: `'centers'` (a shape-$(n, 1)$ numpy arry of the disk centers), `'radii'` (a shape-$(n, 1)$ numpy array of the disk radii), `'eigvalues'` (a shape-$(n, 1)$ numpy array of the eigenvalues)

Part b. Use the function you just created along with matplotlib.pyplot in order to create a figure that shows the Gerschgorin disks on the complex plane along with the true eigenvalues as points on the same plot. Your figure should look roughly like that of Fig 449 in the text, but with the true eigenvalues added. Use your script to produce plots for Problems 1–6 of K20.7.

Paper Submission

(Turn in a PDF to Canvas.)
Points: 1.5

Problem 5

K8-4-6.

Paper Submission

(Turn in a PDF to Canvas.)
Points: 2

Problem 6

K8-5 Problems 1–6. Ignore the textbook instructions; instead for each matrix:

1. Identify the matrix as Hermitian, skew-Hermitian or unitary, and use Theorem 1 to predict the characteristics of the eigenvalues.
2. Calculate and report the actual eigenvalues. You do not need to report eigenvectors.

Paper Submission

(Turn in a PDF to Canvas.)
Points: 1.5

Problem 7

Consider the function with unknown root(s) x:

$$e^x - 3x - 5 = 0$$

Using fixed-point iteration, plot solution trajectories for various initial guesses $x^{(0)}$. That is, plot the number of iterations on the abscissa of your figure and the value $x^{(k)}$ on the ordinate. Your plot should have a suitable number of trajectories to show the convergence behavior over reasonable initial guess of $x^{(0)}$ (and the ordinate should be zoomed appropriately to see the real roots, not diverging solutions).

Repeat with Newton–Raphson iteration.

Paper Submission

Turn in two figures. Both should be clearly labeled and professionally presented.
Points: 2

Problem 8

Consider the function

$$f(x) = \sin x - \frac{x}{10} = 0$$

Part a. Make a plot of this function; be sure to include an $f = 0$ line on your plot such that roots can be clearly identified. Write a `for` loop that finds all the roots of this equation (with `np.optimize.fsolve`), using suitable initial guesses that you create by looking at the plot. Turn in your plot and the list of roots.

Part b. Next we'll examine factoring out roots. Choose any root $x = r_1$ of this equation, and create a new function $g(x)$ that has this root factored out; that is, $g \equiv f/(x - r_1)$. Plot f and g on the same plot, such that it can be seen if f and g still have common roots. (You may want to multiply g by a constant before plotting so that the x intercepts are more visible.) Use your `for` loop to find all the roots of g and verify they are the same as f, except for r_1.

Part c. Repeat part b, but factoring out any two more roots (that is, three total: r_1, r_2, r_3).

Part d. Can you see any numerical problems that might be encountered due to factoring out roots?

Paper Submission

(Turn in a complete solution to all parts.)
Points: 3