Catálogo Grupal de Algoritmos

Integrantes:

- Bertha Brenes Brenes
  Carné: 2017101642
- Joshua Guzmán Quesada
  Carné: 2018084240

1 Tema 7: Valores y Vectores propios

1.1 Método de la Potencia

Código 1: Lenguaje Octave

```
function archivo_potencia
  clc;
  #pkg load symbolic;
  warning('off', 'all');
  resultado = potencia([1.5 0.5; 0.5 1.5], [0; 1]);
endfunction

function resultado = potencia(A, x0)
    % Metodo de la potencia para obtener el autovalor dominante de una matriz A
    % :param A: Matriz A
    % :param x: x0 inicial
    % :param tol: Tolerancia al fallo que debe tener el resultado
    % :return: Autovalor dominante
    norma_inf_ant = 0;  % Resultado de la norma de la iteracion anterior
    itermax = 100;
    xk = x0;
    tol = 10**-10;
    errors = [];
    ks = [];
    for k=1:itermax
        yk = A * xk;  % Calculo de y
        % Calculo de la norma infinita
        ck = norm(yk, inf);
        % Calculo de xk
        xk_n = yk / ck;
        % Calculo del error
        error = norm(xk_n - xk);
        % Reasignacion de xk
        xk = xk_n
        errors = [errors error];
        ks = [ks k];
        % Tolerancia del error
        if error<tol
            break;
        end
    end
```

```
    resultado = xk;
    plot(errors,ks, 'b')
    title("Grafica de puntos de comportamiento")
    ylabel("Iteraciones (k)")
    xlabel("Valores de los errores")
end
```

1.2 Método de la Potencia Inversa

Código 2: Lenguaje C++.

```
#include <armadillo>
#include <math.h>
#include "matplotlibcpp.h"
#include <iostream>
#include <tuple>

using namespace std;
using namespace arma;
namespace grafica = matplotlibcpp;


//                      FUNCION POTENCIA INVERSA

//    Parametros de entrada:
//    A= Matriz a calcular con el metodo de sistemas de Potencia Inversa
//    b = Vector inicial
//    tol= Tolerancia de la aproximacion
//    max_itr= Iteraciones maximas
//    Parametros de salida:
//    Aproximacion del metodo iterativo

tuple<double, mat, int, double>  potencia_inversa(arma::mat A, vec x_0, double tol, int max


    int filas = A.n_rows; //Se obtiene la cantidad de filas de la matriz
    //Se inicializan los valores para las iteraciones del metodo
    mat y_k, x_k = x_0, x_k_n; //Valores de xk y yk
    double c_k; //Variable para las iteraciones del metodo
    double error; //Variable para ir llevando la cuenta del error del metodo

    int interaciones; //Se lleva la cuenta de las iteraciones del metodo

    vector<double> resultados; //Se almacenan los resultados en un vector
    //para mejor legibilidad de los mismos

    while (interaciones < max_itr)// Iteraraciones
    {
        y_k = arma::solve(A, x_k); //Se resuleve el valor de yk con el vector propio y la m
        c_k = norm(yk, "inf");//Se toma la norma del valor de yk para que se guarde en ck d
        x_k_n = (1 / c_k) * y_k; //Se resuelve el valor de la iteracion xk

        error = norm(x_k_n - x_k); //Sr obtiene el erro del metodo
        resultados.push_back(error); //Se agrega el error al metodo
        x_k = x_k_n; //Se agrega el valor de xk

        //Si el error es menor que la tolerancia
        if (error < tol)
            break; //Se roompe el ciclo y se devuelve el error

        interaciones++;
    }
```

```
    return {c_k, x_k.t(), interaciones, error};

    //Se imprime el resultado graficamente

    plt::plot(resultados); //Se usa el vector de resultados para datos
    plt::title("Metodo Potencia Inversa iteraciones vs error"); //Se crea el titulo del gra
    plt::xlabel("Iteraciones k"); //Eje x con las iteraciones
    plt::ylabel("Error"); //Eje y con el error del metodo
    plt::show(); //Se muestra el resultado
}

int main(int argc, char const *argv[])
{
    //matriz simetrica definida positiva a la cual se le debenalcular el vector propio.
    mat A = {{ 3, -1, 0},
             { -1, 2, -1},
             { 0, -1, 3}};


    vec  x_0= {1,1,1}; //vector inicial aleatorio no nulo.
    x_0= x_0.t(); //Se transpone el valor del vector inicial

    potencia_inversa(A, x_0 ,10e-10,9); //Llamado a la funcion

    return 0;
}
```

1.3 Método QR

Código 3: Lenguaje Python.

```
from scipy import linalg as la
from numpy import matrix, zeros
import numpy as np
import matplotlib.pyplot as plt
"""
    Metodo QR para encontrar todos los autovalores de una matriz
    :param A: Matriz A
    :return: Matriz con todos los autovalores de la matriz A
    """
def metodo_qr(A):
    # Inicializo los valores iniciales
    tol = 10 ** -6
    A = matrix(A)
    A_ant = matrix(zeros(A.shape))
    [n,m] = A.shape
    iter_max = 100
    U = np.eye(n)
    ks = []
    errors = []
    for k in range(0,iter_max):
        # Calculo de Q y R
        Q, R = la.qr(A)
        # Almaceno el valor anterior de a
        A_ant = A
        # Calculo el nuevo valor de A
        A = R * Q
        # Calculo el valor de U
        U = U*Q
        # Se verifica la condicion de parada
        norma = la.norm(A_ant - A)
        # agrego a la lista los valores de error
        errors.append(norma)
        ks.append(k)
        # Calculo el error
        if norma < tol:
            break
    plt.rcParams.update({'font.size':14})
    plt.plot(ks,errors, marker='o',color='red')
    plt.ylabel('Iteraciones')
    plt.xlabel('Valores de error')
    plt.show()
    return A


resultado = metodo_qr([[0, 11, -5], [-2, 17, -7], [-4, 26, -10]])
print(resultado)
```