Treball de Fi de Grau

## Grau en Tecnologies Industrials

# Càlcul del corrent crític en materials superconductors

**Implementació out-of-core de l'aïllament de la solució**

**MEMÒRIA**


**Autor:** Wilhelm Auffermann
**Director:** Jaume Amorós

**Convocatòria:** Gener 2017


Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

# Resum

L'objectiu d'aquest treball és resoldre una problemàtica a l'hora de solucionar computacionalment sistemes d'equacions tan grans (Matrius de fins a 64Gb d'espai en el disc) que no caben a la memòria RAM del ordinador. El codi creat té com a finalitat evitar errors del tipus "out of memory"-que l'ordinador no pot manipular una matriu tan gran perquè no hi cap a la memòria que té disponible, o errors del tipus "maximum size allowed exeded"-limitacions en el disseny del programa que manipula les matrius-.

Per dur a terme un projecte d'aquestes característiques s'ha de recórrer a l' utilització d'algoritmes anomenats out-of-core. Aquesta tècnica permet fer operacions a matrius sense tenir-les carregades en memòria.

Per exemple, intentar fer la descomposició qr() amb Matlab per una matriu de 4Gb dóna l'error de manca de memòria per crear-la, ja que no hi cap a la memòria RAM de l'ordinador en qüestió. En canvi, amb el nostre programa, es pot franquejar aquesta limitació.

```
>> [Q,R]=qr(randi(25000,20000));
Error using randi
Out of memory. Type HELP MEMORY for your options.

>>
```

L'abast d'aquest treball és principalment crear una rutina out-of-core que s'aplica al terme A i B del sistema d'equacions següent per simplificar significativament l'aïllament out-of-core posterior de la solució X.

$$\begin{bmatrix} & A & \end{bmatrix}\begin{bmatrix} X \end{bmatrix} = \begin{bmatrix} B \end{bmatrix}$$

A matriu coneguda m·n
B vector conegut m
X vector incògnita n

La rutina prèvia al càlcul de la solució que s'ha adaptat per que sigui out-of-core és la descomposició QR per reflexió d'hiperplans de Householder. D'altre banda, també s'ha adaptat l'operació d'aïllament de la solució per funcionar out-of-core. S'ha programat en llenguatge MATLAB ja que és un llenguatge molt enfocat en la optimització de càlculs matricials, el que ens permetrà fer uns tipus de càlculs que trigarien un temps molt més significatiu si fos programat en un altre llenguatge. Tot el codi desenvolupat en aquest projecte es pot utilitzar també en Octave.

Parlant en termes corrents, la tàctica out-of-core és utilitzar el disc dur per substituir la memòria RAM. Això permet que un ordinador normal pugui fer una tasca molt demandant en recursos, altrament reservada a sistemes d'ordinadors, o a parts de superordinadors. L'únic contrapunt és que, *per se*, aquest mètode triga un temps significatiu.

# Sumari

# 1. Introducció

## 1.1. Presentació dels mètodes out-of-core

L'Out-of-core és un algoritme que permet accedir a les dades només quan es necessiten, és a dir, eficientment i sense comprometre la viabilitat de l'operació, ja que s'evita sobrepassar les capacitats de la memòria RAM. Actualment s'empren en multitud d'aplicacions. Per exemple Google Maps utilitza aquest mètode ja que té moltíssimes dades geogràfiques, que poden ocupar fins a Terabytes [Wikipedia, Out-of-core]. Així compartimenta les dades i en fa un ús incomparablement més eficient. En aquest treball s'utilitzarà aquest mètode en matrius de fins a vuit mil milions de coeficients (64Gb amb dades de tipus 'double'). Altres treballs i implementacions d'àlgebra lineal out-of-core són per exemple l'article de Sivan Toledo [Toledo,1999], en el que explica una miscel·lània d'aplicacions de l'out-of-core a l'àlgebra numèrica lineal.

Toledo recull tàctiques per agilitzar mètodes out-of-core com són la planificació prèvia i l'organització de la utilització de les dades, ja que prenen un paper principal en la tàctica de programació d'aquest mètode. És a dir, al contrari de la programació "comuna" on es declaren totes les variables al principi de la funció, en el cas de out-of-core s'ha de seguir una estratègia per evitar que algunes d'aquestes variables s'acumulin.

Seguint aquesta estratègia, s'ha d'estudiar de quina o quines maneres es llegiran i guardaran transitòriament a la memòria RAM les porcions de dades, i escollir-ho pensant en que aquesta operació serà de les més crítiques, i s'ha d'arribar a fer un balanç entre la mida d'aquesta porció i el número d'accessos, per optimitzar l'operació.

La computació d'aquest treball s'ha implementat amb la CPU, la RAM i el HDD, però s'ha demostrat [Marqués et al, 2009] que la GPU (Graphics processing unit) pot accelerar la solució de metodologies out-of-core, en concret de la llibreria FLAME. La llibreria FLAME és un projecte que articula la derivació sistemàtica d'algoritmes basats en bucles tot i aportant prova dinàmicament de que aquests algoritmes són correctes. Una possible continuació d'aquest projecte és migrar la descomposició de Householder a les GPU.

Similarment, podem trobar aplicacions similars de metodologies parcialment in-core i out-of-core en llenguatges diferents com pot ser la descomposició de SVD i descomposició QR (fins a 32GB)[Rabani et al, 2001]. L'algoritme del treball que ens ocupa s'ha testejat fins a 64Gb i s'ha dissenyat amb MATLAB, que permet optimitzar els càlculs matricials i "for"'s gràcies a la vectorització de bucles. Un altre exemple similar al nostre projecte són les factoritzacions out-of-core de LU, QR i Cholesky [D'Azevedo, 2000], que s'executen en una extensió de ScaLPACK, també escrita en FORTAN. Finalment, un molt bon exemple de teoria de la metodologia out-of-core és el llibre de Jeffrey Scott Vitter [Vitter, 2008].

## 1.2. Programari emprat

En aquest projecte s'ha emprat com a eina única Matlab en un ordinador principal Macbook 2009 (6,1- MC207LL/A ) amb SSD i 8Gb de RAM, i un auxiliar Windows 7 2009 Intel Core i5-3427U--2.3GHZ amb HDD i 8Gb de RAM, tots dos amb la mateixa versió 2012a (32bits). Aquests dos equips no són ni actuals ni molt potents, el que permet garantir que d'ara en endavant la utilització del programari desenvolupat en aquest projecte serà cada cop més ràpida, i que, en qualsevol cas, no hi hauran casos de manca de recursos. També es pot executar el codi amb Octave per a sistemes unix.

### 1.2.1. Característiques de la memòria de l'ordinador principal

```
Maximum possible array:            2046 MB (2.146e+09 bytes) *
Memory available for all arrays:   3430 MB (3.596e+09 bytes) **
Memory used by MATLAB:              349 MB (3.662e+08 bytes)
Physical Memory (RAM):             8056 MB (8.447e+09 bytes)
*  Limited by contiguous virtual address space available.
** Limited by virtual address space available.
```

### 1.2.2. Característiques de la memòria de l'ordinador auxiliar

```
Maximum possible array:            2046 MB (2.146e+09 bytes) *
Memory available for all arrays:   3430 MB (3.596e+09 bytes) **
Memory used by MATLAB:              349 MB (3.662e+08 bytes)
Physical Memory (RAM):             8056 MB (8.447e+09 bytes)

*  Limited by contiguous virtual address space available.
** Limited by virtual address space available.
```

## 1.3. Planificació del temps

El projecte ha constat en les següents fases, organitzades en la Taula 1.a. Les fases 1 i 2 han permès familiaritzar-se amb la dinàmica de l'algoritme Householder i la tercera fase ha permès fer una sèrie de test per verificar el perfecte funcionament del codi, i monitoritzar-ne les característiques (com per exemple la velocitat i els límits de funcionament).

*Taula 1*

| FASE1 | FASE2 | FASE3 |
|---|---|---|
| Creació de Householder in-core<br><br>Verificació i correcció | Creació d'eines auxiliars de lectura-escriptura al disc dur<br><br>Verificació i correcció | Creació de Householder out-of-core<br><br>Verificació, correcció i test real |

En el Gantt es pot observar el temps dedicat a cada part del projecte. Al ser un projecte de programació, no s'estima rellevant detallar cada errada/correcció del codi ja que és un procés de prova i error constant.

*Taula 2*

| | 2016 | | | | | | | | | | | | | 2017 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Setmana | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 1 | 2 | 3 | 4 |
| FASE 1 | 3 setmanes | | | | | | | | | | | | | | | | |
| FASE 2 | | | | 6 setmanes | | | | | | | | | | | | | |
| FASE 3 | | | | | | | | | | 6 setmanes | | | | | | | |
| Memòria | | | | | | | | | | | | | 3 setmanes | | | | |

## Descripció de Householder

En aquest projecte es farà una implementació del mètode out-of-core a la resolució d'un sistema lineal via reflexions per hiperplans de Householder. Es tria Householder ja que és un mètode de descomposició QR [Golub et al, 1996] que facilita la resolució de sistemes lineals amb menys propagació d'error ja que Householder no augmenta el nombre de condició [Golub et al, 1996]. L'error es manté doncs estable durant l'aplicació de la factorització, el que permet seguir tenint una bona fidelitat dels resultats. Per poder justificar que Householder és el mètode més adequat, s'ha elaborat la Taula 1.c comparativa en la que es poden veure els avantatges i inconvenients dels mètodes, acompanyat d'una breu descripció.

Acadèmicament, la utilització del mètode de Householder per aquest projecte ha sigut adient, ja que el concepte de les reflexions de plans/hiperplans s'explica en l'assignatura de Geometria, el que agilitza l'assoliment dels prerequisits necessaris per poder elaborar el codi.

