Επιστημονικός Υπολογισμός
Αναφορά Εργασίας 2021-22 (1ο μέρος)

Αβραμόπουλος Μιχαήλ 1067451 up1067451@upatras.gr

Start date: 14 Δεκεμβρίου 2021
End date: 14 Ιανουαρίου 2022

Περιεχόμενα

Στοιχεία Πειράματος

Έναρξη/λήξη εργασίας	14/12/21 -
Model	Προσωπικός Σταθερός Υπολογιστής
O/S	Windows 10 Education N 21H2
Processor name	AMD Ryzen 5 2600
Processor Speed	3.9 GHz (overclocked)
Number of Processors	1
Total # Cores	6
Total # Theads	12
L1 cache	6x64 KB (384 KB) Instruction & 6x32 KB (192 KB) Data write-back
L2 cache	6x512 KB (3 MB) write-back
L3 cache	2x8 MB (16 MB)
Gflops/s	267.1
Memory	16GB 3200MHz
Memory Bandwidth	43.71 GB/s
MATLAB Version	9.11.0.1809720 (R2021b) Update 1
BLAS	Intel(R) Math Kernel Library Version 2019.0.3 Product Build 20190125 for Intel(R) 64 architecture applications, CNR branch auto
LAPACK	Intel(R) Math Kernel Library Version 2019.0.3 Product Build 20190125 for Intel(R) 64 architecture applications, CNR branch auto, supporting Linear Algebra PACKage (LAPACK 3.7.0)

Table 1: Πίνακας Στοιχείων για τα πειράματα

Τις πληροφορίες για τον επεξεργαστή βρέθηκαν μεσώ προγράμματος CPU-Z, τα Gflops βρίσκονται εδώ και το Memory Bandwidth εδώ.Τέλος τα BLAS & LAPACK βρέθηκαν τρέχοντας τις εντολές στην Matlab version -blas & version -lapack αντίστοιχα.

Computer Type	LU	FFT	ODE	Sparse	2-D	3-D
Windows 10, AMD Ryzen Threadripper(TM) 3970x @ 3.50 GHz	0.2008	0.1881	0.3469	0.4396	0.2029	0.1117
Debian 10(R), AMD Ryzen Threadripper 2950x @ 3.50 GHz	0.3122	0.2377	0.3219	0.5047	0.5941	0.1631
iMac, macOS 11.2.3, Intel Core i9 @3.6 GHz	0.3278	0.2648	0.2674	0.2763	0.6898	0.3946
Windows 10, Intel Xeon(R) W-2133 @ 3.60 GHz	0.4154	0.2991	0.4348	0.4574	0.3167	0.2184
Windows 10, Intel Xeon CPU E5-1650 v3 @ 3.50 GHz	0.4614	0.3030	0.4455	0.4433	0.3559	0.2623
Windows 10, AMD Ryzen(TM) 7 1700 @ 3.00 GHz	0.7507	0.5163	0.4884	0.5441	0.3397	0.1849
This machine	1.1773	0.3869	0.5081	0.5019	0.3982	0.3399
Windows 10, Intel Core i7-10610 @ 1.8 GHz	0.9218	0.4394	0.3666	0.3844	0.7386	0.6251
Surface Pro 3,Windows(R) 10, Intel(R) Core(TM) i7-5600U @ 2.6 GHz	1.7475	0.9090	0.6178	0.5711	0.5713	0.3623
MacBook Pro, macOS 10.15.2, Intel Core i5 @ 2.6 GHz	1.6237	0.9786	0.5446	0.6173	2.5214	2.0229

