Solutions	Midterm exam in ELE 3781 Mathematics elective
Deadline	October 31st, 2022 at 1200

Question 1.
The function f returns the Leslie matrix defined by the vectors a (fertility rates) and b (survival rates). The function g computes the dominant eigenvalue of the given matrix, and returns the corresponding eigenvector (the Frobenius vector) as a real vector with column sum 1.

The next two lines compute the Leslie matrix of the given data, and its Frobenius vector as a column vector. The following three lines of code computes an initial vector $\mathbf{v}$, with constant value 10 in each state, as a column vector. Finally, the for-loop computes $A\mathbf{v}, A^2\mathbf{v}, \ldots, A^{11}\mathbf{v}$ from $\mathbf{v}$, forms the matrix with these vectors as columns, and the final line of code transforms this matrix into a DataFrame data type.

```
import numpy as np
import pandas as pd

a = np.array([0,0,0.3964,1.4939,2.1777,2.5250,2.6282,2.6749,2.6018,2.4419,2.1865,
              1.9044,1.7259,1.4918,1.2415,0.9522,0.7141,0.4618,0.2518,0.0901,0.0035])
b = np.array([0.94697,0.99665,0.99926,0.99899,0.99863,0.99817,0.99753,0.99667,0.99553,
             0.99399,0.99196,0.98926,0.98572,0.98107,0.97511,0.96748,0.95797,0.94631,
             0.93247,0.91649])
n = a.shape[0]

def f(a,b):
    matrix = np.zeros((n,n))
    matrix[0] = a
    for i in range(n-1):
        matrix[i+1,i] = b[i]
    return(matrix)

def g(matrix):
    c,d = np.linalg.eig(matrix)
    dominant = max(abs(c)).astype(np.complex)
    e = d[:,c == dominant].real
    e = e/e.sum()
    return(e)

A = f(a,b)
np.reshape(g(A),(1,n))

initial = 10
v = np.zeros((1,n)) + initial
v = np.reshape(v,(n,1))

m = 12
rows = v.shape[0]
for i in range(m-1):
    last = v.shape[1]
    w = np.reshape(v[:,last-1],(rows,1))
    v = np.append(v,A.dot(w),axis=1)

u = pd.DataFrame(np.transpose(v))
```

Question 2.

(a) We let $u = x^3$, write the equation as $u^2 - u + 1 = 0$, and use the quadratic formula to find u:

$$u = \frac{1 \pm \sqrt{1-4}}{2} = \frac{1 \pm i\sqrt{3}}{2}$$

We write these solutions in polar coordinates as $u_1 = e^{i\,60^\circ}$ and $u_2 = e^{-i\,60^\circ}$. With $x = e^{i\theta}$, consider the equation $x^3 = u_1$, which gives $3\theta = 60^\circ + k \cdot 360^\circ$, or $\theta = 20^\circ + k \cdot 120^\circ$ for $k = 0, 1, 2$. Then we consider $x^3 = u_2$, which gives $3\theta = -60^\circ + k \cdot 360^\circ$, or $\theta = -20^\circ + k \cdot 120^\circ$ for $k = 1, 2, 3$. Combining these cases, we find the solutions

$$\begin{array}{lll} x_0 = e^{i\,20^\circ} & x_1 = e^{i\,100^\circ} & x_2 = e^{i\,140^\circ} \\ x_3 = e^{i\,220^\circ} & x_4 = e^{i\,260^\circ} & x_5 = e^{i\,340^\circ} \end{array}$$

(b) The eigenvalues of A are given by the characteristic equation $-\lambda^3 + c_1\lambda^2 - c_2\lambda + c_3 = 0$, where $c_1 = \mathrm{tr}(A) = 6$, $c_2 = -1 + (-5) + (-1) = -7$ (the sum of the principal 2-minors), and $c_3 = |A| = 0$ (by a direct computation, or by the fact that A does not have maximal rank; see the computation of the null space of A below). Hence we get the equation

$$-\lambda^3 + 6\lambda^2 + 7\lambda = -\lambda(\lambda^2 - 6\lambda - 7) = -\lambda(\lambda - 7)(\lambda + 1) = 0$$

and the eigenvalues are $\lambda = 0$, $\lambda = 7$, and $\lambda = -1$.

(c) The (complex) null space $\mathrm{Null}(A)$ is given by the echelon form

$$A = \begin{pmatrix} 1 & 1-i & 2+i \\ 1+i & 1 & 3i \\ 2-i & -3i & 4 \end{pmatrix} \to \begin{pmatrix} 1 & 1-i & 2+i \\ 0 & -1 & -1 \\ 0 & -1 & -1 \end{pmatrix} \to \begin{pmatrix} 1 & 1-i & 2+i \\ 0 & -1 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

Hence z is free, $-y - z = 0$, or $y = -z$, and $x = -(1-i)(-z) - (2+i)z = (-1-2i)z$, and the complex null space is one-dimensional with base given by $(-1-2i, -1, 1)$.