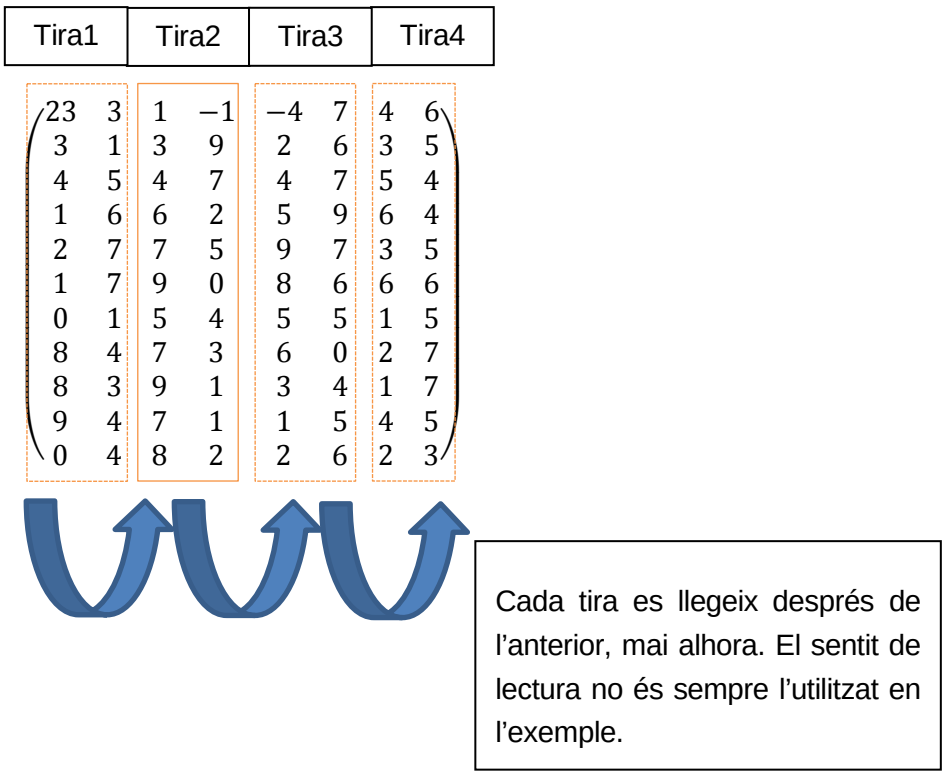
Es tria també Householder ja que és un procés fàcilment adaptable a la metodologia out-of-core, tot i que presenta una càrrega d'operacions gran. Més endavant, en l'explicació del codi, es farà especial menció a les estratègies out-of-core que s'han emprat.

| | Gram-Schmidt | Householder | Rotacions de Givens |
|---|---|---|---|
| Breu explicació dels mètodes utilitzats en la factorització A=QR | Aquest mètode es basa en projeccions geomètriques que resulten en un canvi de base que permet obtenir els elements nuls necessaris per obtenir la matriu R. | Aquest mètode es basa en reflexions en hiperplans que resulten en un canvi de base que permet transformar la matriu original en la matriu triangular superior R. (Explicat en més detall més endavant ja que és el mètode triat). | Aquest mètode aplica rotacions a la matriu original per crear zeros a sota de la diagonal i convertir-la en la matriu triangular R. |
| Avantatges | Implementació més senzilla. Pot servir correctament en casos simples o quan es necessita un resultat aproximatiu i preliminar ràpid. | Molta més estabilitat numèrica. En el cas pràctic, l'error es manté estable. També és l'algoritme amb el disseny més simple. Un altre avantatge és que es pot fer la factorització QR per m ≥ n (sobre-determinat). | També conserva la norma dels vectors, així que l'error està controlat. En el cas pràctic, només s'utilitza un model simplificat i només si la matriu original té ja bastants elements nuls a sota de la diagonal. |
| Inconvenients | L'ortogonalització en si mateixa sol canviar la norma dels vectors quan és aplicada, el que provoca inherentment una "no estabilitat" de l'error, que a la pràctica tendeix a augmentar. Un altre inconvenient és que només es pot utilitzar en matrius quadrades. | L'algoritme és senzill però relativament pesat a nivell d'operacions ja que no es poden computar en paral·lel els processos degut a que cada reflexió s'ha de fer en l'ordre adequat. | És l'algoritme més delicat i laboriós d'implementar, ja que l'ordre d'aplicació de les files a calcular no és trivial si s'aplica el teorema en general. |

*Taula 1.c*

# 2. Programació de Householder out-of-core

La idea principal d'aquests codis out-of-core és d'utilitzar porcions estretes de la matriu G, que anomenem tires, per efectuar les lectures/escriptures de la memòria. Segons la mida de la matriu convé adaptar aquest paràmetre, ja que com més alta es fa la tira, més estreta s'ha de fer per que pugui seguir cabent a la memòria. Una simplificació que és general en aquest projecte, és que s'utilitzen tires iguals (és a dir, el número de tires és múltiple del número total de columnes). Sent així totes les tires iguals en mida. Per exemple, aquí s'ha separat la matriu en tires de dues columnes:




Cada tira es llegeix després de l'anterior, mai alhora. El sentit de lectura no és sempre l'utilitzat en l'exemple.

No s'han utilitzat els algoritmes d'altres estudis citats en la secció 1, ja que a cada lectura de dades de la matriu, agafen una certa mida de files i columnes, però per resoldre un algoritme com el de householder (que només necessita fer reflexions utilitzant dades d'una mateixa columna i dades de referència) és molt més eficient organitzar-ho en forma de tires. Per ara, expliquem amb un exemple l'algorisme de reflexions de householder normal.

**Exemple:**

$$\begin{pmatrix} 1 & 2 \\ 5 & 3 \\ 3 & 7 \end{pmatrix} \cdot M = B$$

Primer es canvia la base del vector total de la primera columna de manera que quedi només el terme superior, sent aquest la norma de l'antic vector.

Així doncs, si tenim un vector:

$$v = \begin{pmatrix} 1 \\ 5 \\ 2 \end{pmatrix}$$

Canviat de base queda:

$$v' = \begin{pmatrix} \|v\| \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \|\sqrt{30}\| \\ 0 \\ 0 \end{pmatrix}$$

Ara trobem el vector de la reflexió u:

$$u = \begin{pmatrix} \|v\| \\ 0 \\ 0 \end{pmatrix} - v$$

*ni el signe ni la norma importen ja que es simplifiquen en la formula de la reflexió.

Ara reflectim totes les altres columnes w i el terme independent B.

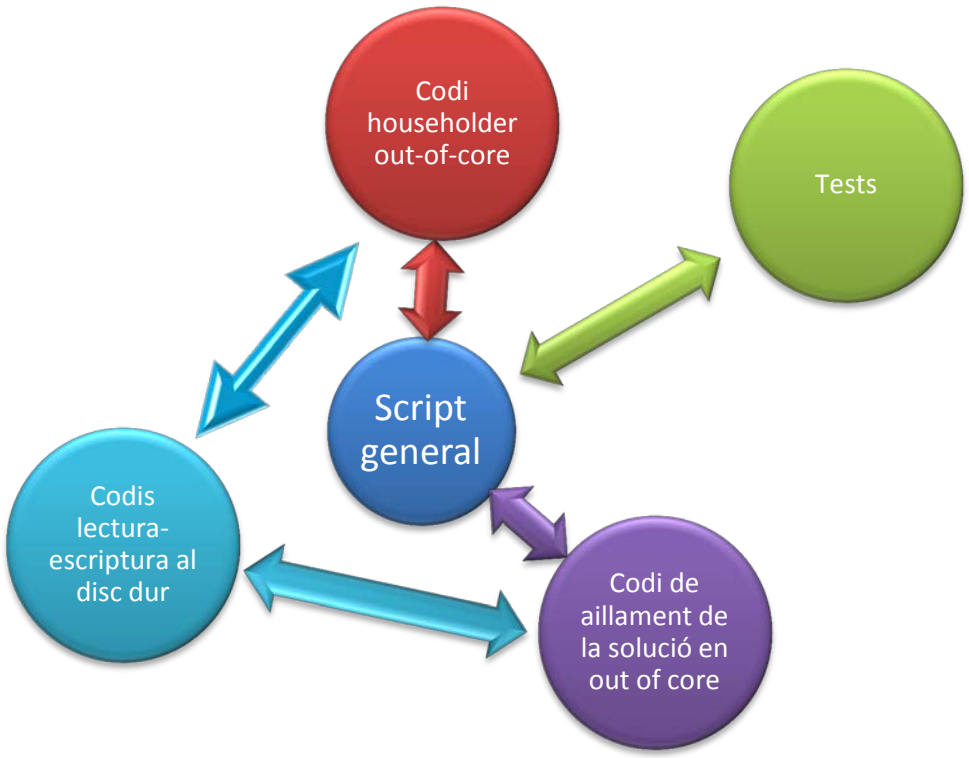
$$w' = Ref(w) = 2\frac{v \cdot u}{u \cdot u}u - v$$

A la següent iteració, baixem una fila per mantenir la triangularitat superior de la matriu, i repetim el procés.

L'algorisme lògic està explicat a part en la secció corresponent del codi, però bàsicament, al estar reflectint una tira, es crea una matriu de reflectors auxiliar, que conté tots els reflectors de totes les columnes d'aquesta tira (recordem que en aquest moment el programa només té a la seva disposició les dades de la tira en qüestió). Ara, en un bucle intern, s'apliquen cada reflexió d'aquesta matriu auxiliar, tira per tira, a totes les tires a la dreta de la matriu, en la bona seqüència i a la bona alçada. Bàsicament, es com si en l'exemple anterior, cada vector fos ell mateix una tira amb columnes.

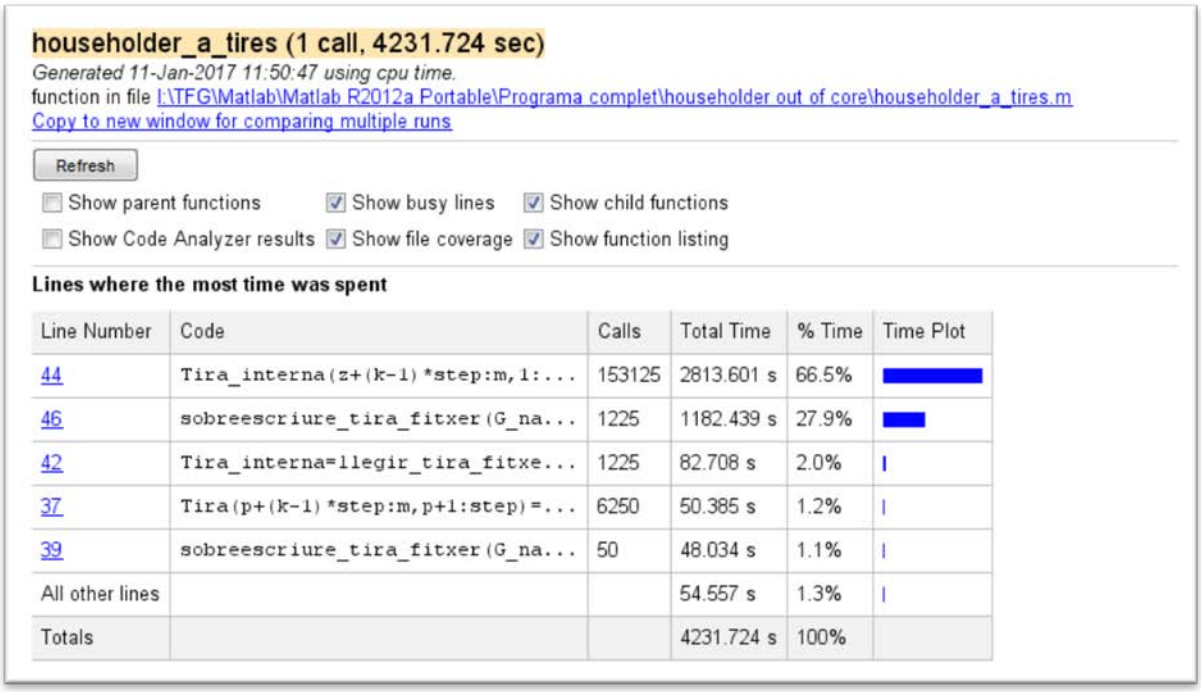
# 3. Codis

L'estructura del programa s'articula per un script general i tres conjunts de funcions que que crida el script. Una quart conjunt es pot utilitzar per accedir a les dades inicials i solucions dels tests als quals es sotmet el codi. El script general serveix per primer cridar el codi householder, que cridarà funcions del conjunt lectura/escriptura, i en un segon pas, el script general cridarà el codi d'aïllament de la solució que aïllarà la solució de manera out-of-core, i aquest codi també utilitzarà els codis de lectura/escriptura. Finalment es pot fer un test, que també pot cridar els elements out-of-core. També es pot crear o llegir la matriu sencera per matrius que quedin dins dels paràmetres in-core. La metodologia per fer aquest codi ha sigut sempre de reflexionar-ho per matrius molt petites i després anar provant si es pot augmentar la mida, verificant llavors amb l'ajuda de la reflexió de qr de matlab.