Ερώτημα 1

```
function [val,row_ip,col_ip] = sp_mat2latex(A, sp_type)
if strcmp(sp_type, 'csr')
A = transpose(A);
end
[M,n,val] = find(A);
sizeA = size(A);
N = zeros(sizeA(2)+1,1);
N(1) = 1;
for  k = 2:sizeA(2)+1
N(k) = N(k-1) + size(n(n==k-1),1);
end
v = size(val);
edit 'erot1.tex';
f = fopen('erot1.tex', 'w');
fprintf(f, '\\begin{center}\n');
fprintf(f, 'val = \\begin{tabular}{|');
for k = 1:v(1)
fprintf(f, 'l|');
end
fprintf(f, '} \\hline\n');
fprintf(f, '%0.4f', val(1,1));
for t = 2:v(1)
fprintf(f, ' & %0.4f', val(t,1));
end
fprintf(f, '\\\\ \\hline\n\\end{tabular}\n\\\\n\\vspace{\\baselineskip}\n');
v = size(M);
fprintf(f, 'IA = \\begin{tabular}{|');
for k = 1:v(1)
fprintf(f, 'l|');
end
fprintf(f, '} \\hline\n');
fprintf(f, '%i', M(1,1));
for t = 2:v(1)
fprintf(f, ' & %i', M(t,1));
end
fprintf(f, '\\\\ \\hline\n\\end{tabular}\n\\\\n\\vspace{\\baselineskip}\n');
v = size(N);
fprintf(f, 'JA = \\begin{tabular}{|');
for k = 1:v(1)
fprintf(f, 'l|');
end
fprintf(f, '} \\hline\n');
fprintf(f, '%i', N(1,1));
for t = 2:v(1)
```

```
fprintf(f, ' & %i', N(t,1));
end
fprintf(f, '\\\\ \\hline\n\\end{tabular}\n\\\\\n\\vspace{\\baselineskip}\n');
fprintf(f, '\\end{center}\n');
fclose(f);
if strcmp(sp_type, 'csr')
col_ip = M;
row_ip = N;
else
col_ip = N;
row_ip = M;
end
```

Για να μπορέσουμε να βρούμε τους πίνακες **val, IA, JA** οφείλουμε να δούμε αν θα κάνουμε αναπαράσταση σε **CSR** ή σε **CSC** για να κάνουμε **transpose** το μητρώο A. Χρησιμοποιώντας την συνάρτηση find της Matlab κάνουμε αναζήτηση στις γραμμές κάθε στήλης για μη-μηδενικά στοιχεία και μας επιστρέφει τις θέσεις τους και τις τιμές τους. Οπότε αποκτούμε έτσι το **val** και **IA** για την αναπαράσταση **CSC**, το μόνο που μένει μετά είναι να φτιάξουμε ένα array που το 1ο του στοιχείο θα είναι 1 και μετά προσθέτουμε αναδρομικά τον αριθμό των κοινών στηλών για να βρούμε το JA. Αντίστοιχα, κάνοντας transpose το A βρίσκουμε το **CSR**.

Μετά μένει η δημιουργία του tex αρχείου που θα συμπεριλάβουμε στην αναφορά, εδώ απαιτείται η χρήση της εντολής **fopen** για να γράψουμε μέσα στο αρχείο, αφού το δημιουργήσουμε με την εντολή **edit** αν δεν υπάρχει. Εκτελώντας την εντολή **fprint** εκτυπώνουμε τους πίνακες και τα κατάλληλα tags της LaTeX

Για μητρώο:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 4 & 0 & 0 & 5 \\ 2 & 0 & 4 & 0 \\ 0 & 1 & 0 & 2 \end{pmatrix}$$

Ακολουθεί το αρχείο tex που παράγεται μετά την εκτέλεση του κώδικα σε μορφή CSR.

val =

1.0000	4.0000	5.0000	2.0000	4.0000	1.0000	2.0000

IA =

2	1	4	1	3	2	4

JA =

1	2	4	6	8

Ερώτημα 2

```
function [F] = blkToeplitzTrid(n,B,A,C)
F = cell(n);
m = size(A);
for i = 1:n
for j = 1:n
if (i == j)
F{i,j} = A;
elseif (i + 1 == j)
F{i,j} = C;
elseif (i == j + 1)
F{i,j} = B;
else
F{i,j} = zeros(m(1));
end
end
end
F = cell2mat(F);
```

