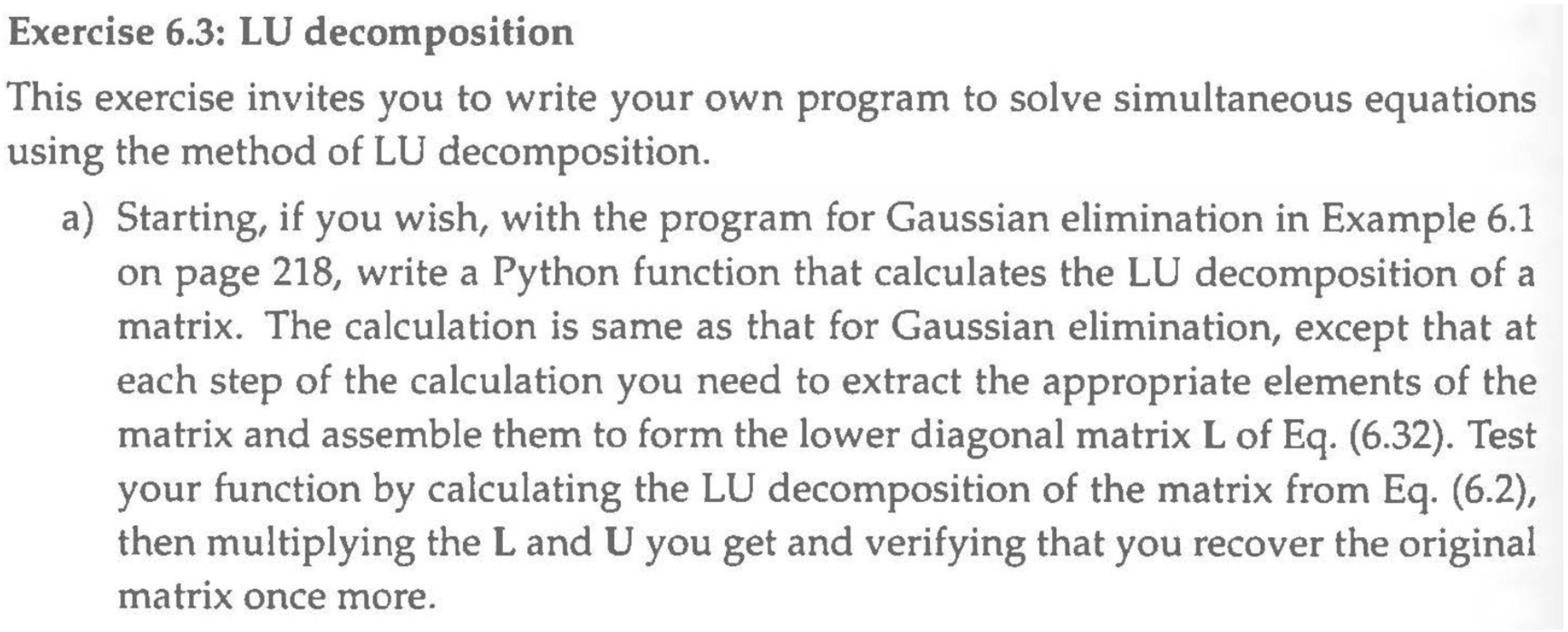
Homework 3

Due Apr 7, 2024

Mar. 25, 2024

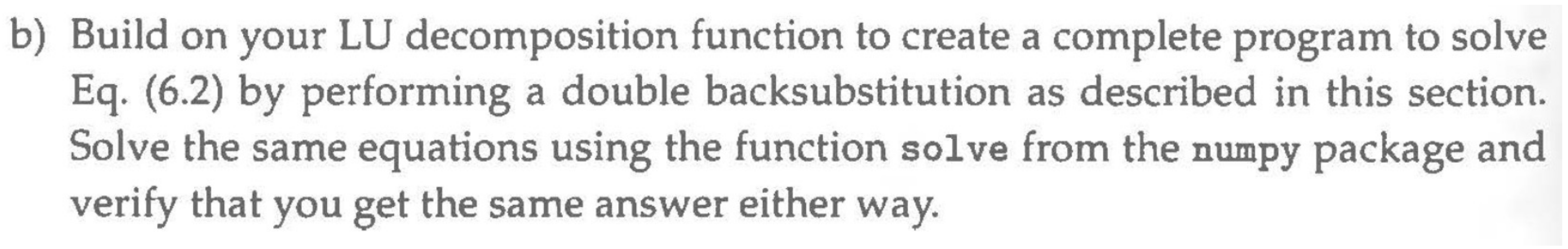
Problem 3.1



Exercise 6.3: LU decomposition

This exercise invites you to write your own program to solve simultaneous equations using the method of LU decomposition.

a) Starting, if you wish, with the program for Gaussian elimination in Example 6.1 on page 218, write a Python function that calculates the LU decomposition of a matrix. The calculation is same as that for Gaussian elimination, except that at each step of the calculation you need to extract the appropriate elements of the matrix and assemble them to form the lower diagonal matrix **L** of Eq. (6.32). Test your function by calculating the LU decomposition of the matrix from Eq. (6.2), then multiplying the **L** and **U** you get and verifying that you recover the original matrix once more.



b) Build on your LU decomposition function to create a complete program to solve Eq. (6.2) by performing a double backsubstitution as described in this section. Solve the same equations using the function `solve` from the `numpy` package and verify that you get the same answer either way.

Problem 3.1 continued

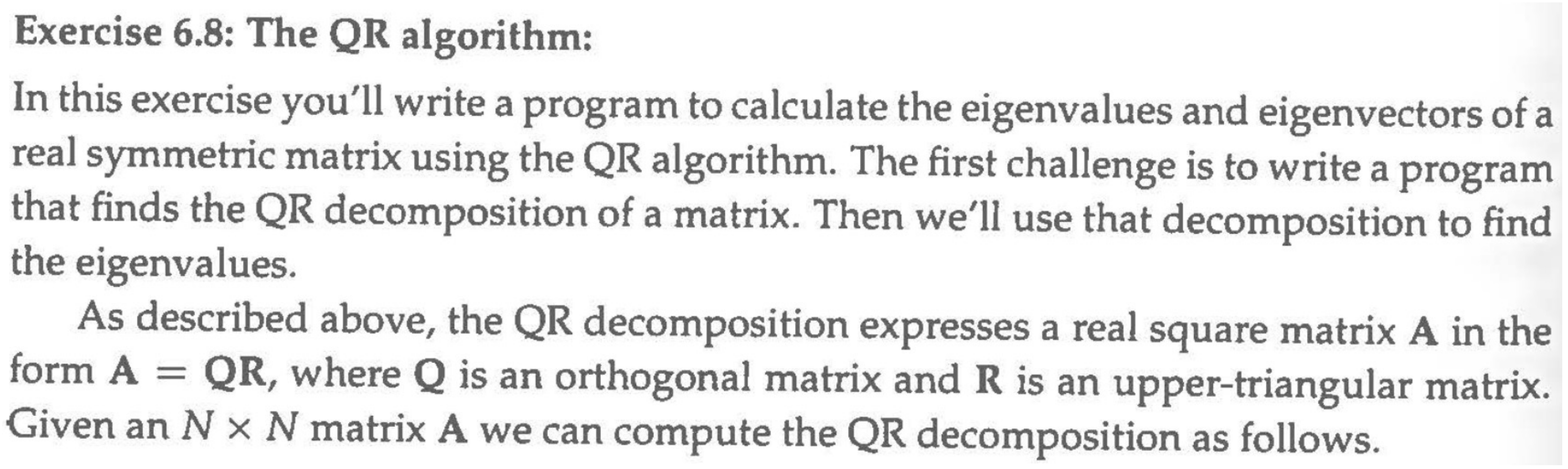
c) If you're feeling ambitious, try your hand at LU decomposition with partial pivoting. Partial pivoting works in the same way for LU decomposition as it does for Gaussian elimination, swapping rows to get the largest diagonal element as explained in Section 6.1.3, but the extension to LU decomposition requires two additional steps. First, every time you swap two rows you also have to swap the same rows in the matrix **L**. Second, when you use your LU decomposition to solve a set of equations $\mathbf{Ax} = \mathbf{v}$ you will also need to perform the same sequence of swaps on the vector **v** on the right-hand side. This means you need to record the swaps as you are doing the decomposition so that you can recreate them later. The simplest way to do this is to set up a list or array in which the value of the ith element records the row you swapped with on the ith step of the process. For instance, if you swapped the first row with the second then the second with the fourth, the first two elements of the list would be 2 and 4. Solving a set of equations for given **v** involves first performing the required sequence of swaps on the elements of **v** then performing a double backsubstitution as usual. (In ordinary Gaussian elimination with pivoting, one swaps the elements of **v** as the algorithm proceeds, rather than all at once, but the difference has no effect on the results, so it's fine to perform all the swaps at once if we wish.)