## 3.1. Codi Householder out-of-core

Householder_a_tires és una funció que aplica reflexions d'hiperplans de householder a la matriu de nom G_name guardada al disc dur. Retorna el terme independent B reflectit, ja que aquest últim sí que hi cap a la memòria, i la matriu reflectida és la mateixa matriu original, guardada amb nom G_name. Aquest codi s'ha elaborat a la FASE 1 del projecte, sense ser out-of-core, però plantejant ja la metodologia, i s'ha adaptat al final de la FASE 2, un cop ja fetes les funcions de lectura/escriptura de tires de la matriu guardada a la memòria. És important comentar que no s'han calcat els algorismes existents citats en la secció d'introducció, ja que en aquests casos s'agafaven parts de matrius. En el nostre cas, les tires optimitzen eficaçment el codi. A més, no podem fer anar cap for en paral·lel (parfor) que ens agilitzaria el codi, perquè el nostre problema és que no podem tenir més tires a la memòria. La durada de computació del codi ve marcada quasi totalment pels càlculs de les reflexions fetes a les tires a la dreta de la tira actual (bucle intern-intern). A la captura següent tenim un exemple de la tasca dominant el nostre codi householder, que és la línia 44: les reflexions de tires successives per cada tira a la dreta de la tira actual.



**householder_a_tires (1 call, 4231.724 sec)**

*Generated 11-Jan-2017 11:50:47 using cpu time.*

function in file I:\TFG\Matlab\Matlab R2012a Portable\Programa complet\householder out of core\householder_a_tires.m

Copy to new window for comparing multiple runs

Refresh

Show parent functions | Show busy lines | Show child functions

Show Code Analyzer results | Show file coverage | Show function listing

**Lines where the most time was spent**

| Line Number | Code | Calls | Total Time | % Time | Time Plot |
|---|---|---|---|---|---|
| 44 | `Tira_interna(z+(k-1)*step:m,1:...` | 153125 | 2813.601 s | 66.5% | |
| 46 | `sobreescriure_tira_fitxer(G_na...` | 1225 | 1182.439 s | 27.9% | |
| 42 | `Tira_interna=llegir_tira_fitxe...` | 1225 | 82.708 s | 2.0% | |
| 37 | `Tira(p+(k-1)*step:m,p+1:step)=...` | 6250 | 50.385 s | 1.2% | |
| 39 | `sobreescriure_tira_fitxer(G_na...` | 50 | 48.034 s | 1.1% | |
| All other lines | | | 54.557 s | 1.3% | |
| Totals | | | 4231.724 s | 100% | |

D'altra banda, separar sempre que es pugui el codi en columnes es 50% més eficient a nivell de càlcul que en horitzontal perquè MATLAB emmagatzema internament les matrius en columnes.

### 3.1.1. Inputs

G_name: és el nom de l'arxiu que conté la matriu a reflectir. Fitxer .dat o .bin.

B: és el terme independent de G*M=B.

m: nombre de files de la matriu G

n: nombre de columnes de la matriu G

Step: número de columnes que tindrà cada tira

### 3.1.2. Output

El vector terme independent B reflectit, normalment l'anomenem B_new.

### 3.1.3. Explicació de l'algorisme

L'estratègia que seguim, per estalviar lectures/escriptures, és guardar a la memòria la tira de matriu Q corresponent a la tira que estigui llegint el codi.

L'algoritme és el següent:

Bucle extern: recorre les tires (és a dir, for per k de 1 al número de tires)

> Llegeix del disc la tira k-èsima
>
> Bucle intern: recorre cada columna de la tira per aplicar la reflexió de householder a l'alçada que correspongui i guarda aquests vectors reflectors en una tira ús (de la mida d'una tira, i es sobreescriu a cada bucle extern).
>
> Tanca el bucle intern

Escriu en disc dur la tira k-èsima ja reflectida

Aplica els reflectors del bloc k-èsima al terme independent del sistema

Bucle intern-intern: recorre les tires l-èsima a la dreta de la tira k

> llegeix de disc dur la tira l-èsima
>
> Ara s'utilitza el conjunt de vectors reflectors ús de la tira per reflectir una a una les tires que estan a la dreta de la tira k. S'han d'aplicar en el bon ordre.

escriu en disc dur la tira l-èsima reflectida

fi dels bucles extern i interns

### 3.1.4. Codi householder_a_tires(G_name,B,m,n,step)

```
function [B]=householder_a_tires(G_name,B,m,n,step)
% Jaume Amoros i Wilhelm Auffermann, UPC, Barcelona
% 2016/12/29
h = waitbar(0,'Initializing waitbar...');
H = waitbar(0,'Initializing waitbar...');
ntires=n/step;%guarda el numero de tires que tenim
for k=1:ntires %Bucle extern que recorre les tires
    waitbar(k/ntires,h,'external loop...')
    Tira=llegir_tira_fitxer(G_name,m,k,step );
    w = zeros(m,step);
    us=zeros(m,step); %Prelocation de tots els vectors reflectors de la
tira(optimitzaciÛ)
    for p= 1:step %recorre cada columna per trobar i desar les
reflexions
        v=Tira(p+(k-1)*step:m,p); %Agafa a partir de la (k-1)*step
component de la columna
        %Posem el valor absolut de v a on cau
        w(p+(k-1)*step,p)=norm(v);%posem la norma a la diagonal
        %Escribim la nova columna reflectora
        Tira(p+(k-1)*step:m,p)=w(p+(k-1)*step:m,p);
        %Trovem el vector normal amb v i w de la reflexio de la columna
actual
        u=v-w(p+(k-1)*step:m,p); %calculem el vector reflector de la
columna p
        us(p+(k-1)*step:m,p)=u; %Guardem els vectors reflectors
        %s'aplica la reflexiÛ al terme independent
        B(p+(k-1)*step:m,1)= B(p+(k-1)*step:m,1)-(u.'*B(p+(k-
1)*step:m,1))*((2*u)/(u.'*u));
        %Apliquem la reflexiÛ a la tira.
        Tira(p+(k-1)*step:m,p+1:step)=Tira((p+(k-1)*step):m,(p+1):step)-
2/(u.'*u)*u*(u.'*Tira(p+(k-1)*step:m,(p+1):step));
    end
    sobreescriure_tira_fitxer(G_name,m,k,Tira,step)
    for l=k+1:ntires %Bucle intern que recorre les tires a la dreta de
la actual
        waitbar(l/(ntires-k),H,'inner loop...')
        Tira_interna=llegir_tira_fitxer(G_name,m,l,step );%llegeix la
tira
        for z=1:step %recorre els reflectors per a una tira l
            Tira_interna(z+(k-1)*step:m,1:step)=Tira_interna(z+(k-
1)*step:m,1:step)-2/(us(z+(k-1)*step:m,z).'*us(z+(k-
1)*step:m,z))*us(z+(k-1)*step:m,z)*(us(z+(k-
1)*step:m,z).'*Tira_interna(z+(k-1)*step:m,1:step));
        end
        sobreescriure_tira_fitxer(G_name,m,l,Tira_interna,step );
    end

end
end
```

## 3.2. Codis lectura-escriptura al disc dur

Householder_a_tires i trobar_incognites criden llegir_tira_fitxer i sobreescriure_tira_fitxer, que llegeixen o sobreescriuen una tira concreta de la matriu G_name, sense llegir ni les tires anteriors ni les tires posteriors. S'han utilitzat les funcions fread, fwrite i fseek, a les que se'ls ha d'especificar quina codificació de matlab tindran les dades. S'ha utilitzat la classe "double", en la que cada dada ocupa 8bytes, (64bits), ja que sempre s'utilitzaran dades decimals. Remarcar que la funció fseek indexa l'arxiu a llegir per bytes, i com que cada dada ocupa 8 bytes, s'ha hagut de multiplicar l'índex per 8, ja que sinó dóna números sense donar error però aquests números resulten d'una codificació errònia i no tenen sentit, ja que els llegeix per defecte com integrals de 1bit ( 8bytes). A més que la matriu és 8 vegades més llarga. Un error que es tenia és que es llegia aquesta matriu com a dades de 1 byte i això provocava que tingués un espai al disc de 1/8 del que hauria de tenir.

### 3.2.1. Codi llegir_tira_fitxer

#### 3.2.1.1. Inputs

L'input 1 de la funció és el nom del fitxer (string). per exemple: "G.bin"

L'input 2 són les m files de la matriu

L'input 3 és l'índex de la tira t. Queda acotat en el número total de tires

L'input 4 és quantes columnes té una tira (step)

#### 3.2.1.2. Output

Com a output retorna la tira com a m files i step columnes

#### 3.2.1.3. Algorisme

Primer s'obre l'arxiu.

Ara busquem el bon índex dins del nostre arxiu,

Ara es llegeixen tants bytes com faci falta, en una sola lectura, ja que les dades s'emmagatzemen seqüencialment de columna a columna de dalt a baix.[1]

Tanquem el fitxer

[1] *Així doncs, llegint d'una tirada un múltiple de llargada de columna, es poden llegir de cop més columnes. Després només cal organitzar-ho amb les bones mides, que seran de [m,step].*

#### 3.2.1.4. Codi

```
function [tira]=llegir_tira_fitxer(G_name,m,t,step )
    c=(t-1)*step; %l'index de la primera columna de la tira t
    fileIDllegir=fopen(G_name, 'r');
    fseek(fileIDllegir,m*(c)*8,'bof'); %Busca el index, per bytes
    tira=fread(fileIDllegir, [m,step],'double'); %Llegeix la tira
    fclose(fileIDllegir);
end
```

### 3.2.2. Codi sobreescriure_tira_fitxer

#### 3.2.2.1. Inputs

L'input 1 de la funció és el nom del fitxer (string). per exemple: "G.bin"

L'input 2 són les m files de la matriu

L'input 3 és l'índex de la columna p. p comprès entre 1:n

L'input 4 és la nova tira que volem reescriure en el fitxer

L'input 5 és la step de la tira (quantes columnes té cada tira)

#### 3.2.2.2. Algorisme

Obrir el fitxer amb drets de lectura/escriptura sense esborrar el contingut (r+).

