Εργαστηριακή Άσκηση (Μέρος 1) 2021-2022

Βασιλική Στάμου ΑΜ:1059543

16 Ιανουαρίου 2022

Τμήμα Μηχ. Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Περιεχόμενα

1 Στοιχεία υπολογιστικού συστήματος

1.1 Διάρκεια Εργασίας

08 Ιανουαρίου 2022 - 16 Ιανουαρίου 2022

1.2 Στοιχεία Υπολογιστικού Συστήματος

model	προσωπικό λάπτοπ HUAWEI
O / S	Windows 10 Home 20H2
Processor name	AMD Ryzen 5 Mobile 3500U with Radeon Vega Mobile Gfx
Processor speed	2.10GHz
Number of processors	8
Total # cores	4
Total # threads	8
FMA 3 instruction	yes
L1 CACHE	256KB Instruction, 128KB Data write-back
L2 CACHE	(per core) 512KB, write back
L3 CACHE	(shared) 4MB
Gflops /s	117.3
Memory	8GB
Memory Bandwidth	35.76 GB/s
MATLAB Version	9.9.0.1467703 (R2020b)
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 Version 3.7.0

Computer Type	LU	FFT	ODE	Sparse	2-D	3-D
Windows 10, AMD Ryzen Threadripper(TM) 3970x @ 3.50 GHz	0.1930	0.1892	0.3545	0.4085	0.1999	0.2188
Debian 9(R), AMD Ryzen Threadripper 3970x @ 3.50 GHz	0.2612	0.1259	0.3393	0.4216	0.3005	0.2809
Windows 10, Intel Xeon(R) W-2133 @ 3.60 GHz	0.4010	0.3255	0.4494	0.5081	0.3484	0.3166
Windows 10, Intel Xeon CPU E5-1650 v3 @ 3.50 GHz	0.4571	0.3189	0.4957	0.4492	0.3445	0.3922
iMac, macOS 10.15.3, Intel Core i9 @3.6 GHz	0.3286	0.2994	0.3307	0.2971	0.8115	0.5337
Windows 10, AMD Ryzen(TM) 7 1700 @ 3.00 GHz	0.7786	0.5169	0.5180	0.5948	0.3184	0.2160
Surface Pro 3, Windows(R) 10, Intel(R) Core i5-4300U @ 1.9 GHz	1.7749	0.9768	0.7254	0.6882	0.6982	0.6290
MacBook Pro, macOS 10.15.2, Intel Core i5 @ 2.6 GHz	1.5406	0.9646	0.6172	0.6083	2.0698	1.4805
This machine	2.8724	0.9069	0.9126	0.8812	3.1879	3.5122

Σχήμα 1: Αποτελέσματα της bench

2 Αραιές αναπαραστάσεις και κατασκευές μητρώων

2.1 sp_mat2latex

Στην παρούσα άσκηση κατασκευάστηκε η συνάρτηση $sp\_mat2latex$ η οποία επιστρέφει σε κώδικα LaTeX, την αραιή αναπαράσταση CSR ή CSC ενός μητρώου το οποίο είναι σε αραιή μορφή στη MATLAB. Δηλαδή, δοθέντος τύπου sparse και string sp_type, ο κώδικας παράγει εντολές LaTeXπου παράγουν τους πίνακες val, row_ptr, col_idx.
Για την κατασκευή της, φάνηκε χρήσιμη η συνάρτηση $matrix2latex.m$ του M. Koehler (διαθέσιμη από Matlab File Exchange).
Στη συνέχεια παρουσιάζεται η εκτέλεση της συνάρτησης για το μητρώο του παραδείγματος.

