Optimització de la qualitat d'una malla

Pol Tibau
Universitat Politècnica de Catalunya

October 16, 2023

0.1 Introducció

En aquesta pràctica tractarem d'utilitzar mètodes numèrics per resoldre sistemes d'equacions no lineals amb l'objectiu de minimitzar la distorció d'una malla. En concret posarem en pràctica el mètode de Newton-Raphson aplicat al gradient de la funció de distorsió de la malla, donant així un problema de zeros de funcions.

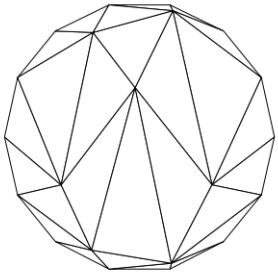

Figure 1: Malla inicial

0.2 Funció de distorsió

Començarem construint la funció que calcula la distorsió de la malla al codi. A partir de la seva definició analítica i les matrius X i T, la escriurem a Python com:

```
def calculaDistorsioMalla(X, T):
    suma = 0
    for i in range(np.shape(T)[0]):
        D = np.zeros((2,2))\\
        D[0] = X[T[i][1]]-X[T[i][0]]
        D[1] = X[T[i][2]]-X[T[i][0]]
        d_phi = np.matmul(np.transpose(D),A)
        suma +=((norm(d_phi,'fro')**2)/(2*abs(det(d_phi))))**2
        dist = np.sqrt(suma)
    return dist
```

(On hem escrit `np.linalg.norm` com norm i `np.linalg.det` com det)

Podem veure usant aquesta funció aplicades a les matrius inicials X,T que la distorció de la malla inicial és de 17.332030989939604.

0.3 Plantegem minimitzar la funció de distorsió

Com hem dit, usarem el mètode de Newton-Raphson considerant que volem trobar un zero del gradient de la funció de distorsió. Per tant, el nostre sistema d'equacions no lineal ve donat per:

$$\nabla F = 0 \tag{1}$$

On F és la que hem mencionat anteriorment com funció de distorsió. En el codi la calcularem usant el fitxer cedit com `differentiation.py` on tenim la funció `derivadaNumerica` que ens retorna el jacobià de la funció donada.

0.4 Aplicació del mètode Newton-Raphson

De nou usarem el fitxer `differentiation.py` on també tenim la funcio `hessiana Numerica` la qual a partir de la funció F ens retorna la seva hessiana. Això ens serà útil ja que correspon a la jacobiana de la funció ∇F que correspon a una matriu quadrada amb la que podem usar el mètode de Newton-Raphson.

Com sabem, cada iteració del mètode de Newton-Rapshon ve donat per:

$$J(\overrightarrow{x}^k)\Delta\overrightarrow{x}^k = -\overrightarrow{f}(\overrightarrow{x}^k) \rightarrow \Delta\overrightarrow{x}^k \tag{2}$$

$$\overrightarrow{x}^{k+1} = \overrightarrow{x}^k + \Delta\overrightarrow{x}^k \tag{3}$$

En el nostre cas, $J(\overrightarrow{x}^k)$ es correspon a la Hessiana de F, i per obtenir $\overrightarrow{f}(\overrightarrow{x}^k)$ sols hem d'evaluar ∇F amb $\overrightarrow{x}^k$. Com que és molt costós calcular la inversa del $J(\overrightarrow{x}^k)$, el que farem a cada iteració és resoldre el sistema lineal d'equacion on les incògnites són cadascun dels termes de $\Delta\overrightarrow{x}^k$. Per resoldre el sistema usarem la funció de la llibreria Numpy `numpy.linalg.solve`. En el codi del programa quedaria redactada cada iteració amb la següent funció:

```
def iteracio(I,y,R,E):
    R = derivadaNumerica(F,y)
    J = hessianaNumerica(F,y)
    I = np.linalg.solve(J,-R)
    y += I
    E.append(np.log10((norm(I)/norm(y))))
return R,E
```

(On hem escrit `np.linalg.norm` com norm)

On R és el residu, J la jacobiana de ∇F, I l'increment $\Delta\overrightarrow{x}^k$ i E és el vector que guarda el logaritme en base 10 de l'error relatiu de cada iteració. y es correspon a $\overrightarrow{x}^k$.

0.5 Obtenció de la malla final

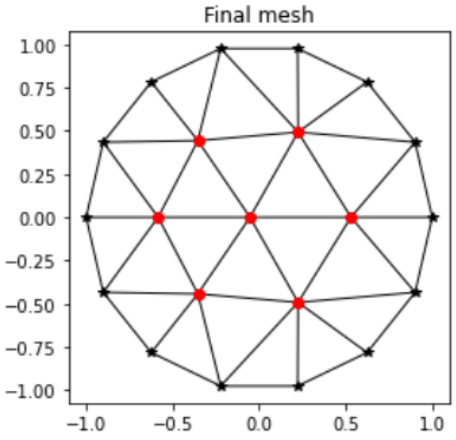

Figure 2: Malla final

Podem observar que obtenim una malla amb uns triangles "molt més equilàters" que els de la malla inicial. Si evaluem ara la funció de distorsió observem que val: 5.358517623300091 i la posició del primer node interior és: [5.30207045e-01, 3.19597139e-11].

0.6 Gràfica de convergència

A partir del vector E que hem obtingut a mesura de iterar el mètode de Newton-Raphson, podem obtenir un grafic del logaritme en base 10 de l'error relatiu. Hem aplicat una tolèrància de 510^{-7}, obtenint així 11 iteracions fins la convergència. El gràfic mencionat és el següent:

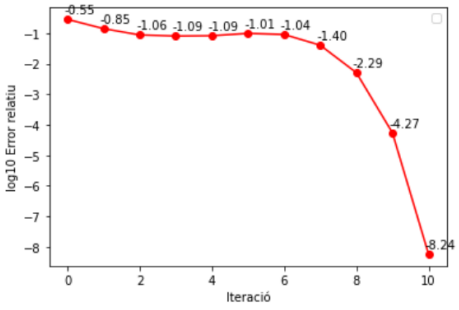

Figure 3: Gràfic del logaritme en base 10 de l'error relatiu

0.7 Tipus de convergència

Finalment, podem concloure que en aquest cas el mètode de Newton té convergència quadràtica ja que a partir de la iteració 7 podem observar que l'error es redueix de forma quadràtica, això principalment es deu a que la Jacobiana (la Hessiana de F) és regular. Altrament si observem el vector que ens dona el nombre de la condició de la matriu Jacobiana veurem que aquest a les primeres iteracions és elevat, però a partir de la setena iteració es redueix considerablement, causa també de que aparegui aleshores la convergència quadràtica.

Vector de matriu de condició: [17027.0684, 6192.300720869473, 1268.5211, 278.9688, 66.1817, 17.9053, 6.0543, 3.5673, 3.4819, 3.4887, 3.4887]