Anar a l'índex de tira adequat

Escriure la nova tira[2]

Tancar l'arxiu

#### 3.2.2.3. Codi

```
function sobreescriure_tira_fitxer(G_name,m,p,nova_tira,step )
    fileID=fopen(G_name, 'r+');%Obrir el fitxer amb drets de
lectura/escriptura sense borrar el contingut (r+).
    c=(p-1)*step; %la primera columna de la tira p
    fseek(fileID,m*(c)*8,'bof');% m*(c)*8 ja que agafa el primer double
despres de saltarse
    fwrite(fileID, nova_tira,'double');
    fclose(fileID);
end
```

[2] *La tira s'escriu de dalt a baix de columna a columna, així doncs, les dades queden ordenades i es poden recuperar seguint una descodificació inversa.*

### 3.2.3. Codi llegir_matriu_sencera

Aquesta funció és utilitzada per llegir tota la matriu i posar-la a la memòria en cas de que hi càpiga a la memòria. S'utilitza per testejar.

#### 3.2.3.1. Inputs

L'input 1 de la funció és el nom del fitxer (string). per exemple: "G.bin"

L'input 2 són les m files de la matriu

L'input 3 són les n columnes de la matiru

L'input 4 és la step de la tira (quantes columnes té cada tira)

#### 3.2.3.2. Output

Retorna la matriu sencera [m,n].

#### 3.2.3.3. Algorisme

Obrir el fitxer amb drets de lectura

Llegir en bucle cada nova tira

Tancar el bucle

Tancar el fitxer

#### 3.2.3.4. Codi

```
function [novaG]=llegir_matriu_sencera(G_name,m,n,step)
novaG=zeros(m,n);%Allocation per estalviar memòria
fileID=fopen(G_name, 'r');
for a=1:step:n %reccorre les columnes per tires
    novaG(1:m,a:a+step-1)=fread(fileID, [m,step],'double');
end
fclose(fileID);
end
```

### 3.2.4. Codi creacio_fitxer_per_tires

Aquesta funció és utilitzada per crear una matriu massa gran per que hi càpiga a la memòria. Ho fa de manera iterativa, i s'utilitza per testejar el codi.

#### 3.2.4.1. Inputs

L'input 1 de la funció és el nom del fitxer (string). per exemple: "G.bin"

L'input 2 són les m files de la matriu

L'input 3 són les n columnes de la matiru

L'input 4 és la step de la tira (quantes columnes té cada tira)

#### 3.2.4.2. Algorisme

Obrir el fitxer amb drets d'escriptura

Escriure en bucle cada nova tira composta de números aleatoris

Tancar el bucle

Tancar el fitxer

#### 3.2.4.3. Codi

```
function creacio_fitxer_per_tires(G_name,m,n,step)
fileID = fopen(G_name,'w');
for p =1:step:n
    Tira=randi([-5,5],m,step);%creem numeros enters aleatoris
    fwrite(fileID,Tira,'double');%escribim la tira
end
fclose(fileID); %tanca el fitxer al acabar
end
```

## 3.3. Codi d'aïllament de la solució en out-of-core

Ara que hem pogut reflectir l'equació matricial, tal que:

$G \cdot M = B$

$<=> Q \cdot R \cdot M = B$

$<=> Q^{-1}Q \cdot R \cdot M = B$

$<=> Q^t \cdot Q \cdot R \cdot M = Q^t \cdot B$ (Els termes reflectits)

$<=> R \cdot M = Q^t \cdot B$ (Els termes reflectits)

Aquest codi s'encarrega d'aïllar la solució M de l'equació anterior, és a dir, fer:

$M = R^{-1} \cdot Q^t \cdot B$

#### 3.3.1.1. Inputs

L'input 1 de la funció és el nom del fitxer (string). per exemple: "G.bin"

L'input 2 és el terme independent ja reflectit.

L'input 3 són les m files de la matriu

L'input 4 són les n columnes de la matiru

L'input 5 és la step de la tira (quantes columnes té cada tira)

#### 3.3.1.2. Algorisme

Bucle extern que llegeix les tires del final fins al principi

Llegeix la tira actual

Bucle intern que llegeix les columnes del final fins al principi

Aplica la divisió adequada a cada columna de M, és a dir, "despeja" la darrera incògnita de la columna actual

Aplica també una resta al terme independent per actualitzar-lo segons la iteració actual

Tanca els bucles

#### 3.3.1.3. Codi

```
function [M,Br]=trobar_incognites(R_name,Br,m,n,step)
ntires=n/step; %Calculem el numero de tires (suposem divisio exacte)
M=zeros(n,1); %Utilitzem el metode de PRELOCATION per optimitzar el
codi.
for k=ntires:-1:1 %aixo vol dir que k va cap enrera
[tira]=llegir_tira_fitxer(R_name,m,k,step );
    for l=step:-1:1; %novament l va enrera, de la darrera columna a la
primera de la tira)
      M((k-1)*step+l)=Br((k-1)*step+l)/tira((k-1)*step+l,l);
      Br(1:(k-1)*step+l)=Br(1:(k-1)*step+l)-M((k-1)*step+l)*tira(1:(k-
1)*step+l,l);
    end
```

## 3.4. calculG_tires

És una funció feta per L'Albert Purtí i el Jaume Amorós. Aquesta funció crea una matriu com creació_matriu_sencera, però és una matriu que "mapeja" una proveta superconductora en nodes. Aquest codi s'ha utilitzat a la part d'aplicació d'aquest projecte. L'objectiu d'aquest codi concatenat amb el nostre és calcular la imantació escalar M a partir del camp magnètic vertical B mesurat a una certa alçada sobre la mostra amb una sonda Hall, tema més discutit a l'apartat Aplicació.

## 3.5. Resultats

### 3.5.1. Tests Acadèmics

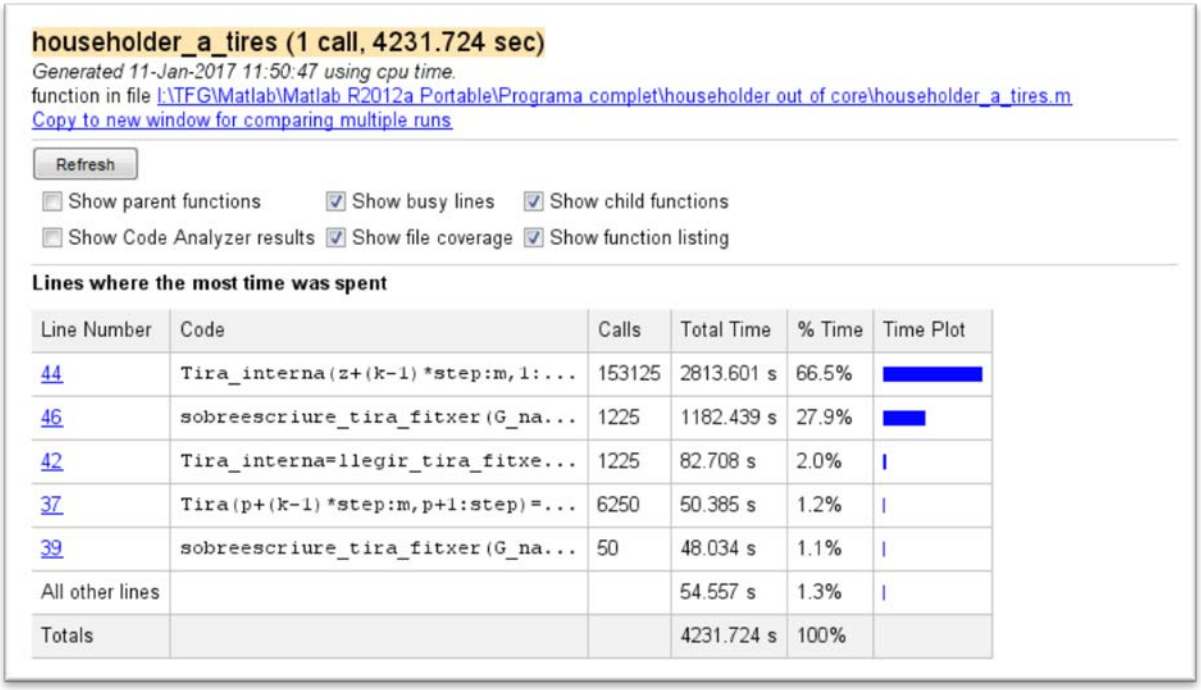
Els tests acadèmics consisteixen en crear una matriu G coneguda, i un terme independent també conegut, amb el que es pot (per dimensions no molt grans) conèixer la solució amb l'eina qr() de matlab, i anotar quan triga en efectuar el codi.

| Mida | Temps householder[3][4] | Temps aïllar solució |
|---|---|---|
| 500Mb | 1 hora 10minuts | 1 minut 11 segons |
| 4Gb | 8 hores 04minuts | 5 minuts 22 segons |
| 16Gb | 3 dies 7hores 43minuts | 19 minuts 13 segons |
| 32Gb | 10 dies 3hores 12 minuts | 39 minuts 43 segons |
| 64Gb | 21 dies 11 hores 15 minuts | 1 hora 12 minuts |

En l'extracte del profiler següent veurem per una matriu de mida no tan gran, com ja, el temps d'escriure en el disc dur no és molt significatiu, això que ens preocupava, ja que es volia que el codi gastés el seu temps de computació bàsicament en aplicar el Householder, i no en escriure al disc. Així doncs, podem veure que la nostra funció està optimitzada cap al fet de reflectir les tires a la dreta de la tira actual (és a dir, bucle intern-intern).

[3] La computació de householder és molt demandant a nivell de rendiment, i tot i que no dóna error (cosa que donaria amb l'eina qr(G) de Matlab) dilata aquesta potència de computació requerida en el temps.

