Aproximación de un Polinomio Característico

Luis Eduardo Robles Jiménez

0224969

Input

In [ ]:
```
#matrix = np.array([[3, 1, 5], [3, 3, 1], [4, 6, 4]]) #1, -10, 4, -40
#matrix = np.array([[3, 2, 4], [2, 0, 2], [4, 2, 3]]) #1, -6, -15, -8
matrix = np.array([[1, -1, 4], [3, 2, -1], [2, 1, -1]])
#matrix = np.array([[5, -2, 0], [-2, 3, -1], [0, -1, 1]])
```

Method

In [ ]:
```
Leverrier_Faddeev(matrix)
```

In [ ]:
```
Krilov(matrix, np.array([1, 0, 0]))
```

Krilov

In [ ]:
```
def Krilov(A, y = np.ones(0)):
    n = A.shape[0]
    b = np.empty((n, n))
    if y.size == 0: y = np.ones(n)
    b[0] = y
    print("Matrix:\n\n", A, "\n\nUsing vector:\n\n", y, "\n\nVectors calculated:\n")
    for i in range(1, n): b[i] = A @ b[i-1]
    print(b)
    a, s = np.linalg.solve(np.transpose(b), A @ b[n-1]), "λ^" + str(n)
    for i in np.flip(a):
        n -= 1
        s += " + " + str(-i) + "λ^" + str(n)
    return s
```

Leverrier Faddeev

In [ ]:
```
def Leverrier_Faddeev(A):
    print("Matrix:\n\n", A, "\n\n")
    n = A.shape[0]
    b, B, i = np.empty(n+1), np.empty((n+1, n, n)), np.identity(n)
    b[n], B[0] = 1, np.zeros((n, n))
    for k in range(1, n+1):
        B[k] = (A @ B[k-1]) + (b[n-k+1] * i)
        b[n-k] = -np.trace(A @ B[k])/k
    s = ""
    n += 1
    for i in np.flip(b):
        n -= 1
        if len(s): s += " + "
        s += str(i) + "*λ^" + str(n)

    return s
```

Run first

In [ ]:
```
import numpy as np
from sympy import *
x, lmbd = symbols("x"), symbols("lambda")
```