$$\text{Για το μητρώο A} = \begin{pmatrix} 0.3984 & 0.1895 & 0.8423 & 0 \\ 0 & 0.5458 & 0 & 0 \\ 0.9416 & 0.4122 & 0.1788 & 0 \\ 0 & 0 & 0.7134 & 0 \end{pmatrix}, \text{έχουμε :}$$

$$val = \boxed{0.3984} \boxed{0.1895} \boxed{0.8423} \boxed{0.5458} \boxed{0.9416} \boxed{0.4122} \boxed{0.1788} \boxed{0.7134}$$

$$IA = \boxed{1} \boxed{2} \boxed{3} \boxed{2} \boxed{1} \boxed{2} \boxed{3} \boxed{3}$$

$$JA = \boxed{1} \boxed{4} \boxed{5} \boxed{8} \boxed{9}$$

Matlab Code

```
function [val,row_ip,col_ip]=sp_mat2latex(A,sp_type)
%Author: V. Stamou, AM 1059543, Date: 12/1/2022
alignment = 'l';
fid = fopen(sp_type, 'w');


if sp_type=='csc'
   %%CRC
   [row_ip,J,val] = find(A);
   col_ip = accumarray(J+1,1);
   col_ip = cumsum(col_ip)+1;

   length_row_ip=size(row_ip,1);length_col_ip=size(col_ip,1);length_val=size(val,1);


elseif sp_type=='csr'
   %%CRS
   [col_ip,J,val] = find(A');
   row_ip = accumarray(J+1,1);
   row_ip = cumsum(row_ip)+1;

   length_row_ip=size(row_ip,1);length_col_ip=size(col_ip,1);length_val=size(val,1);

end


%%%
col_ip = num2cell(col_ip);row_ip = num2cell(row_ip);val = num2cell(val);

for lr=1:length_row_ip
   row_ip{lr} = num2str(row_ip{lr});
end
for lr=1:length_col_ip
   col_ip{lr} = num2str(col_ip{lr});
end
```

```
for lr=1:length_val
   val{lr} = num2str(val{lr});
end
%%%

%val
fprintf(fid,'\\\\\r\nval=');
      fprintf(fid, '\\begin{tabular}{|');
      for i=1:length_val
         fprintf(fid, '%c|', alignment);
      end
      fprintf(fid, '}\r\n');
      fprintf(fid, '\\hline\r\n');

      for lv=1:length_val-1
         fprintf(fid, '%s&', val{lv});
      end
      fprintf(fid, '%s\\\\\\hline\r\n', val{length_val});
      fprintf(fid, '\\end{tabular}\r\n');

if sp_type=='csc'
%ri
fprintf(fid,'\\\\\r\nIA=');
      fprintf(fid, '\\begin{tabular}{|');

      for i=1:length_row_ip
         fprintf(fid, '%c|', alignment);
      end
      fprintf(fid, '}\r\n');
      fprintf(fid, '\\hline\r\n');

      for lr=1:length_row_ip-1
         fprintf(fid, '%s&', row_ip{lr});
      end
      fprintf(fid, '%s\\\\\\hline\r\n', row_ip{length_row_ip});
      fprintf(fid, '\\end{tabular}\r\n');
%cp
fprintf(fid,'\\\\\r\nJA=');
      fprintf(fid, '\\begin{tabular}{|');

      for i=1:length_col_ip
         fprintf(fid, '%c|', alignment);
      end
      fprintf(fid, '}\r\n');
      fprintf(fid, '\\hline\r\n');

      for lc=1:length_col_ip-1
         fprintf(fid, '%s&', col_ip{lc});
      end
      fprintf(fid, '%s\\\\\\hline\r\n', col_ip{length_col_ip});
      fprintf(fid, '\\end{tabular}\r\n');

elseif sp_type=='csr'
%ci
fprintf(fid,'\\\\\r\nIA=');
      fprintf(fid, '\\begin{tabular}{|');

      for i=1:length_col_ip
         fprintf(fid, '%c|', alignment);
      end
      fprintf(fid, '}\r\n');
      fprintf(fid, '\\hline\r\n');
```

```
98
99        for lc=1:length_col_ip-1
100          fprintf(fid, '%s&', col_ip{lc});
101       end
102       fprintf(fid, '%s\\\\\\hline\r\n', col_ip{length_col_ip});
103       fprintf(fid, '\\end{tabular}\r\n');
104
105   %rp
106   fprintf(fid,'\\\\\r\nJA=');
107       fprintf(fid, '\\begin{tabular}{|');
108
109       for i=1:length_row_ip
110          fprintf(fid, '%c|', alignment);
111       end
112       fprintf(fid, '}\r\n');
113       fprintf(fid, '\\hline\r\n');
114
115       for lr=1:length_row_ip-1
116          fprintf(fid, '%s&', row_ip{lr});
117       end
118       fprintf(fid, '%s\\\\\\hline\r\n', row_ip{length_row_ip});
119       fprintf(fid, '\\end{tabular}\r\n');
120
121   end
122
123      fclose(fid);
```