[4] Tots aquests tests han sigut efectuats amb l'ordinador auxiliar



**householder_a_tires (1 call, 4231.724 sec)**

*Generated 11-Jan-2017 11:50:47 using cpu time.*

function in file I:\TFG\Matlab\Matlab R2012a Portable\Programa complet\householder out of core\householder_a_tires.m

Copy to new window for comparing multiple runs

Refresh

Show parent functions | Show busy lines | Show child functions

Show Code Analyzer results | Show file coverage | Show function listing

**Lines where the most time was spent**

| Line Number | Code | Calls | Total Time | % Time | Time Plot |
|---|---|---|---|---|---|
| 44 | Tira_interna(z+(k-1)*step:m,1:... | 153125 | 2813.601 s | 66.5% | |
| 46 | sobreescriure_tira_fitxer(G_na... | 1225 | 1182.439 s | 27.9% | |
| 42 | Tira_interna=llegir_tira_fitxe... | 1225 | 82.708 s | 2.0% | |
| 37 | Tira(p+(k-1)*step:m,p+1:step)=... | 6250 | 50.385 s | 1.2% | |
| 39 | sobreescriure_tira_fitxer(G_na... | 50 | 48.034 s | 1.1% | |
| All other lines | | | 54.557 s | 1.3% | |
| Totals | | | 4231.724 s | 100% | |

```
0.03      1   18 H = waitbar(0,'Initializing waitbar...');
          1   19 ntires=n/step;%guarda el numero de tires que tenim
          1   20 for k=1:ntires %Bucle extern que recorre les tires
0.05      8   21     waitbar(k/ntires,h,'external loop...')
0.26      8   22     Tira=llegir_tira_fitxer(G_name,m,k,step );
0.03      8   23     w = zeros(m,step); %Prelocation de tots els vectors reflectors de la tira(optimització)
0.03      8   24     us=zeros(m,step);
          8   25     for p= 1:step %recorre cada columna per trobar i desar les reflexions
0.04   1600   26         v=Tira(p+(k-1)*step:m,p); %Agafa a partir de la (k-1)*step component de la columna
              27         %Posem el valor absolut de v a on cau
0.19   1600   28         w(p+(k-1)*step,p)=norm(v);%posem la norma a la diagonal
              29         %Escribim la nova columna reflectora
0.04   1600   30         Tira(p+(k-1)*step:m,p)=w(p+(k-1)*step:m,p);
              31         %Trovem el vector normal amb v i w de la reflexio de la columna actual        clear v
0.04   1600   32         u=v-w(p+(k-1)*step:m,p);
0.02   1600   33         us(p+(k-1)*step:m,p)=u;
              34         %s'aplica la reflexió al terme independent
0.21   1600   35         B(p+(k-1)*step:m,1)= B(p+(k-1)*step:m,1)-(u.'*B(p+(k-1)*step:m,1))*((2*u)/(u.'*u));
              36         %Apliquem la reflexió a la tira.
8.85   1600   37         Tira(p+(k-1)*step:m,p+1:step)=Tira((p+(k-1)*step):m,(p+1):step)-2/(u.'*u)*u*(u.'*Tira(p+(k-1)*step:m,(p+1):step));
< 0.01 1600   38     end
2.98      8   39     sobreescriure_tira_fitxer(G_name,m,k,Tira,step)
< 0.01    8   40     for l=k+1:ntires %Bucle intern que recorre les tires a la dreta de la actual
0.33     28   41         waitbar(1/(ntires-k),H,'inner loop...')
1.83     28   42         Tira_interna=llegir_tira_fitxer(G_name,m,l,step );%llegeix la tira
         28   43         for z=1:step %recorre els reflectors per a una tira l
69.39  5600   44             Tira_interna(z+(k-1)*step:m,1:step)=Tira_interna(z+(k-1)*step:m,1:step)-2/(us(z+(k-1)*step:m,z).'*us(z+(k-1)*step:m,z))*us(z+(k-1)*step:
0.02   5600   45         end
11.92    28   46         sobreescriure_tira_fitxer(G_name,m,l,Tira_interna,step );
         28   47     end
              48
< 0.01    8   49 end
          1   50 end
```

A la segona captura podem evidenciar que on més temps passa computant és a nivell de les reflexions successives de les tires.

# 4. Proves Reals

## 4.1. Aplicació al Càlcul De Corrents Crítics En Materials Superconductors

### 4.1.1. Presentació dels materials superconductors

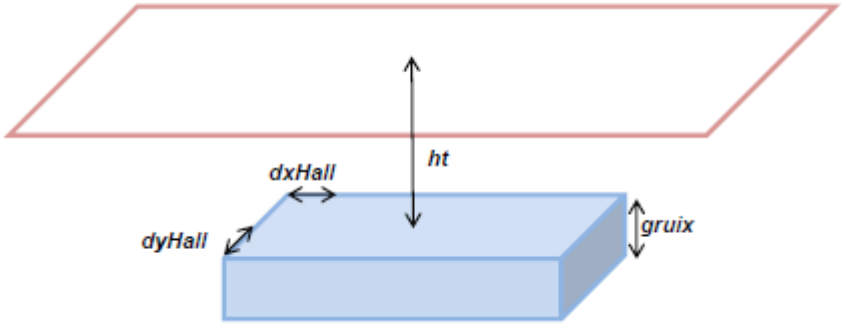
En aquesta aplicació, es tracta amb el material superconductor ceràmic YBaCuO. Té la particularitat de que a temperatures baixes no oposa resistència al flux d'electricitat que hi circula. L'interès de fabricar una peça de mida operativa (és a dir, de mida centimètrica) és evident per les moltes aplicacions que pot tenir a nivell tecnològic. Per això, en aquesta aplicació, es busca la distribució del corrent dins del superconductor, per poder fer controls de qualitat.

### 4.1.2. Treball Previ a aquest projecte

Aquesta aplicació es basa en un treball fet pel Jaume Amorós, el Miquel Carrera, Xavier Granados i Abel Purtí ([Carrera et al, 2003], [Amorós et al, 2012] [Purti, 2016]), que tracta de calcular el corrent crític en materials superconductors gràcies a una sonda Hall que mesura el camp magnètic a una certa alçada h de la mostra, mentre és recorreguda per un cert corrent.

La llei de Biot-Savart és una igualtat que permet trobar el camp magnètic present en un prisma de base rectangular, gràcies al corrent elèctric que passa per ell. El que s'ha volgut fer en aquest cas, ha sigut l'invers, i.e., trobar la distribució del corrent elèctric dins de la mostra, sabent els valors del camp magnètic exteriors.

Aquest codi d'Amorós, Purtí i altres permet doncs calibrar els valors de la sonda Hall que mesura el camp, ajustant la distancia entre mesures, l'alçada en la qual es fan les mesures, i el gruix de la peça en l'eix z).




[5]

Després, es retalla la finestra de mesures de Bz, ja que el pla superior es sobredimensiona a propòsit. D'aquesta malla superior només s'agafa un interval. A nivell pràctic, aquest interval li capem resolució fins a una mida de m files i n columnes. Aquests paràmetres de calibratge són: pasenfila, que ajusta m i pasencolumna, que ajusta n. Com més grans són aquestes variables d'ajust, menys resolució hi haurà.

D'altre banda, la mostra es subdivideix en una malla rectangular de mi*ni elements.

Seguidament, el seu codi genera el total d'elements de la matriu G, efectuant un seguit d'integrals triples, el que resulta en una mida de m*mi files i n*ni columnes.

Quan ja es té el sistema GM=B, on G és la matriu resultant del càlcul, B és la component vertical del camp magnètic i M és la imantació, s'ha de resoldre utilitzant la resolució per Householder elaborada en aquest projecte, ja que en molts casos la matriu G supera la memòria RAM disponible.

S'executa doncs el codi:

```
%apliquem householder a G  i a B
[B_reflectit]=householder_a_tires(fitxerG,B,m*n,mi*ni,mi);
% Resol el sistema triangular de equacions en out of core
[M,Residu]=trobar_incognites(G_name,B_reflectit,m*n,mi*ni,mi);
```

Un cop resolt el sistema, es pot trobar fàcilment el corrent que passa per la mostra fent:

$$J = rot(M)$$

Seguidament, el seu codi emmagatzema els resultats i genera gràfiques de:

**El camp magnètic vertical B**

**El camp imantació M**

**La densitat de corrent i corbes de nivell del camp elèctric vertical Jv**

[5] *Propietat de l'Abel Purti*

**El camp vectorial del camp elèctric del pla x-y**

## 4.2. Resolució 1 de la mostra 1

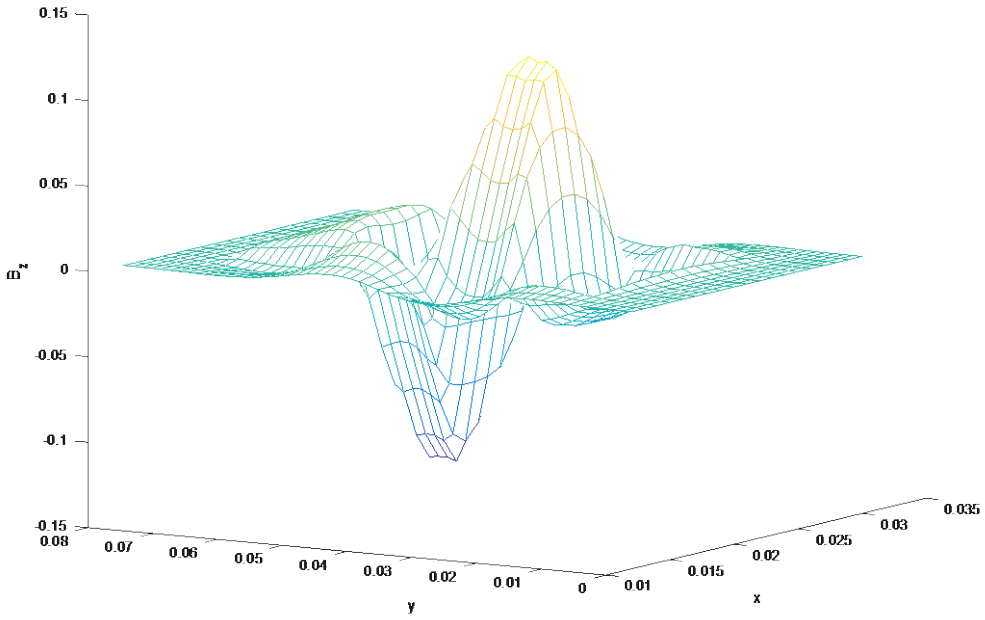
**Codi de la mostra:** **stackSPCuL60_2polsnou2_2016_3_30_12_45_**

Els paràmetres de la matriu G d'aquest primer anàlisi de la mostra són de poca precisió, tot i així, el resultat és satisfactori i les dades. El temps d'execució es molt petit.

| Mida | m | n | mi | ni | pasenfila | pasencolumna |
|---|---|---|---|---|---|---|
| 5.71Mb | 19 | 57 | 15 | 44 | 200 | 7 |

La matriu G té doncs 1.083 files i 660 columnes.