Αρχικά δημιουργούμε ένα **cell array** μεγέθους $n \times n$ και αποθηκεύουμε το μέγεθος του A (θα μπορούσαμε να αποθηκεύσουμε και το B ή C αλλά έχουν το ίδιο μέγεθος). Έπειτα ξεκινάμε να φορτώνουμε το κάθε μπλοκ του **cell array**, αν βρισκόμαστε στην κεντρική διαγώνιο τότε βάζουμε το μητρώο A, αλλιώς αν είμαστε στην πάνω διαγώνιο το C, αλλιώς αν είμαστε στην κάτω το B και τέλος στα υπόλοιπα σημεία βάζουμε ένα τετραγωνικό μητρώο μεγέθους A αποτελούμενο από μηδενικά. Τελευταίο βήμα είναι η μετατροπή του **cell array** σε ένα μητρώο, το οποίο το επιτυγχάνουμε με την χρήση της συνάρτησης **cell2mat** της Matlab. Για παράδειγμα με τα μητρώα:

$$A = \begin{pmatrix} 7 & 5 & 10 \\ 7 & 1 & 2 \\ 8 & 3 & 9 \end{pmatrix} B = \begin{pmatrix} 9 & 3 & 10 \\ 1 & 9 & 2 \\ 4 & 5 & 3 \end{pmatrix} C = \begin{pmatrix} 2 & 6 & 9 \\ 2 & 6 & 7 \\ 9 & 2 & 4 \end{pmatrix}$$

Έχουμε το μητρώο F με $n = 4$:

$$F = \begin{pmatrix} 7 & 5 & 10 & 2 & 6 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 1 & 2 & 2 & 6 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 3 & 9 & 9 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 3 & 10 & 7 & 5 & 10 & 2 & 6 & 9 & 0 & 0 & 0 \\ 1 & 9 & 2 & 7 & 1 & 2 & 2 & 6 & 7 & 0 & 0 & 0 \\ 4 & 5 & 3 & 8 & 3 & 9 & 9 & 2 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 3 & 10 & 7 & 5 & 10 & 2 & 6 & 9 \\ 0 & 0 & 0 & 1 & 9 & 2 & 7 & 1 & 2 & 2 & 6 & 7 \\ 0 & 0 & 0 & 4 & 5 & 3 & 8 & 3 & 9 & 9 & 2 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 9 & 3 & 10 & 7 & 5 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 9 & 2 & 7 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 5 & 3 & 8 & 3 & 9 \end{pmatrix}$$

Ερώτημα 3

```
function [val,brow_idx,bcol_ptr] = sp_mx2bccs(A,nb)
s = size(A,1);
if (mod(s,nb) == 0)
m(1,1:s/nb) = nb;
A = mat2cell(A, m, m);
F = cellfun(@find, A, 'UniformOutput', false);
f = cellfun(@isempty, F);
Sum = sum(f(:) == 0);
val = zeros(1,Sum*nb*nb);
bcol_ptr = zeros(1,1+s/nb);
bcol_ptr(1,1) = 1;
[i,j] = find(~f);
brow_idx = transpose(i);
n = size(i, 1);
for a = 1:n
if a == 1
p = 1;
else
p = (a - 1) * nb^2 + 1;
end
for b = 1:nb
for c = 1:nb
val(1, p) = A{i(a,1),j(a,1)}(c,b);
p = p + 1;
end
end
end
for  k = 2:s/nb+1
bcol_ptr(k) = bcol_ptr(k-1) + size(j(j==k-1),1);
```

```
end
else
disp('Error, nb must have 0 remainder in division with the size of A!')
end
```