Question 3.

(a) We find a right echelon form of A, and mark the right pivots:

$$A = \begin{pmatrix} 1 & 0 & 2 & 1 \\ 3 & 2 & 0 & -1 \\ 4 & 2 & 2 & 0 \\ 1 & -2 & 8 & 5 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 2 & 1 \\ 4 & 2 & 2 & 0 \\ 4 & 2 & 2 & 0 \\ -4 & -2 & -2 & 0 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 2 & 1 \\ 4 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The null space is given by the equations $x + 2z + w = 0$ and $4x + 2y + 2z = 0$, where x and y can be considered as free variables, hence $z = -2x - y$ and $w = -x - 2(-2x - y) = 3x + 2y$. It follows that the vectors in $\mathrm{Null}\, A$ are given by

$$(x, y, z, w) = (x, y, -2x - y, 3x + 2y) = x(1, 0, -2, 3) + y(0, 1, -1, 2)$$

(b) We find a right echelon form of A, and mark the right pivots:

$$A = \begin{pmatrix} 3 & 1 & 4 & -1 \\ 1 & 1 & 2 & 1 \\ 4 & 2 & 7 & 0 \\ -1 & 1 & 0 & 4 \end{pmatrix} \to \begin{pmatrix} 3 & 1 & 4 & -1 \\ 4 & 2 & 6 & 0 \\ 4 & 2 & 7 & 0 \\ 11 & 5 & 16 & 0 \end{pmatrix} \to \begin{pmatrix} 3 & 1 & 4 & -1 \\ 0 & 0 & -1 & 0 \\ 4 & 2 & 7 & 0 \\ 11 & 5 & 16 & 0 \end{pmatrix}$$

$$\to \begin{pmatrix} 3 & 1 & 4 & -1 \\ 0 & 0 & -1 & 0 \\ 4 & 2 & 0 & 0 \\ 11 & 5 & 0 & 0 \end{pmatrix} \to \begin{pmatrix} 3 & 1 & 4 & -1 \\ 0 & 0 & -1 & 0 \\ 4 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Since there is a right pivot in every column, we have that $\mathrm{Null}\, A = 0$.

Question 4.
See below for the python code for `rechelon(matrix)`. We get the following results when using this function on the matrices A and B:

(a) Reduced echelon form of A:

$$A \to \begin{pmatrix} 1 & 1 & 1 & 3 & -1 \\ 4 & 5 & 7 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(b) Reduced echelon form of B:

$$B \to \begin{pmatrix} 1 & 3 & 1 \\ -2 & -5 & 0 \\ 1.8 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

```
# Python code: right Gaussian elimination
import numpy as np

# Elementary row operations

def Rswitch(matrix,i,j):
    r = matrix[i-1].copy()
    matrix[i-1] = matrix[j-1]
    matrix[j-1] = r
    return(matrix)

def Rmult(matrix,i,c):
    matrix[i-1]=matrix[i-1]*c
    return(matrix)

def Radd(matrix,i,j,c):
    matrix[j-1]=matrix[j-1] + c*matrix[i-1]
    return(matrix)


# A simple version of rechelon that will find a right echelon form

def rechelon(matrix):
    # check the number of rows
    if matrix.shape[0]<=1:
        return(matrix)
    # get the rightmost column, nonzero positions
    n = matrix.shape[1]
    rcol = matrix[:,n-1]
    nz = np.arange(rcol.size)[rcol != 0]
    # when zero column, move to next column, if any
    if nz.size==0:
        if matrix.shape[1]<=1:
            return(matrix)
        rechelon(matrix[:,:n-1])
        return(matrix)
    # find first non-zero entry in column
    p=nz[0]
    if p!=0:
        Rswitch(matrix,1,p+1)
    # get zeros under the pivot
```

```
    for r in range(1,rcol.size):
        Radd(matrix,1,r+1,-matrix[r,n-1]/matrix[0,n-1])
    if matrix.shape[1]<=1:
        return(matrix)
    # if there is anything left to do after deleting first row/column, call recursively
    rechelon(matrix[1:,:n-1])
    return(matrix)

# Some tests that you can run

A = np.array([[1,1,1,3,-1],[1,2,4,7,3],[2,3,5,10,2]])
B = np.array([[1,3,1],[1,4,3],[2,3,5],[-1,10,2]]).astype(float)

# find a right echelon form of A
print(rechelon(A))

# find a right echelon form of B
print(rechelon(B))
```