Solutions	Midterm exam in ELE 3781 Mathematics elective
Deadline	November 1st, 2021 at 1200

Question 1.

When the function `f(x)` is called with a positive integer $x > 0$ as argument, the function will use recursion to run through a sequence of numbers with $x_0 = x$ and

$$x_{i+1} = \begin{cases} x_i/2 & \text{if } x_i \text{ is even} \\ 3x_i + 1 & \text{if } x_i \text{ is odd} \end{cases}$$

It will terminate when it reaches the number 1. For example, if $x = 10$, then we would run through the sequence $10, 5, 16, 8, 4, 2, 1$. The function `f(x)` will return the length of this sequence using recursion. For instance, in the example above, it would return $f(10) = 7$. Note that there is no guarantee that the sequence will terminate, but according to *Collatz conjecture* it should.

Question 2.

(a) We write $64 = 64e^{i\,0^\circ}$ in polar coordinates, and let $x = re^{i\theta}$. This gives $r^6 = 64$ and that $6\theta = k\cdot 360^\circ$, or $r = 2$ and $\theta = k\cdot 60^\circ$ for $k = 0, 1, 2, 3, 4, 5$. The complex solutions are therefore given by

$$\begin{aligned} &x_0 = 2 && x_1 = 2e^{i\,60^\circ} = 1 + i\sqrt{3} && x_2 = 2e^{i\,120^\circ} = -1 + i\sqrt{3} \\ &x_3 = -2 && x_4 = 2e^{i\,240^\circ} = -1 - i\sqrt{3} && x_5 = 2e^{i\,300^\circ} = 1 - i\sqrt{3} \end{aligned}$$

(b) 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 $u = e^{i\,120^\circ}$ and $u^2 = e^{i\,240^\circ}$. With $x = e^{i\theta}$, consider the equation $x^3 = u$, which gives $3\theta = 120^\circ + k\cdot 360^\circ$, or $\theta = 40^\circ + k\cdot 120^\circ$ for $k = 0, 1, 2$. Then we consider $x^3 = u^2$, which gives $3\theta = 240^\circ + k\cdot 360^\circ$, or $\theta = 80^\circ + k\cdot 120^\circ$ for $k = 0, 1, 2$. Combining these cases, we find the solutions

$$\begin{aligned} &x_0 = e^{i\,40^\circ} && x_1 = e^{i\,80^\circ} && x_2 = e^{i\,160^\circ} \\ &x_3 = e^{i\,200^\circ} && x_4 = e^{i\,280^\circ} && x_5 = e^{i\,320^\circ} \end{aligned}$$

(c) The eigenvalues of A are given by the characteristic equation $-\lambda^3 + c_1\lambda^2 - c_2\lambda + c_3 = 0$, where $c_1 = \mathrm{tr}(A) = 2$, $c_2 = 5 + (-8) + 7 = 4$ (the sum of principal 2-minors), and $c_3 = |A| = 0$. Hence we get the equation

$$-\lambda^3 + 2\lambda^2 - 4\lambda = -\lambda(\lambda^2 - 2\lambda + 4) = 0$$

and the complex eigenvalues are $\lambda = 0$ and $\lambda = (2 \pm \sqrt{4-16})/2 = 1 \pm i\sqrt{3}$. For $\lambda = 0$, we get the eigenspace $E_0 = \mathrm{Null}(A)$, and since

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

is an echelon form of A, we get that z is free, $5y + 7z = 0$, or $y = -7z/5$, and $x - y - 3z = 0$, or $x = y + 3z = -7z/5 + 3z = 8z/5$. Hence the vector $\mathbf{v}_1 = (8, -7, 5)$ is a base of E_0, and all real multiples $t\mathbf{v}_1$ with $t \in \mathbb{R}$ are real eigenvectors of A. For the eigenvalues $\lambda = 1 \pm i\sqrt{3}$, the eigenspaces are

$$E_{1+i\sqrt{3}} = \mathrm{Null}\begin{pmatrix} -i\sqrt{3} & -1 & -3 \\ 2 & 2 - i\sqrt{3} & 1 \\ 3 & 2 & -3 - i\sqrt{3} \end{pmatrix}$$

and

$$E_{1-i\sqrt{3}} = \mathrm{Null}\begin{pmatrix} i\sqrt{3} & -1 & -3 \\ 2 & 2 + i\sqrt{3} & 1 \\ 3 & 2 & -3 + i\sqrt{3} \end{pmatrix}$$

and we see that they do not contain any real eigenvectors. In the first case, if (x, y, z) is a real vector solution, then we have

$$-i\sqrt{3}x - y - 3z = 0, \quad 2x + (2 - i\sqrt{3})y + z = 0, \quad 3x + 2y + (-3 - i\sqrt{3})z = 0$$

The first equation implies that $x = 0$ since $-i\sqrt{3}x = y + 3z$ is real. Similarly, the second and third equation implies that $y = 0$ and $z = 0$. Hence there are no real eigenvectors (x, y, z). The second case is similar.

Question 3.

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

(a) Reduced echelon form of A:

$$A = \begin{pmatrix} 1 & 1 & 1 & 3 & -1 \\ 1 & 2 & 4 & 7 & 3 \\ 2 & 3 & 5 & 10 & 2 \end{pmatrix} \quad \to \quad \begin{pmatrix} 1 & 0 & -2 & -1 & -5 \\ 0 & 1 & 3 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(b) Reduced echelon form of B:

$$B = \begin{pmatrix} 1 & 3 & 1 \\ 1 & 4 & 3 \\ 2 & 3 & 5 \\ -1 & 10 & 2 \end{pmatrix} \quad \to \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

```
# Python code: Gauss-Jordan 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 Gauss that will find an echelon form

def Gauss(matrix):
    # check the number of rows
    if matrix.shape[0]<=1:
        return(matrix)
    # get the leftmost column, nonzero positions
    lcol = matrix[:,0]
    nz = np.arange(lcol.size)[lcol != 0]
    # when zero column, move to next column, if any
    if nz.size==0:
        if matrix.shape[1]<=1:
```

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


# A version of reduced that will find a reduced echelon form

def reduced(matrix):
    # first find an echelon form
    Gauss(matrix)
    m = matrix.shape[0]
    n = matrix.shape[1]
    # go through the rows in reverse order
    for r in range(m,0,-1):
        nz = np.arange(n)[matrix[r-1,:] != 0]
        # skip zero rows
        if nz.size == 0:
            continue
        # find the pivot position in row r
        p = nz[0]
        # make the pivot = 1
        Rmult(matrix,r,1/matrix[r-1,p])
        # make all entries over the pivot zero
        for i in range(1,r):
            Radd(matrix,r,i,-matrix[i-1,p]/matrix[r-1,p])
    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]])

# find reduced echelon form of A
print(reduced(A))

# find reduced echelon form of B
print(reduced(B))
```