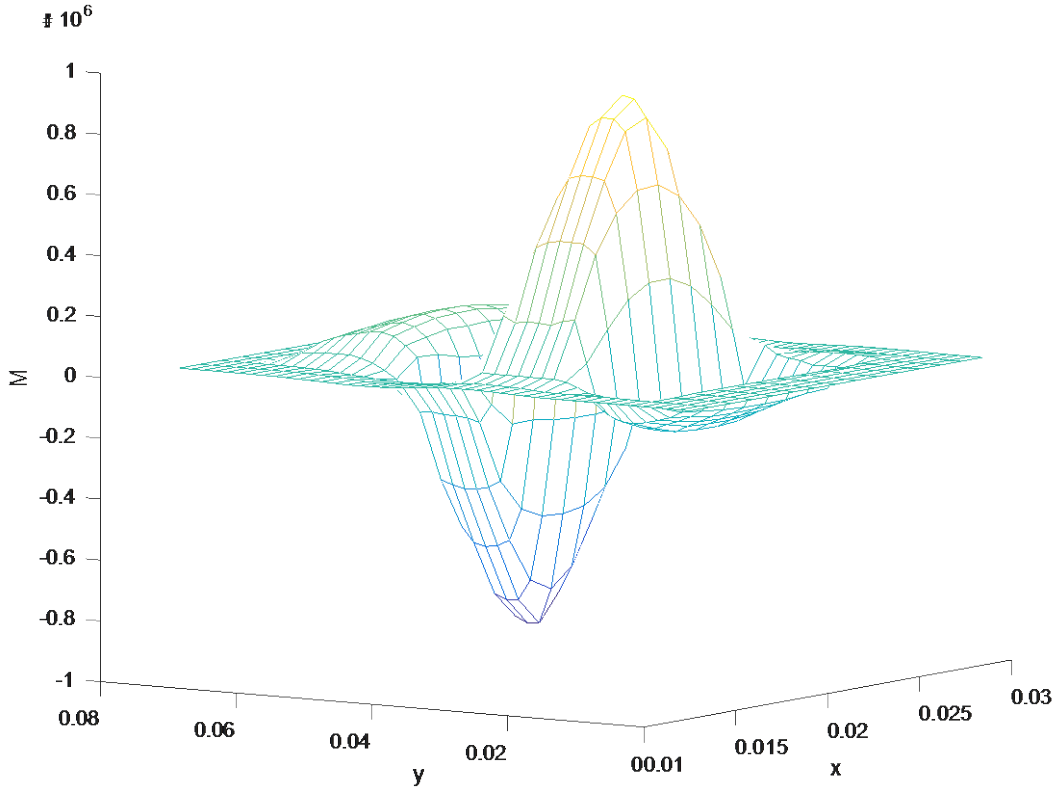
**El camp magnètic vertical Bz**




*Gràfic 1*

Al gràfic 1 podem veure el resultat del tractament de les mesures de la sonda Hall, valor màxim 0.011T. Observem que el resultat és l'esperat. (forma de dos pics de signe diferent). Així doncs, confirmem que el mallatge que hem agafat és correcte (tot i que molt bàsic).

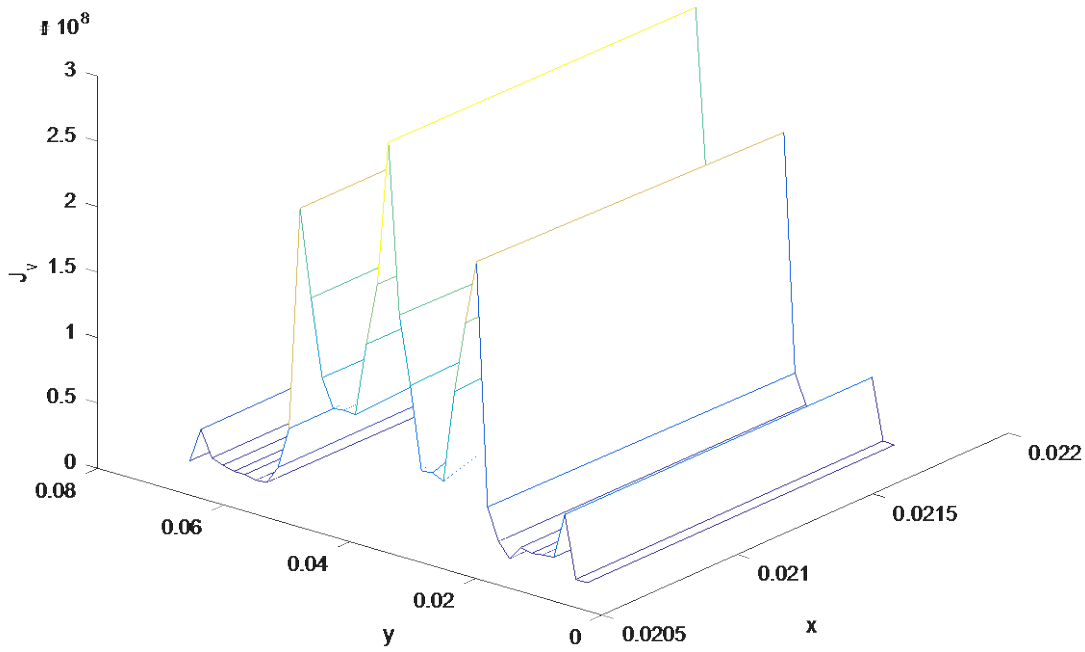
**El camp imantació M**




*Gràfic 2*

El camp imantació s'assembla molt al camp magnètic, i és que tenim G*M=Bz, així que són dependents amb certa proporció. El seu valor màxim es 8.8E5. Així doncs, al ser aquest el primer resultat del mètode out-of-core del projecte, podem afirmar que els resultats són molt bons, ja que ha donat el resultat esperat.

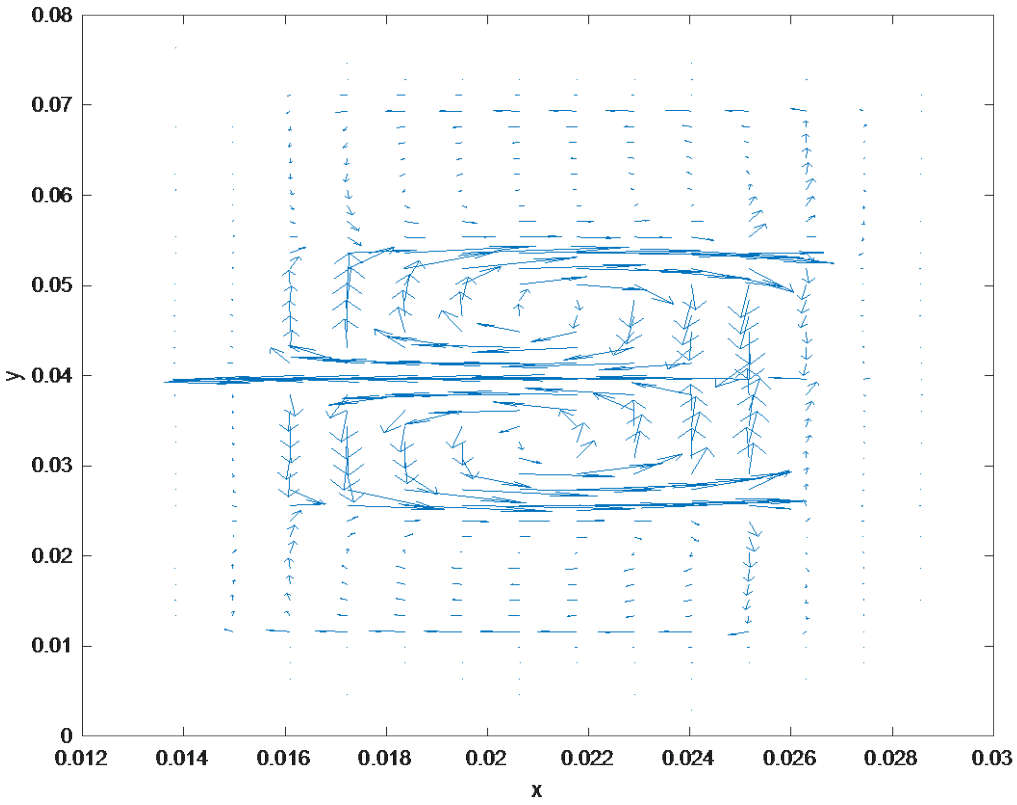
**La densitat de corrent i corbes de nivell del camp elèctric vertical Jv**




*Gràfic 3*

Degut a la poca precisió, no acabem de tenir la forma característica completa, tot i que sí que queda entre els paràmetres que es poden esperar.

**El camp vectorial del camp elèctric dels corrents Jx i Jy (en el pla de la mostra)**




*Gràfic 4*

Observem que la forma és satisfactòria, ja que veiem dos sentits (un horari i un altre anti-horari), que generen els camps magnètics en sentits contraris (positiu i negatiu) vist anteriorment.

## 4.3. Resolució 2 de la mostra 1

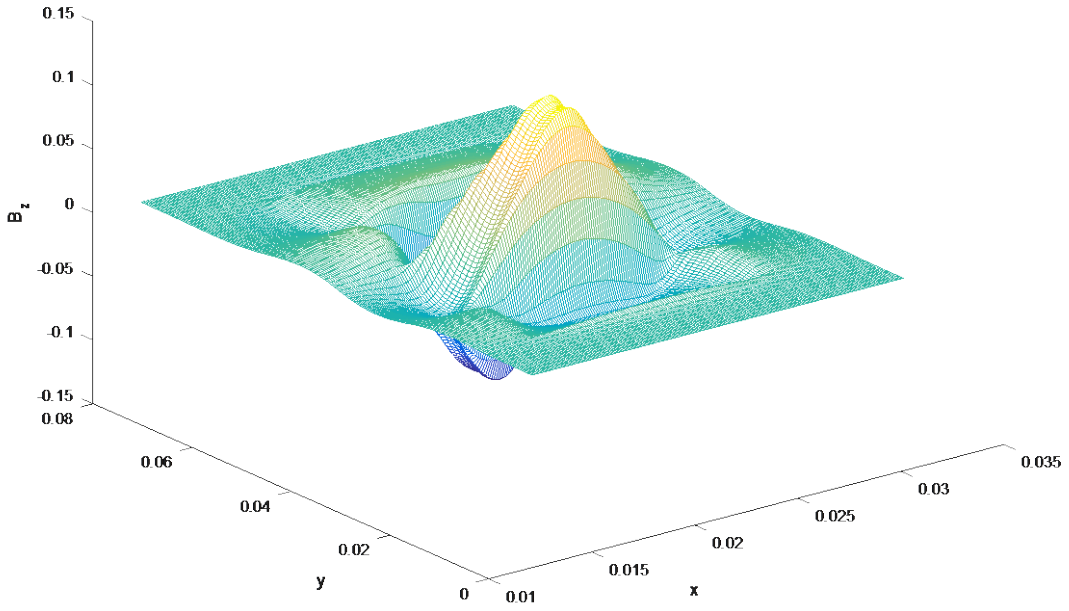
**Codi de la mostra:** **stackSPCuL60_2polsnou2_2016_3_30_12_45_**

Els paràmetres de la matriu G d'aquest segon anàlisi són de més precisió.

| Temps | Mida | m | n | mi | ni | pasenfila | pasencolumna |
|---|---|---|---|---|---|---|---|
| 18 minuts | 155,8Mb | 181 | 80 | 32 | 42 | 20 | 5 |

La matriu G té doncs 14.480 files i 1344 columnes.