Problem 3.1 continued

Modify the function you wrote for part (a) to perform LU decomposition with partial pivoting. The function should return the matrices **L** and **U** for the LU decomposition of the swapped matrix, plus a list of the swaps made. Then modify the rest of your program to solve equations of the form $\mathbf{Ax} = \mathbf{v}$ using LU decomposition with pivoting. Test your program on the example from Eq. (6.17), which cannot be solved without pivoting because of the zero in the first element of the matrix. Check your results against a solution of the same equations using the `solve` function from `numpy`.

LU decomposition with partial pivoting is the most widely used method for the solution of simultaneous equations in practice. Precisely this method is used in the function `solve` from the `numpy` package. There's nothing wrong with using the `solve` function—it's well written, fast, and convenient. But it does nothing you haven't already done yourself if you've solved this exercise.

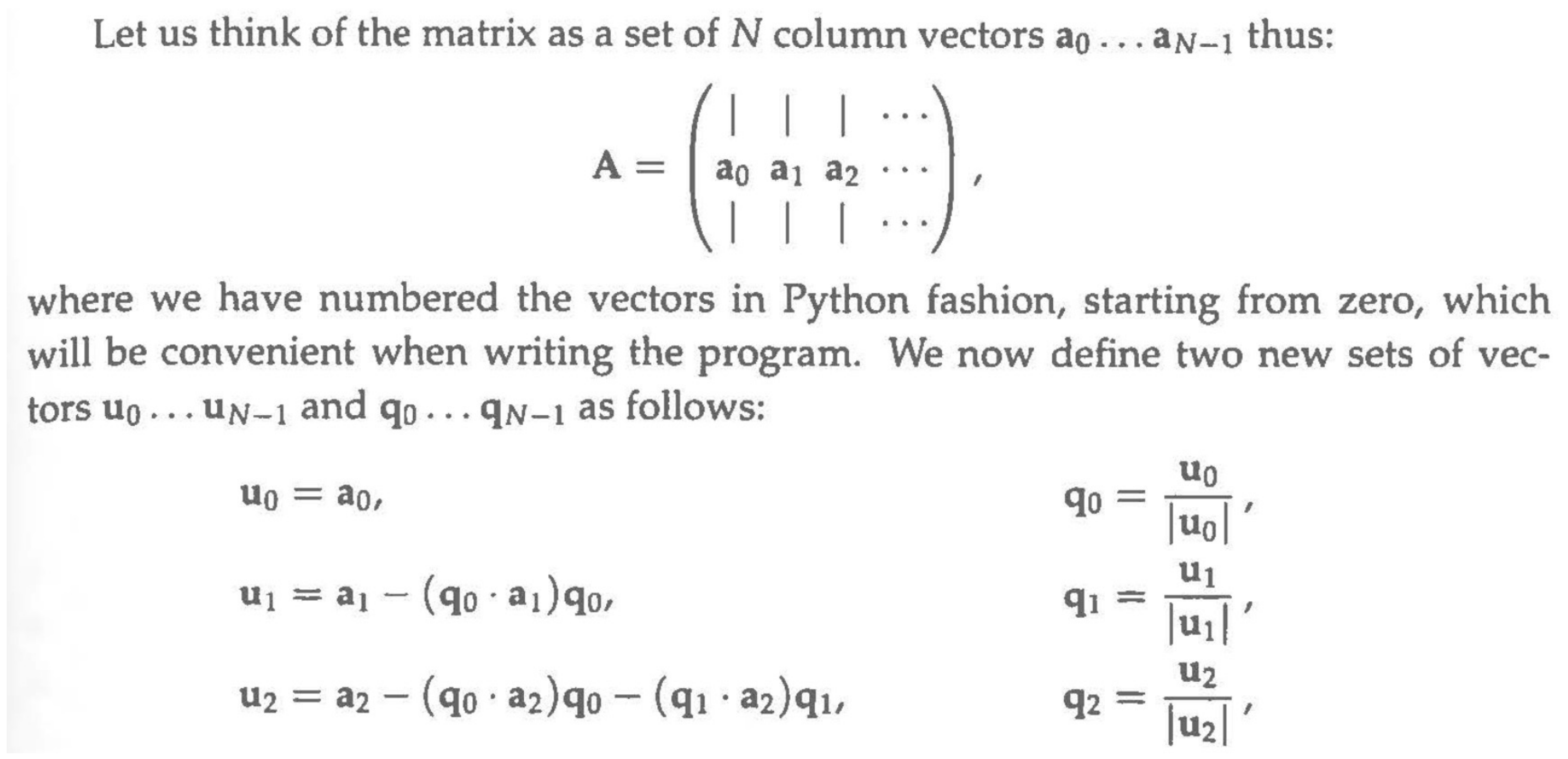
Problem 3.2



Exercise 6.8: The QR algorithm:

In this exercise you'll write a program to calculate the eigenvalues and eigenvectors of a real symmetric matrix using the QR algorithm. The first challenge is to write a program that finds the QR decomposition of a matrix. Then we'll use that decomposition to find the eigenvalues.

As described above, the QR decomposition expresses a real square matrix $\mathbf{A}$ in the form $\mathbf{A} = \mathbf{QR}$, where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is an upper-triangular matrix. Given an $N \times N$ matrix $\mathbf{A}$ we can compute the QR decomposition as follows.



