Exercici Difusió Calor ALN

Daniel Vilardell

Març 2021

a) Calculem primer el cost de l'eliminació Gaussiana en banda, i després el de la substitució enrera en banda. L'eliminació gaussiana te tres fors, un de mida n i dos de mida min(n - i, s) on i es la variable d'iteració del primer. El mes interior té una operació i el intermig en té 2. Sigui m = min(n, i + s) aleshores

$$\sum_{i=1}^{n}\sum_{j=i+1}^{m}(2+\sum_{j=i+1}^{m}1)\approx\sum_{i=1}^{n}\sum_{j=i+1}^{m}\sum_{j=i+1}^{m}1=\sum_{i=1}^{n-s}\sum_{j=1}^{s}\sum_{j=1}^{s}1+\sum_{i=n-s+1}^{n}\sum_{j=i+1}^{n}\sum_{j=i+1}^{n}1$$

Observem que el segon sumatori es el mateix que el que vam trobar al buscar el nombre d'operacions del metode de gauss pero en aquest cas en una matriu de mida sxs, per tant el ordre del nombre d'operacions es $\Theta(s^3)$.

$$\#operacions \approx (n-s)s^2+\Theta(s^3)$$

Si considerem que $n >> s$ aleshores

$$\#operacions = \Theta(ns^2)$$

El cost de aplicar a substitució enrere, si considerem altre cop $n >> s$ es el següent

$$\#operacions = \sum_{i=1}^{n}\sum_{j=i+1}^{m}1=\Theta(ns)$$

Cal apuntar però que el simple fet de crear la matriu ja es del ordre de $\Theta(n^2)$ i si, tal com hem considerat, $n >> s \implies n >> s^2$, aleshores el nombre d'operacions total el domina el fet de crear la matriu mes que resoldre el sistema.

$$\#operacions_{tot} = \Theta(n^2)+\Theta(ns^2)+\Theta(ns)=\Theta(n^2)$$

Si nomes ens guardessim la banda de la matriu i no ens guardessim els 0 aleshores això no seria problema, tot i que al codi de difusió de calor no es fa així.

b)

```
def eliminacioGaussiana_banda(A, b,s):
    n=len(A)
    if b.size != n:
        raise ValueError("Error: les dimensions de A i b no coincideixen", b.size, n)
    for k in range(n-1):
        for i in range(k+1, min(n,k+s)):
            alpha = A[i,k]/A[k,k]
            A[i,k]=0
            A[i, k+1:min(n, k + s)] -= alpha*A[k,k + 1:min(n, k + s)]
#            for j in range(k+1, min(n,k+s)):
#                A[i,j] = A[i,j] - alpha*A[k,j]
            b[i] = b[i] - alpha*b[k]
    x = substitucioEnrera_banda(A,b,s)
    return x

def substitucioEnrera_banda(A, b, s):
    n = len(A)
    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        banda = b[i]
        banda -= np.dot(A[i,i+1:min(n,i+s)], x[i+1:min(n,i+s)])
#        for j in range(i + 1, min(n, i + s)):
#            banda -= x[j]*A[i,j]
        x[i] = banda/A[i,i]
    return x
```

A dins del main, per a comprobar que la solució era correcta hem introduit el següent codi.

```
b = fred.copy()
A_ini = Ared.copy()
ured = eliminacioGaussiana_banda(Ared, fred, 5*nRefinament)

b2 = A_ini@ured
sonIguals = max(abs(b - b2))<1e-9
print("Sistema resolt:", sonIguals)
print("Temps: ", time.time() - t0)
#Output:
#    Sistema resolt: True
#    Temps:  0.0015139579772949219
```

