

David Alejandro Molano Vásquez

Parcial 1 Análisis Numérico

22 de agosto de 2018

1. Sea n el tamaño del problema y $f(n)$ la eficiencia del algoritmo, medida como el número mínimo de operaciones requeridas para resolver el problema

b) Diseñe, implemente en R o Python un algoritmo que le permita sumar los elementos de una matriz cuadrada A_n . Imprima varias pruebas, para diferentes valores de n y exprese $f(n)$ en notación $O(\)$

R / / Enlace al código: <https://repl.it/@elMolda/Suma-Cuadrados>

```
import numpy as np
def es_cuadrada (A):
    return all (len (row) == len (A) for row in A)

def suma_matriz_cuadrada(A):
    if es_cuadrada(A):
        suma = 0
        for r in range(len(A)):
            for c in range(len(A[0])):
                suma += A[r][c]
        return suma
    else:
        return -1

A = np.random.random_integers(0, 100, (3, 3))
print('Matriz')
print(A)
print('Suma: ' + str(suma_matriz_cuadrada(A)))

A = np.random.random_integers(0, 100, (4, 4))
print('Matriz')
print(A)
print('Suma: ' + str(suma_matriz_cuadrada(A)))

A = np.random.random_integers(0, 100, (5, 5))
print('Matriz')
print(A)
print('Suma: ' + str(suma_matriz_cuadrada(A)))

A = np.random.random_integers(0, 100, (20, 20))
print('Matriz')
print(A)
print('Suma: ' + str(suma_matriz_cuadrada(A)))
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
```

```
Matriz
```

```
[[19 62 33]
 [69  2 92]
 [13 88 87]]
```

```
Suma: 465
```

```
Matriz
```

```
[[64 54 19 32]
 [14 62 15 81]
 [ 3 42 58 40]
 [36 18 34 49]]
```

```
Suma: 621
```

```
Matriz
```

```
[[18 89 78 36 30]
 [10 82 53  6  5]
 [21 35 71 58 98]
 [34  1 69 19 36]
 [35 54 31 31 94]]
```

```
Suma: 1094
```

```
Matriz
```

```
[[ 30 23 47 74 82 27 26  9 43 69 68  2 77 86 69 87 90 22
   8 13]
 [ 95 77 12 60 83 27  6 85  0 46 88 53 12 20 100 63 17 43
  82 10]
 [ 10 66 76 91 90 45 39 92 96 16 11 25 92 81 48 74 76 42
  65 89]
 [ 62 41 70 72 13 55 71 49 31 64 69 18 82 54 33  9 23 71
  63 36]
 [100 12 40  9  1  9 85 100  7  8 71 97 90 52 75 11  0 77
  47 38]
 [ 55 13 53 37 55 17 91 96 47 53 87 100 51 13 35 30 77 35
  65  8]
 [ 25 89 21 89 48 56 69 49 100 65 69 41 61 99 43 75 53  9
  40 82]
 [ 60 81 95 54 52 62 66  2 49 10 28 60 29 58 20 41 22 52
  29 99]
 [ 17 51 12 60 96 35 71 74 49 70 44 66 53 12 68 34 38 85
  22 90]
 [ 66 16 71 33  0 89 29  8  2 58  5 44 85  1  8  8 77 31
  86 91]
 [  7 46 97 85 88 22 51  2 59 97 67 76 85 72  0 59 22 53
  11 92]
 [ 64 22 45  2 21 39  2 79  3 88 16 49 11 51 99 40 62 64
  14 14]
 [ 80 78 64 96 79 62  6 50  7 64 31  4 100 15 23 100 51 37
  93 85]
 [ 25 21 66 45 13 64 92 22 57 32 84 13 70 87 52 17 63 66
  16 60]
 [ 25 76 81 87 61 40 12 44 92 39 20 56 99 100 99 43 35 65
  32 61]
 [100 61 47 21 97 36 57 32 79 83 51 100 43 19 97 84  5 30
  67 74]
 [ 86 24 17  7  0 92 11 91 81 94 36 41 35  9 81 86 13 74
  21 50]
 [ 10 87 32 100 82 80 74 10  4 27 71  6 56 25 85 93 11 64
  83 81]
 [ 81 14 73 81 66 70  9 61 37 10 38 33  9 33 34 75 13 59
  68 19]
 [ 64 59 77 30 64 60 78 25 64 73 62 55 40  3 68 88 30 32
  47 93]]
```

```
Suma: 20436
```

La notación $O()$ se refiere al caso promedio de eficiencia de un algoritmo. Para este caso, se debe recorrer la matriz por medio de dos ciclos. Cada uno de estos se ejecutará n veces, siendo n la dimensión de la matriz cuadrada, sin que interfiera ningún otro factor, es decir siempre se ejecutarán todos los ciclos n veces. Por lo anterior la complejidad del algoritmo planteado es $O(n^2)$

2. En R: Sean $f(x) = \ln(x + 2)$ y $g(x) = \sin(x)$ dos funciones de valor real.

a) Utilice la siguiente formula recursiva con $E = 10^{-7}$ para el punto de intersección

Se deben igualar ambas funciones $f(x) = g(x)$, y luego se obtiene una nueva función tal que $f(x) - g(x) = 0$.

$$f(x) = \ln(x + 2) - \sin(x) = 0$$

R// Enlace al código: <https://repl.it/@elMolda/RecursivoRaiz>

```

r saved
f <- function(x) log(x+2) - sin(x)

raiz <- function(f,x0,x1){
  x2 <- (x0-((f(x0)*(x0-x1))/(f(x0)-f(x1))))
  print(x2)
  error = abs(x2-x1)
  if( error < 1e-7){
    return(x2)
  }else{
    return(raiz(f,x1,x2))
  }
}

punto <- raiz(f,-1.7,-1)

cat(formatC( c(punto,f(punto)), digits=7, width = -10, flag = " "), "\n")

```

```

using GNU R Version 3.5.0 (2018-04-23)
>
[1] -1.558969
[1] -1.712466
[1] -1.622542
[1] -1.630345
[1] -1.631458
[1] -1.631444
[1] -1.631444
x      f(x)
-1.631444 1.500577e-12
>

```

g) Gauss-Seidel: evalúe la matriz de transición de la matriz a dada.

Enlace al código: <https://repl.it/@elMolda/Gauss-Seidel>

```
def iteracion(a, b, x):
    n = len(x)
    for i in range(n):
        s = 0
        for j in range(n):
            if i!=j:
                s = s+a[i][j]*x[j]
            ##end if
        ##end for
        x[i] = (b[i]-s)/a[i][i]
    ##end for
    return x
##end def

def gauss(a, b, x, e, m):
    t = x.copy()
    for i in range(m):
        x = iteracion(a, b, x)
        d = maximoerror(x, t)
        if d<e:
            return x, i
        ##end if
        t = x.copy()
    ##end for
    return [[], m]
##end def

##-----
##main
##-----

m = 40
a = [
    [4,-1,-1,-1],
    [-1,4,-1,-1],
    [-1,-1,4,-1],
    [-1,-1,-1,4]
]
b = [1,1,1,1]
x = [0,0,0,0]
x, k = gauss(a, b, x, 0.0001, m)

for i in range(len(x)):
    print("x"+str(i+1)+"\t"+str(x[i]))

print("\nIteraciones: "+str(k))
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)  
[GCC 4.8.2] on linux
```

```
>
```

```
x1      0.9999018943198641  
x2      0.9999147582454708  
x3      0.9999259354126575  
x4      0.9999356469944981
```

```
Iteraciones: 16
```

```
> █
```