Αρχικά αφού υποθέτουμε ότι το nb είναι ίδιο ως προς της στήλες και τις γραμμές τότε πρέπει να είναι διαιρέτης του αριθμού των γραμμών ή των στηλών του τετραγωνικού μητρώου που δίνεται. Μετά τον έλεγχο αυτόν μετατρέπουμε το μητρώο A σε μορφή **cell array** οπού το κάθε cell έχει μέγεθος $nb \times nb$. Με την βοήθεια της συνάρτησης **cellfun** βρίσκουμε τα cells με τα μη μηδενικά στοιχεία και τις θέσεις του μέσα στα cells. Επίσης σημειώνουμε με λογικό 0 τα cells με μη μηδενικά στοιχεία το οποίο το χρειαζόμαστε για να βρούμε τον αριθμό των cells με μη μηδενικά στοιχεία, το οποίο άθροισμα επί το τετράγωνο του nb μας δίνει το μέγεθος του πίνακα val. Αφού κάνουμε initialize με μηδενικά τους πίνακες val και bcol_ptr αποθηκεύουμε στις μεταβλητές i, j τις γραμμές, στήλες των cells με μηδενικά στοιχεία (το i είναι στην πραγματικότητα το brow_idx).

Τώρα φτάνουμε στην εισαγωγή των cell με μη μηδενικά στοιχεία στον πίνακα val, το κάθε cell έχει nb^2 στοιχεία, οπότε το αρχικό cell θα ξεκινάει από την θέση 1 έως την θέση $nb^2 + 1$, τη 2η φορά από την θέση $2 * (nb^2) + 1$, δηλαδή $\boldsymbol{p = (a - 1) * nb^2 + 1}$ οπού a ο αριθμός των επαναλήψεων (εκτός της πρώτης φοράς που το $p = 1$). Άρα:

$$p(\alpha) = \begin{cases} 1 & \alpha = 1 \\ (\alpha - 1) * nb^2 + 1 & \alpha > 1 \end{cases}$$

Δηλαδή το p μας δίνει την εναρκτήρια θέση εισαγωγής τιμών του κάθε cell. Χρησιμοποιώντας ένα if ελέγχουμε αν είναι η 1η επανάληψη ή αλλιώς υπολογίζουμε το p. Έπειτα έχουμε 2 for loops για κάθε στοιχείο του cell (δηλαδή σύνολο επαναλήψεων nb^2) που γίνεται η εισαγωγή των τιμών στον πίνακα val. Τέλος υπολογίζουμε το bcol_ptr αναδρομικά (με 1η τιμή να είναι 1) υπολογίζοντας το άθροισμα των ίδιων τιμών του πίνακα j, δηλαδή τον αριθμό των μη μηδενικών cell στην ίδια στήλη. Για παράδειγμα με μητρώο A και $nb = 2$:

$$A = \begin{pmatrix} 4 & 2 & 4 & 2 & 0 & 0 & 0 & 0 \\ 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 2 & 5 & 0 & 0 & 0 & 0 \\ 3 & 0 & 3 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 2 & 0 & 4 & 4 & 1 & 5 \\ 5 & 1 & 3 & 2 & 1 & 3 & 0 & 0 \\ 3 & 2 & 3 & 1 & 2 & 3 & 3 & 5 \\ 4 & 4 & 1 & 2 & 3 & 4 & 5 & 3 \end{pmatrix}$$

Έχουμε τα εξής val, brow_idx, bcol_ptr:

val =

4	4	2	2	0	3	4	0	3	5	0	1	3	4	2	4	4	0	2	0	2	3	5	1
2	3	0	2	3	1	1	2	4	1	4	3	2	3	3	4	1	0	5	0	3	5	5	3

brow_idx =

1	2	3	4	1	2	3	4	3	4	3	4

bcol_ptr =

1	5	9	11	13

Ερώτημα 4

```
function [y]= spmv_bccs(y,x,nb,val,brow_idx,bcol_ptr)
n = length(bcol_ptr)-1;
p=1;
for j=1:n
pos1=bcol_ptr(j);
pos2=bcol_ptr(j+1)-1;
for w = pos1:pos2
for col=nb-1:-1:0
for i = nb-1:-1:0
y(brow_idx(w)*nb-i) = y(brow_idx(w)*nb-i) + x(j*nb-col)*val(p);
p = p + 1;
end
end
end
end
```