c) En primer lloc hem afegit un for que ens repeteix el codi 10 cops, i cada cop recopilavem les dades, aquí podem veure un resum de 10 iteracions del algoritme, cadascuna amb un refinament diferent. Totes les dades son aplicant l'algoritme de eliminació Gaussiana en banda.

```
Refinament:  1
   Temps:  0.001341104507446289
   Dim sistema:  8
   Coefs no nuls abans:  28
   Coefs no nuls despres:  27
   Sistema resolt: True
Refinament:  2
   Temps:  0.0030608177185058594
   Dim sistema:  45
   Coefs no nuls abans:  197
   Coefs no nuls despres:  377
   Sistema resolt: True
Refinament:  3
   Temps:  0.011131763458251953
   Dim sistema:  112
   Coefs no nuls abans:  516
   Coefs no nuls despres:  1497
   Sistema resolt: True
Refinament:  4
   Temps:  0.025218725204467773
   Dim sistema:  209
   Coefs no nuls abans:  985
   Coefs no nuls despres:  3821
   Sistema resolt: True
Refinament:  5
   Temps:  0.05974102020263672
   Dim sistema:  336
   Coefs no nuls abans:  1604
   Coefs no nuls despres:  7754
   Sistema resolt: True
Refinament:  6
   Temps:  0.17484617233276367
   Dim sistema:  493
   Coefs no nuls abans:  2373
   Coefs no nuls despres:  13704
   Sistema resolt: True
Refinament:  7
   Temps:  0.2126011848449707
   Dim sistema:  680
   Coefs no nuls abans:  3292
   Coefs no nuls despres:  22069
   Sistema resolt: True
Refinament:  8
   Temps:  0.34513306617736816
   Dim sistema:  897
   Coefs no nuls abans:  4361
   Coefs no nuls despres:  33249
   Sistema resolt: True
Refinament:  9
   Temps:  0.44358301162719727
   Dim sistema:  1144
   Coefs no nuls abans:  5580
   Coefs no nuls despres:  47647
   Sistema resolt: True
Refinament:  10
   Temps:  0.6282551288604736
   Dim sistema:  1421
   Coefs no nuls abans:  6949
   Coefs no nuls despres:  65666
   Sistema resolt: True
```

A partir d'aquestes dades hem graficat el logaritme del temps de calcul en funció del logaritme de la dimensió del sistema i el logaritme del nombre de coeficients no nuls en funció del logaritme de la dimensió.

En la primera i segona grafica podem concloure que el creixement es aproximadament lineal (amb excepcio dels refinaments petits que te un pendent inferior). Tot i això podem veure que el pendent en la funció de Gauss sense banda es de aproximadament $6 \cdot 10^{-3}$ mentres que en el Gauss amb banda el pendent es de aproximadament $4 \cdot 10^{-4}$, cosa que mostra la gran millora respecte al altre metode.

