Universidad Autónoma de Querétaro

Facultad de Ingeniería

Tarea 2: Métodos cerrados para ecuaciones de una variable

Análisis numérico

Autor:
J.A. Salinas Sánchez
Febrero 2022

Índice general

Capítulo 1

Introducción

Ya sabemos lo que es un sistema de ecuaciones, los métodos de suma y resta, Cramer, eliminación Gaussiana, el pivoteo, el método de Gauss-Jordan y una gráfica. Así que procederemos a los problemas.

Capítulo 2

Metodología

Para las gráficas, se hizo un script por separado que generaba varios puntos de la función a evaluar y los graficaba. Para el problema 9.3, se hizo un script específico para multiplicar matrices mediante el algoritmo incluido en el módulo numpy. Finalmente, los métodos de Cramer,Gauss y Gauss-Jordan, así como todas las funciones auxiliares como el pivoteo, crear matrices o cambiar columnas y calcular determinates, se incluyen en el mismo script.

```
#coding:utf8
import numpy as np


def main():

    uk=int(input('Introduzca el n mero de inc gnitas: '))
    matriz1=np.zeros((uk,uk+1))
    tempmatrix=np.zeros(uk)

    for i in range(0,uk):
        for j in range(0,uk+1):
            if(j==uk):
                matriz1[i][j]=float(input(f'Introduzca el
   coeficiente del T.I: '))
            else:
                matriz1[i][j]=float(input(f'Introduzca el
   coeficiente de x{j}: '))


    l=['Gauss sin pivoteo[G]','Gauss con pivoteo[GP]','Gauss-Jordan
   [GJ]','Krammer[K]']
    print(l)
    metodo=str(input('Introduzca su m todo de resoluci n: '))
    if(metodo=='GP'):
        gausspivote(matriz1,tempmatrix,uk)
    elif(metodo=='G'):
        gauss(matriz1,tempmatrix,uk)
    elif(metodo=='GJ'):
        gaussjordan(matriz1,tempmatrix,uk)
    elif(metodo=='K'):
        krammer(matriz1,tempmatrix,uk)

    return(0)

def gauss(m1,tm,uk):
    i=0
    j=0
    k=0
```

```
37
38      for i in range(uk):
39          for j in range(i+1, uk):
40              ratio = m1[j][i]/m1[i][i]
41              for k in range(uk+1):
42                  m1[j][k] = m1[j][k]-ratio*m1[i][k]
43
44      tm[uk-1] = m1[uk-1][uk]/m1[uk-1][uk-1]
45
46      for i in range(uk-2,-1,-1):
47          tm[i] = m1[i][uk]
48          for j in range(i+1,uk):
49              tm[i] = tm[i] - m1[i][j]*tm[j]
50          tm[i] = tm[i]/m1[i][i]
51
52      print(tm)
53      return(0)
54
55  def gaussjordan(m1,tm,uk):
56      i=0
57      j=0
58      k=0
59
60      for i in range(uk):
61          for j in range(uk):
62              if i!=j:
63                  ratio = m1[j][i]/m1[i][i]
64                  for k in range(uk+1):
65                      m1[j][k] = m1[j][k]-ratio*m1[i][k]
66
67      for i in range(uk):
68          tm[i]=m1[i][uk]/m1[i][i]
69
70      print(tm)
71
72
73  def gausspivote(m1,tm,uk):
74      i=0
75      j=0
76      k=0
77
78      for i in range(uk):
79          for j in range(i+1, uk):
80
81              if(m1[i][i]==0):
82                  m1 = pivoteo(m1,uk,i)
83
84              ratio = m1[j][i]/m1[i][i]
85              for k in range(uk+1):
86                  m1[j][k] = m1[j][k]-ratio*m1[i][k]
87
88      tm[uk-1] = m1[uk-1][uk]/m1[uk-1][uk-1]
89
90      for i in range(uk-2,-1,-1):
91          tm[i] = m1[i][uk]
92          for j in range(i+1,uk):
93              tm[i] = tm[i] - m1[i][j]*tm[j]
94
95          tm[i] = tm[i]/m1[i][i]
96
97      print(tm)
98      return(0)
99
```

```
def krammer(m1,tm,uk):

    dets=[]
    respuestas=[]

    for k in range(uk+1):
        m2=changecols(m1,uk,k)
        dets.append(np.linalg.det(m2))

    for k in range(uk):
        respuestas.append(dets[k]/dets[uk])

    print(respuestas)
    return(0)


def changecols(m,uk,col):
    temp1=np.zeros((uk,uk+1))
    temp2=np.zeros((uk,uk))

    i=0
    index=uk

    while(i!=col):
        i+=1

    if(i!=uk):
        for j in range(uk):
            temp1[j][col]=m[j][index]
            temp1[j][index]=m[j][col]

        for i in range(uk):
            for j in range(uk+1):
                if (j!=col and j!=index):
                    temp1[i][j]=m[i][j]
                    temp1[i][j]=m[i][j]

    else:
        temp1=m.copy()
    for i in range(uk):
        for j in range(uk):
            temp2[i][j]=temp1[i][j]

    print(temp2)
    return(temp2)


def pivoteo(m,uk,row):
    temp=np.zeros((uk,uk+1))

    if(row==uk):
        index=row-1
    else:
        index=row+1

    for j in range(uk+1):
        temp[row][j]=m[index][j]
        temp[index][j]=m[row][j]

    for i in range(uk):
        if (i!=row and i!=index):
            for j in range(uk+1):
                temp[i][j]=m[i][j]
```

```
                temp[i][j]=m[i][j]

    return (temp)

if __name__=='__main__':
    main()
```

2.1. Problemas

2.1.1. 9.3

```
#coding:utf8

import numpy as np


def main():
    A=[[1,6],[3,10],[7,4]]
    B=[[1,3],[0.5,2]]
    C=[[2,-2],[-3,1]]

    print(np.dot(A,B))
    print(np.dot(A,C))
    print(np.dot(B,C))
    print(np.dot(C,B))









if __name__=='__main__':
    main()
```

2.1.2. 9.5

```
#coding:utf8

import matplotlib.pyplot as plt

def main():
    i=0
    x=[]
    y1=[]
    y2=[]
    answer=[]
    answerx=[]
    while(i<500):
        x.append(i)
        y1.append((120+1.1*i)/10)
        y2.append((174+2*i)/17.4)
        ea=abs((y1[i]-y2[i])/y2[i])*100
        if (ea<0.01):
            answer.append(y1[i])
            answerx.append(i)
        else:
            answer.append(0)
            answerx.append(i)
        i+=1
```

```
24
25      plt.plot(x,y1)
26      plt.plot(x,y2)
27      plt.scatter(answerx,answer)
28      plt.show()
29
30
31
32  if __name__=='__main__':
33      main()
```

2.1.3. 9.7

2.1.4. 9.9

2.1.5. 9.13