Let us think of the matrix as a set of N column vectors $\mathbf{a}_0 \ldots \mathbf{a}_{N-1}$ thus:

$$\mathbf{A} = \begin{pmatrix} | & | & | & \cdots \\ \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \cdots \\ | & | & | & \cdots \end{pmatrix},$$

where we have numbered the vectors in Python fashion, starting from zero, which will be convenient when writing the program. We now define two new sets of vectors $\mathbf{u}_0 \ldots \mathbf{u}_{N-1}$ and $\mathbf{q}_0 \ldots \mathbf{q}_{N-1}$ as follows:

$$\mathbf{u}_0 = \mathbf{a}_0, \qquad \mathbf{q}_0 = \frac{\mathbf{u}_0}{|\mathbf{u}_0|},$$

$$\mathbf{u}_1 = \mathbf{a}_1 - (\mathbf{q}_0 \cdot \mathbf{a}_1)\mathbf{q}_0, \qquad \mathbf{q}_1 = \frac{\mathbf{u}_1}{|\mathbf{u}_1|},$$

$$\mathbf{u}_2 = \mathbf{a}_2 - (\mathbf{q}_0 \cdot \mathbf{a}_2)\mathbf{q}_0 - (\mathbf{q}_1 \cdot \mathbf{a}_2)\mathbf{q}_1, \qquad \mathbf{q}_2 = \frac{\mathbf{u}_2}{|\mathbf{u}_2|},$$

Problem 3.2 continued

and so forth. The general formulas for calculating $\mathbf{u}_i$ and $\mathbf{q}_i$ are

$$\mathbf{u}_i = \mathbf{a}_i - \sum_{j=0}^{i-1} (\mathbf{q}_j \cdot \mathbf{a}_i)\mathbf{q}_j, \qquad \mathbf{q}_i = \frac{\mathbf{u}_i}{|\mathbf{u}_i|}.$$

a) Show, by induction or otherwise, that the vectors $\mathbf{q}_i$ are orthonormal, i.e., that they satisfy

$$\mathbf{q}_i \cdot \mathbf{q}_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