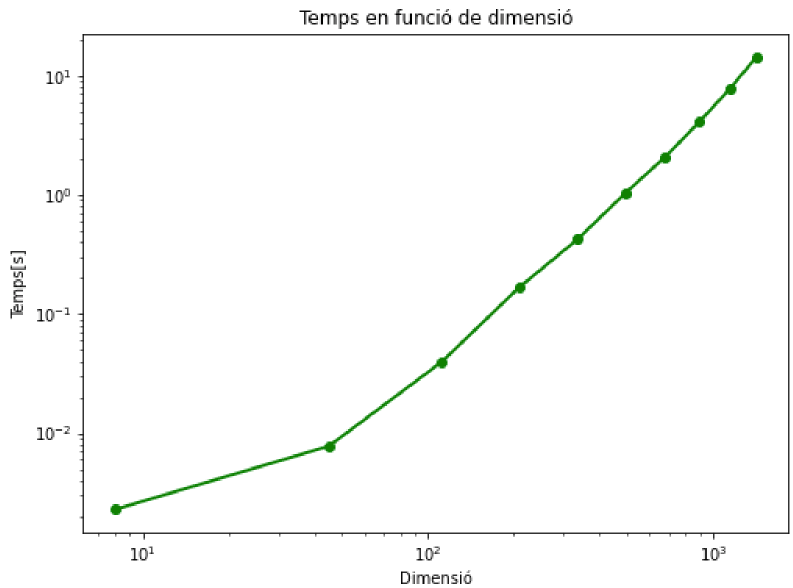

Figura 1: Gauss sense banda

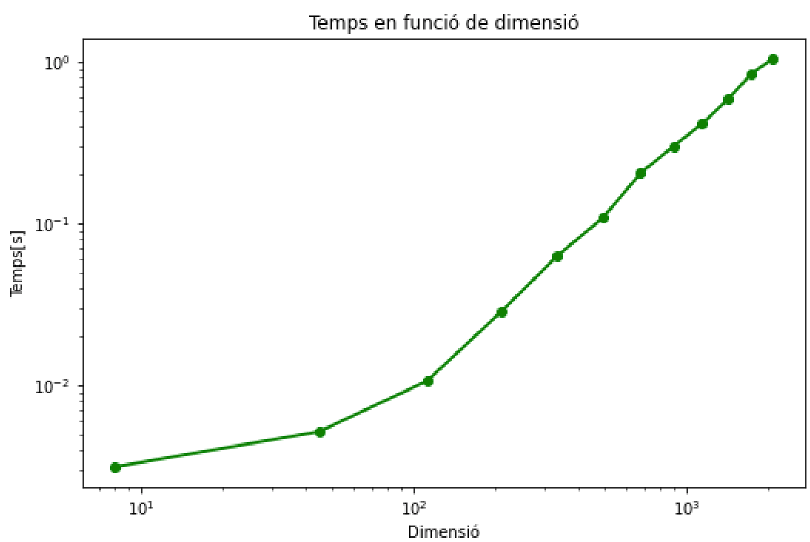

Figura 2: Gauss amb banda

Podem veure per altra banda que el nombre de coeficients no nuls abans de cridar la funció d'eliminació Gaussiana i despres varia bastant, arribant a tenir una proporcio amb el refinament 14 de $\frac{2 \cdot 10^5}{1.5 \cdot 10^4} \approx 15$. Si calculem el pendent veiem que la primera grafica te pendent de 3 mentres que la segona te un pendent de 100.

Cal apuntar que la funció np.count_nonzero conta el nombre d'elements no nuls, i durant el metode de Gauss hi poden haver errors de precisió. Cal per tant contar tots els coeficients que son majors a un epsilon.

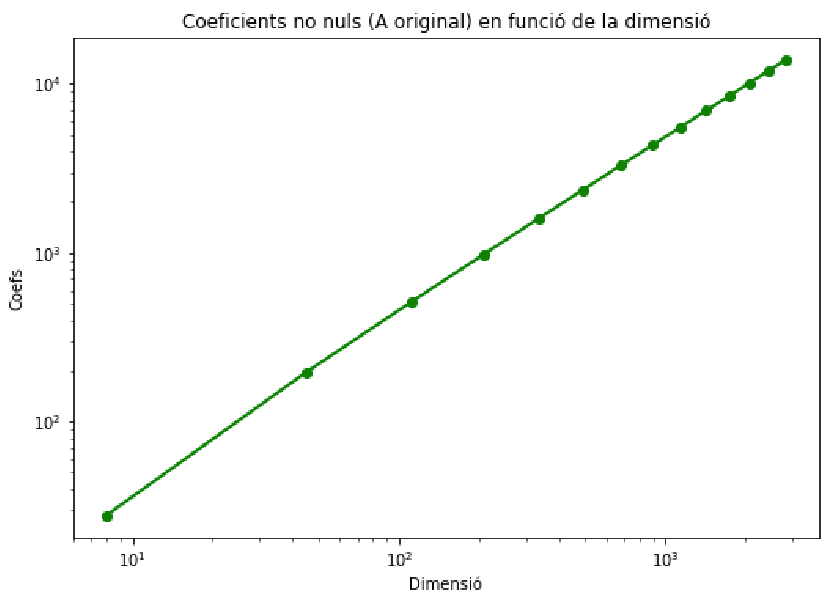

Figura 3: Coeficients no nuls en la matriu original

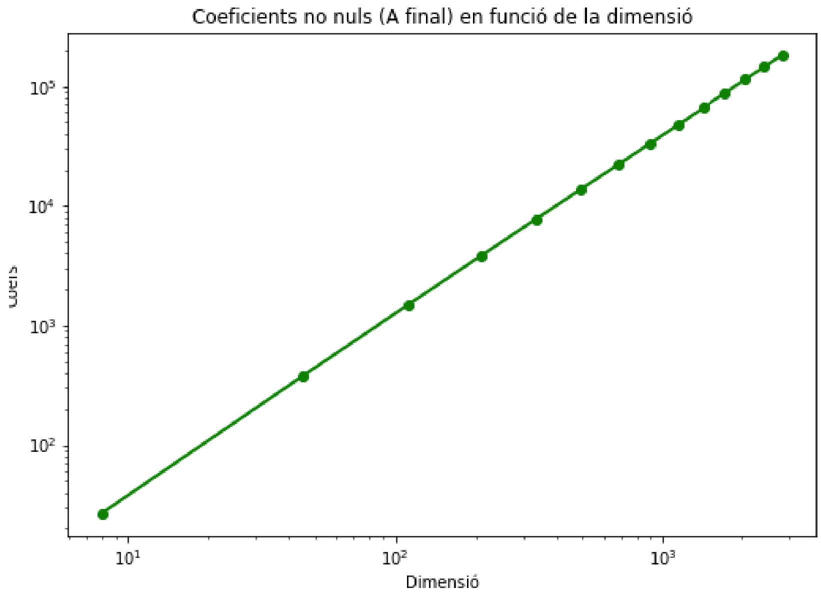

Figura 4: Coeficients no nuls en la matriu final