Αρχικά βρίσκουμε το μέγεθος του bcol_ptr αφού θα βασιστούμε σε αυτό για τις πράξεις μας (δηλαδή ο αριθμός των μπλοκ σε κάθε στήλη). Έπειτα αποθηκεύουμε το p που αρχικά δείχνει την θέση 1 του πίνακα val και αυξάνεται σε κάθε πράξη για να μας δείξει τον επόμενο αριθμό. Έπειτα υπολογίζουμε τα pos1, pos2 που μας δίνουν τον αριθμό των μπλοκ που θα γίνουν οι πράξεις.

Για κάθε στήλη που περιέχουν τα μπλοκ πραγματοποιείται η πράξη $y \leftarrow y + Ax$, σε κάθε μπλοκ γίνεται η πράξη ανά γραμμή και μετά αλλάζει η στήλη καθώς οι τιμές του val είναι σε αυτή τη σειρά. Δηλαδή λ.χ. με $nb = 2$ σε ένα τυχαίο μπλοκ έχουμε το μπλοκ w (που είναι η επανάληψη pos1 έως pos2) επί το nb πλην i που μας δίνει τις γραμμές μέσα στο μπλοκ, η επόμενη στήλη του μπλοκ καθορίζεται από την επανάληψη του col. Η τιμή του x καθορίζεται, λοιπόν, από j (δηλαδή σε ποια στήλη μπλοκ βρισκόμαστε) επί το nb (δηλαδή τον αριθμό των στηλών μέσα στο κάθε μπλοκ) πλην την επανάληψη col. Μετά από κάθε πράξη $y \leftarrow y + Ax$ αυξάνουμε το p για να μας δώσει την επόμενη τιμή του val.

Στο παράδειγμα θα χρησιμοποιήσουμε το μητρώο A σε μορφή BCCS που φτιάξαμε στο προηγούμενο ερώτημα. Οπότε ορίζουμε x, y ως εξής:

x =

1	2	3	4	5	6	7	8

y =

8	7	6	5	4	3	2	1

Και αποκτούμε αποτέλεσμα z:

z =

36	15	40	21	104	50	111	122

(Τα x, y, z είναι transposed για να είναι το κείμενο πιο ευανάγνωστο)

Ερώτημα 5

```
function [error] = erotima5(m,n)
T = toeplitz([4,-1,zeros(1,m-2)]);
S = blkToeplitzTrid(n,inv(T),T^2,T);
y = eye(n*m,1);
x = ones(n*m,1);
[val,brow_idx,bcol_ptr] = sp_mx2bccs(S,m);
y1 = y + S*x;
[y2]= spmv_bccs(y,x,m,val,brow_idx,bcol_ptr);
error = norm(y1-y2);
```

Για το ερώτημα 5 δημιουργήθηκε η function **[error] = erotima5(m,n)**, δηλαδή ο χρήστης ορίζει τα n, m και επιστρέφεται η διαφορά των διανυσμάτων $y \leftarrow y + Ax$ και το αποτέλεσμα της κλήσης της spmv_bccs ως προς τη νόρμα-2. Ουσιαστικά αυτό που κάνει είναι να δημιουργεί το μητρώο T όπως ζητείται στην εκφώνηση και το S χρησιμοποιώντας την function του ερωτήματος 2. Έπειτα δημιουργούνται τα x, y όπως ζητείται στην εκφώνηση και μετατρέπεται το S σε μορφή BCCS. Τέλος γίνεται η πράξη $y \leftarrow y + Ax$ και η κλήση της spmv_bccs και υπολογίζεται η διαφορά ως προς τη νόρμα-2. Με τιμές $n = 64$ και $m = 32$ το $error = 0$, όπως και παραμένει και με άλλες τιμές που δοκιμάστηκαν.