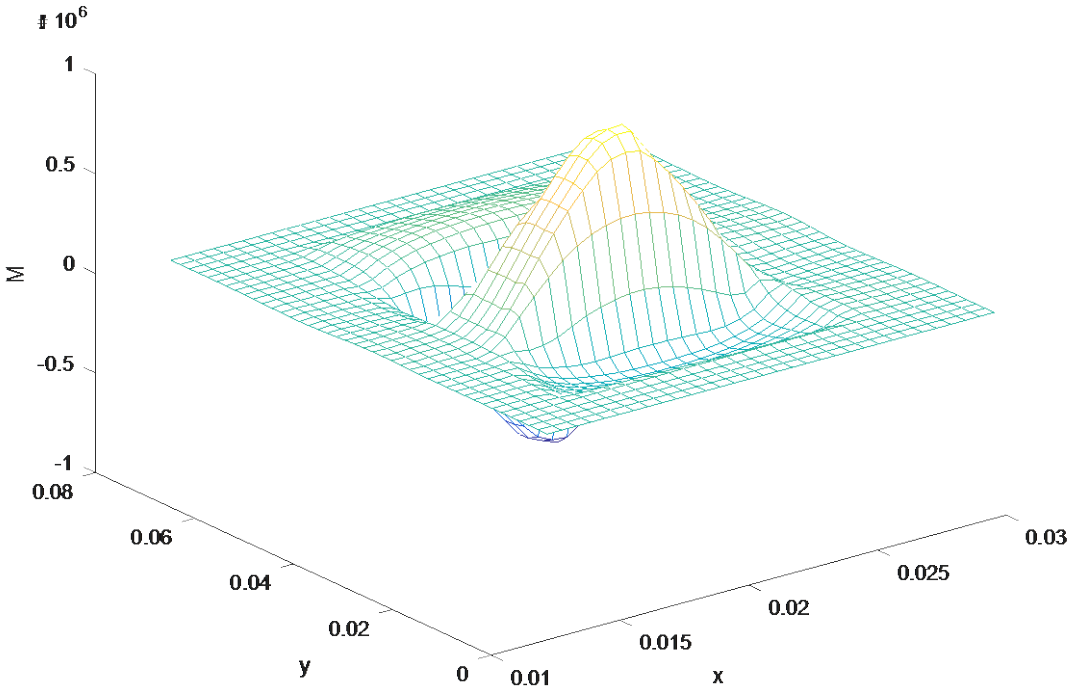
**El camp magnètic vertical Bz**




*Gràfic 5*

El gràfic ja té molta més definició.

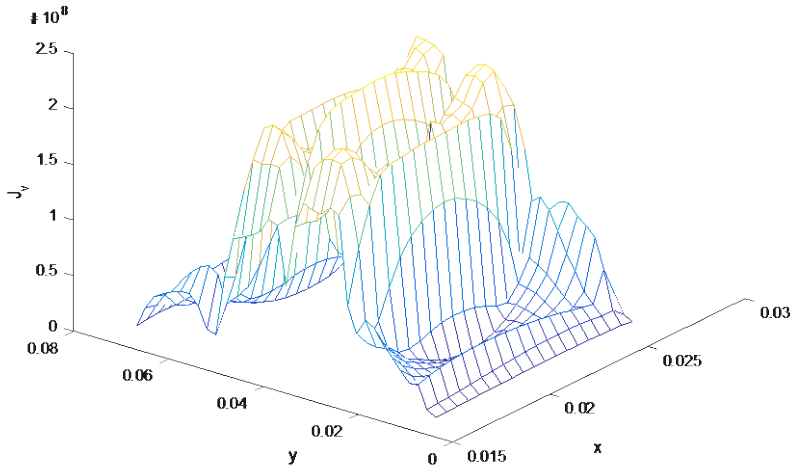
**El camp imantació M**



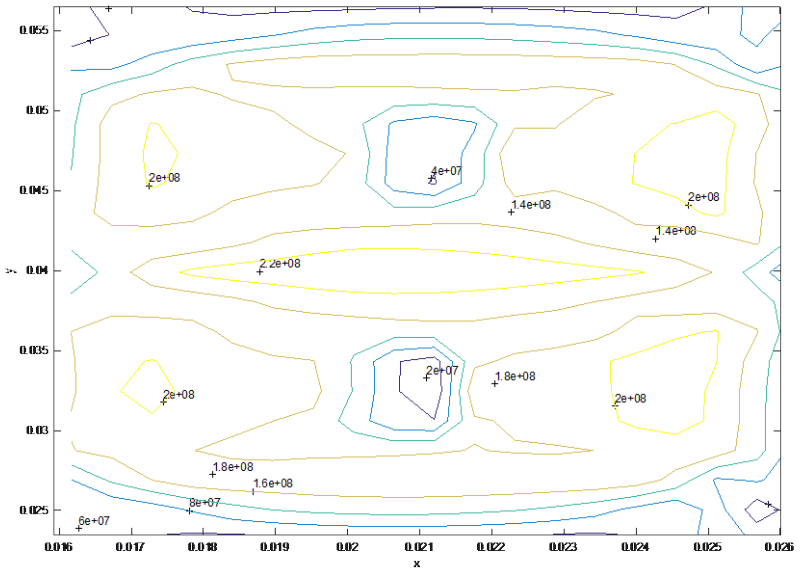

*Gràfic 6*

Com era d'esperar, s'ha refinat també el plot de M. S'ha comparat el plot de M obtingut amb el nostre codi Householder amb el plot de M seguint l'aillament in-core de matalb, i s'ha obtingut un error absolut molt satisfactori, de 3,1*10^-14.

**La densitat de corrent i corbes de nivell del camp elèctric vertical Jv**




*Gràfic 7*

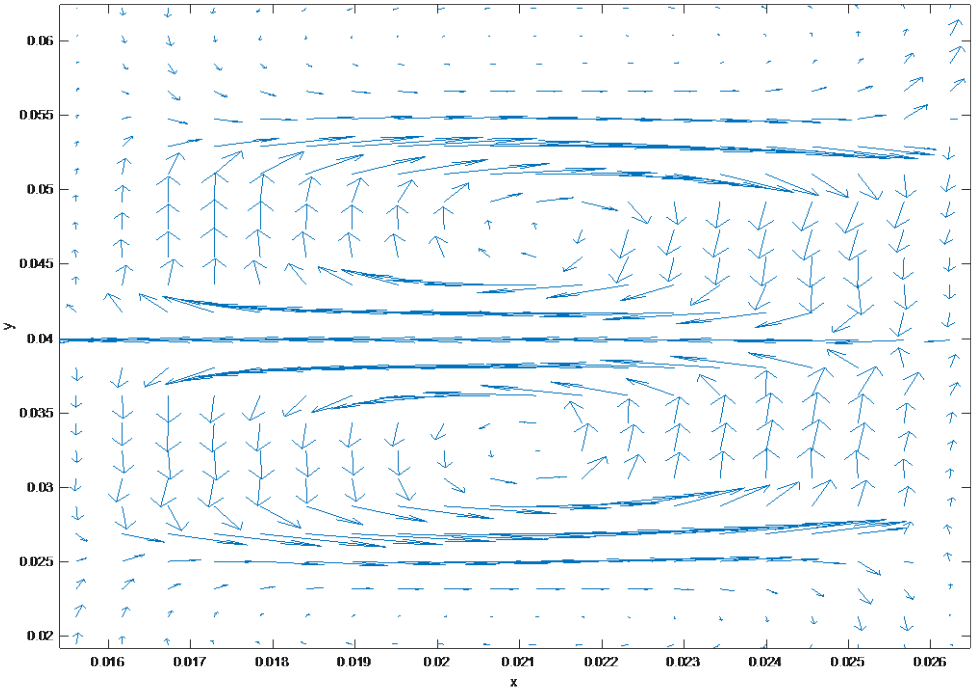



Gràfic 8

En aquest cas apreciem una diferència molt important, comparat amb el primer assaig poc precís. La distribució de Jv en x i y ja adquireix la forma que ha de tenir. (Els màxims corrents conformen dos rectangles i els mínims es troben en el seu interior).

ETSEIB

**El camp vectorial del camp elèctric dels corrents Jx i Jy (en el pla de la mostra)**




*Gràfic 9*

Observem que la forma és totalment satisfactòria, ja que veiem dos sentits (un horari i un altre anti-horari), que generen els camps magnètics en sentits contraris (positiu i negatiu) vist anteriorment.

## 4.4. Resolució 3 de la mostra 1

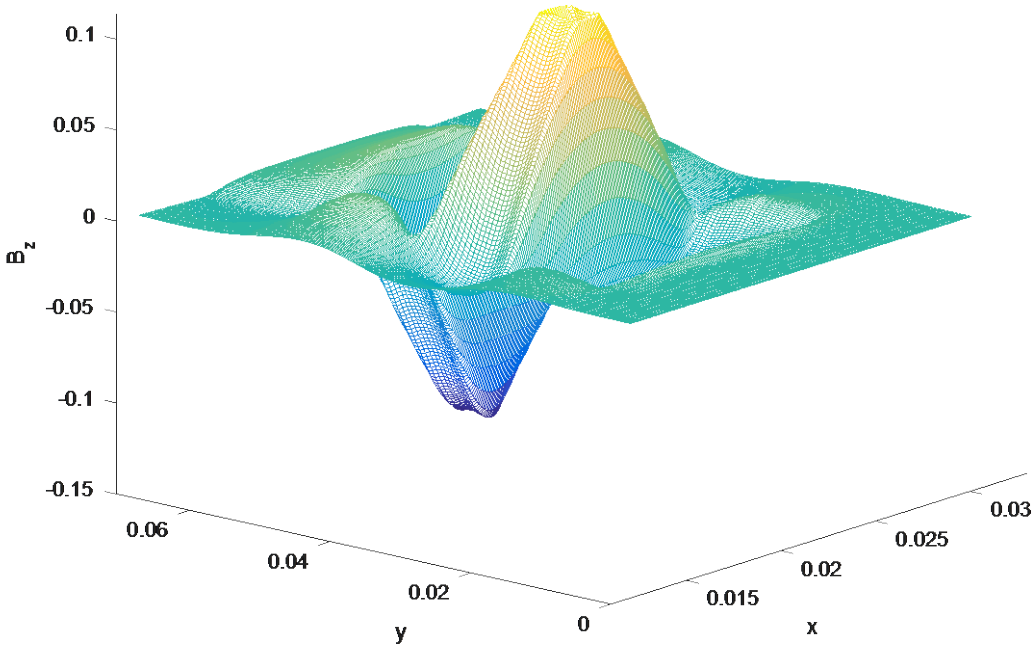
**Codi de la mostra:** **stackSPCuL60_2polsnou2_2016_3_30_12_45_**

Els paràmetres de la matriu G d'aquest tercer anàlisi son de molta més precisió, i es un exemple out-of-core.

| Temps[6] | Mida | m | n | mi | ni | pasenfila | pasencolumna |
|---|---|---|---|---|---|---|---|
| 1 dia 11 hores | 11.42Gb | 451 | 197 | 120 | 134 | 8 | 2 |

La matriu G té doncs 88.847 files i 16.080 columnes.