Now, rearranging the definitions of the vectors, we have

$$\begin{aligned} \mathbf{a}_0 &= |\mathbf{u}_0|\, \mathbf{q}_0, \\ \mathbf{a}_1 &= |\mathbf{u}_1|\, \mathbf{q}_1 + (\mathbf{q}_0 \cdot \mathbf{a}_1)\mathbf{q}_0, \\ \mathbf{a}_2 &= |\mathbf{u}_2|\, \mathbf{q}_2 + (\mathbf{q}_0 \cdot \mathbf{a}_2)\mathbf{q}_0 + (\mathbf{q}_1 \cdot \mathbf{a}_2)\mathbf{q}_1, \end{aligned}$$

and so on. Or we can group the vectors $\mathbf{q}_i$ together as the columns of a matrix and write all of these equations as a single matrix equation

$$\mathbf{A} = \begin{pmatrix} | & | & | & \cdots \\ \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \cdots \\ | & | & | & \cdots \end{pmatrix} = \begin{pmatrix} | & | & | & \cdots \\ \mathbf{q}_0 & \mathbf{q}_1 & \mathbf{q}_2 & \cdots \\ | & | & | & \cdots \end{pmatrix} \begin{pmatrix} |\mathbf{u}_0| & \mathbf{q}_0 \cdot \mathbf{a}_1 & \mathbf{q}_0 \cdot \mathbf{a}_2 & \cdots \\ 0 & |\mathbf{u}_1| & \mathbf{q}_1 \cdot \mathbf{a}_2 & \cdots \\ 0 & 0 & |\mathbf{u}_2| & \cdots \end{pmatrix}.$$

(If this looks complicated it's worth multiplying out the matrices on the right to verify for yourself that you get the correct expressions for the $\mathbf{a}_i$.)

Problem 3.2 continued

Notice now that the first matrix on the right-hand side of this equation, the matrix with columns $\mathbf{q}_i$, is orthogonal, because the vectors $\mathbf{q}_i$ are orthonormal, and the second matrix is upper triangular. In other words, we have found the QR decomposition $\mathbf{A} = \mathbf{QR}$. The matrices $\mathbf{Q}$ and $\mathbf{R}$ are

$$\mathbf{Q} = \begin{pmatrix} | & | & | & \cdots \\ \mathbf{q}_0 & \mathbf{q}_1 & \mathbf{q}_2 & \cdots \\ | & | & | & \cdots \end{pmatrix}, \qquad \mathbf{R} = \begin{pmatrix} |\mathbf{u}_0| & \mathbf{q}_0 \cdot \mathbf{a}_1 & \mathbf{q}_0 \cdot \mathbf{a}_2 & \cdots \\ 0 & |\mathbf{u}_1| & \mathbf{q}_1 \cdot \mathbf{a}_2 & \cdots \\ 0 & 0 & |\mathbf{u}_2| & \cdots \end{pmatrix}.$$

b) Write a Python function that takes as its argument a real square matrix $\mathbf{A}$ and returns the two matrices $\mathbf{Q}$ and $\mathbf{R}$ that form its QR decomposition. As a test case, try out your function on the matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 4 & 8 & 4 \\ 4 & 2 & 3 & 7 \\ 8 & 3 & 6 & 9 \\ 4 & 7 & 9 & 2 \end{pmatrix}.$$

Check the results by multiplying $\mathbf{Q}$ and $\mathbf{R}$ together to recover the original matrix $\mathbf{A}$ again.

c) Using your function, write a complete program to calculate the eigenvalues and eigenvectors of a real symmetric matrix using the QR algorithm. Continue the calculation until the magnitude of every off-diagonal element of the matrix is smaller than 10^{-6}. Test your program on the example matrix above. You should find that the eigenvalues are $1, 21, -3$, and -8.