**El camp magnètic vertical Bz**

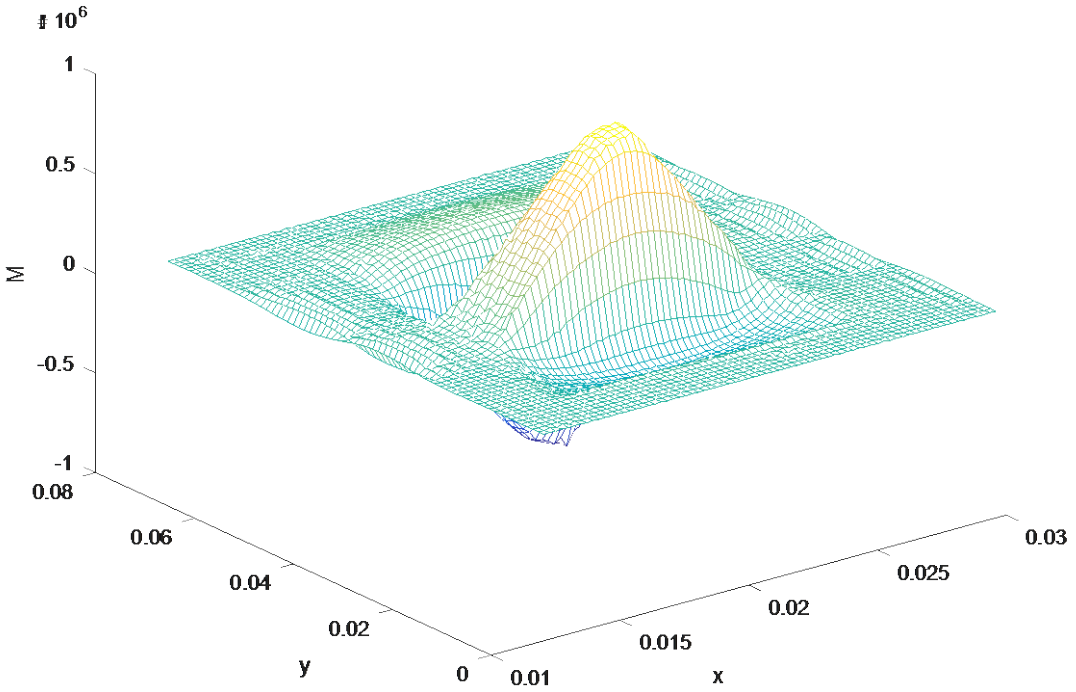



*Gràfic 10*

Al gràfic 10 podem veure com en aquest tercer mallatge la definició és més precisa

[6] Temps en fer la reflexió per householder i l'aillament de la solució.

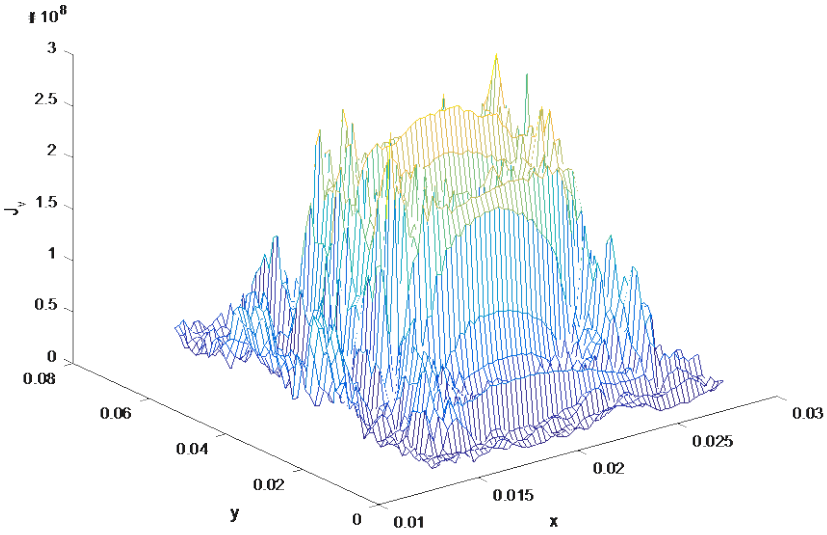
**El camp imantació M**



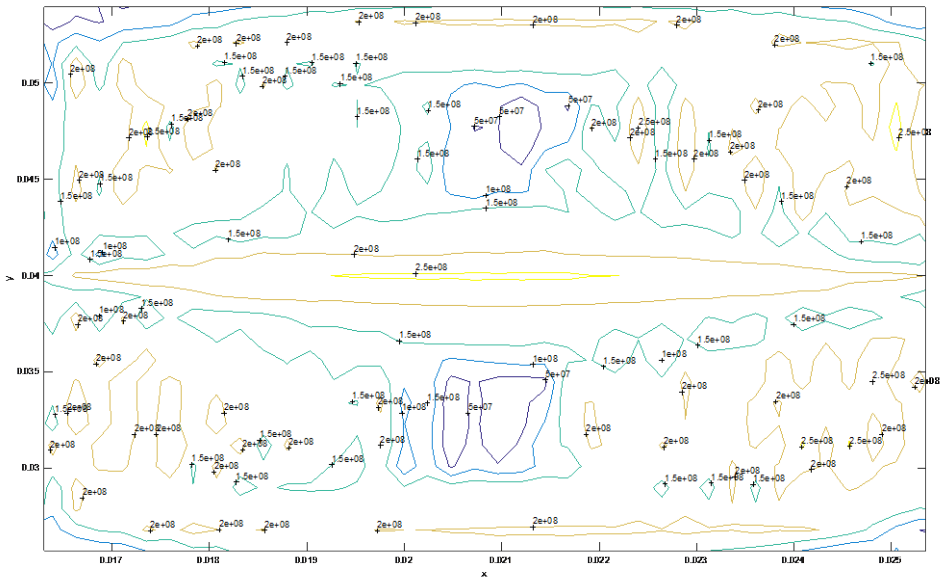

*Gràfic 11*

Així doncs, al ser aquest el primer resultat del mètode out-of-core del projecte, podem afirmar que els resultats son satisfactoris.

**La densitat de corrent i corbes de nivell del camp elèctric vertical Jv**



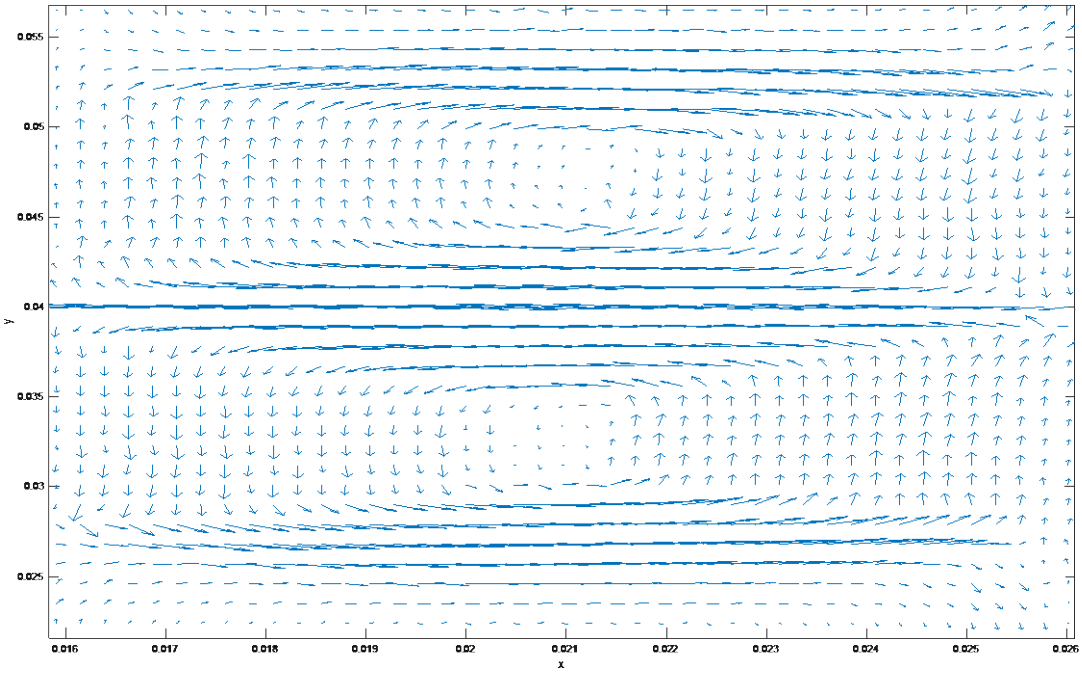

*Gràfic 12*




*Gràfic 13*

Observem en els gràfics 11 i 12 uns resultats més precisos que anteriorment i igualment corresponents amb el que haurien de donar.

**El camp vectorial del camp elèctric dels corrents Jx i Jy (en el pla de la mostra)**




*Gràfic 14*

Observem en el gràfic 14 que la forma es igualment satisfactòria, ja que seguim veient dos sentits (un horari i un altre anti-horari), que generen els camp magnètic en sentits contraris (positiu i negatiu) vist també en els dos exemples anteriors.

# Conclusions

Podem concloure que hem pogut franquejar satisfactòriament la limitació “out of memory” inherent a treballar amb matrius més grans que la memòria RAM de la màquina. A més, aquest programa té un potencial d’utilització molt gran, ja que es pot utilitzar per a qualsevol usuari que vulgui fer una descomposició QR per hiperplans de householder i aïllament de la solució del seu sistema, convertint-ho en una opció real low-cost aplicable a qualsevol laboratori modest -escola, universitat, empresa- que no tingui accés a un superordinador però sí que disposi de temps d’execució.

# Bibliografia

**J. Amorós, M. Carrera, X. Granados.**

*An effective model for fast computation of current distribution in operating HTS tapes from magnetic field measurements in non-destructive testing.*

Supercond. Sci. Technol. 25 104005.

2012

**D'Azevedo, Dongarra**

*The design and implementation of the parallel out-of-core Scalapack LU, QR and Cholesky*

*Factorization*

2000

**M. Carrera, J. Amorós, X. Obradors, J. Fontcuberta.**

*A new method of computation of current distribution maps in bulk high-temperature superconductors: analysis and validation*

Superconductor Science and Technology 16 1187--1194.

2003

**Eran Rabani and Sivan Toledo**

*Out-of-Core SVD and QR Decompositions*

2001

**G.H. Golub, C.F. Van Loan.**

*Matrix Computations. 3rd Edition.*

John Hopkins Press,

1996.

**Mercedes Marqués, Gregorio Quintana-Ortí Enrique S. Robert van de Geijn**

*Using Graphics Processors to Accelerate the Solution of Out-of-Core Linear Systems*

2009

**Purtí del Ruste, Abel**

*Càlcul del corrent crític en materials superconductors*

Treball de Final de Grau, grau en Tecnologies Industrials

ETSEIB, UPC. 2016

**Toledo, Sivan.**

*A Survey of Out-of-Core Algorithms in Numerical Linear Algebra*

Palo Alto Research Center, California,

1999

**Vitter, Jeffrey Scott –**

*Algorithms and Data Structures for External Memory*

2008

## *Bibliografia web*

*https://en.wikipedia.org/wiki/Out-of-core_algorithm*

*https://en.wikipedia.org/wiki/QR_decomposition#Using_the_Gram.E2.80.93Schmidt_process*

*https://en.wikipedia.org/wiki/QR_decomposition#Using_Householder_reflections*

*https://en.wikipedia.org/wiki/QR_decomposition#Using_Givens_rotations*

*https://es.mathworks.com/company/newsletters/articles/programming-patterns-maximizing-code-performance-by-optimizing-memory-access.html*

*http://www.cs.utexas.edu/~flame/web/methodology.html*