2.2 blkToeplitzTrid

Κατασκευάστηκε συνάρτηση $blkToeplitzTrid(n, B, A, C)$ που δοθέντων των τετραγωνικών μητρώων A, B, C μεγέθους $m \times m$, κατασκευάζει σε αραιή μορφή το μπλοκ Toeplitz τριδιαφώνιο μητρώο:

$$\begin{pmatrix} A & C & 0 & \dots & 0 \\ B & A & C & 0 & \vdots \\ 0 & B & A & C & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & B & A & C \\ 0 & \dots & 0 & B & A \end{pmatrix}$$

Το μητρώο έχει n μπλοκ στην κύρια διαγώνιο και επομένως είναι $mn \times mn$.
Λήφθηκε υπόψιν η ανάπτυξη κώδικα με όσο το δυνατόν λιγότερες γραμμές και κλήσεις των build-in συναρτήσεων της MATLAB. Για την υλοποίηση χρησιμοποιήθηκε indexing.

Matlab Code

```
function T=blkToeplitzTrid(n,B,A,C)
%Author: V. Stamou, AM 1059543, Date: 7/1/2022

%Verify the inputs are 2-d arrays.
  if (length(size(A))~=2) || ...
     (length(size(B))~=2) || ...
     (length(size(C))~=2)
    error 'Inputs must be 2d arrays if a replication factor is provided'
  end

%Get block sizes & check for consistency
  [m,q] = size(A);
  if m~=q
    error 'Blocks must be square arrays'
  end
  if isempty(A)
    error 'Blocks must be non-empty arrays or scalars'
  end
  if any(size(A)~=size(B)) || any(size(A)~=size(C))
    error 'A, B, C are not identical in size'
  end
  if isempty(n) || (length(n)>1) || (n<1) || (n~=floor(n))
    error 'n must be a positive scalar integer'
  end

%Scalar inputs
  if m==1
    if n==1
      T = A;
    else
      T = spdiags(repmat([B A C],n,1),-1:1,n,n);
    end
    return
  end

  %Main diagonal elements of each array
  v = repmat(A(:),n,1);
  if n>1
    %sub-diagonal
```

```
    v=[v;repmat(B(:),n-1,1)];
    %super-diagonal
    v=[v;repmat(C(:),n-1,1)];
  end


%Generate the index arrays
 %main diagonal
[ind1,ind2,ind3]=ndgrid(0:m-1,0:q-1,0:n-1);
rind = 1+ind1(:)+m*ind3(:);
cind = 1+ind2(:)+q*ind3(:);
% then the sub and super diagonal blocks.
if n>1
  %sub-diagonal
  [ind1,ind2,ind3]=ndgrid(0:m-1,0:q-1,0:n-2);
  rind = [rind;1+m+ind1(:)+m*ind3(:)];
  cind = [cind;1+ind2(:)+q*ind3(:)];
  %super-diagonal
  rind = [rind;1+ind1(:)+m*ind3(:)];
  cind = [cind;1+q+ind2(:)+q*ind3(:)];
end
%final array
T = sparse(rind,cind,v,n*m,n*q);
```

2.3 sp_mx2bccs

Κατασκευάστηκε η συνάρτηση $sp\_mx2bccs$ που δοθέντος ενός τετραγωνικού αραιού μητρώου A και ενός ακεραίου nb (block size), επιστρέφει μια αναπαράσταση που την ονομάζουμε BCCS (block compressed column storage).
Η BCCS είναι η κατά μπλοκ στήλες αναπαράσταση που αντιστοιχεί στην (εκτενέστερα μελετημένη στη βιβλιογραφία) BCRS (block compressed row storage).
Μια απλοποίηση εδώ είναι ότι υποθέτουμε το ίδιο nb ως προς της στήλες και τις γραμμές.
Το διάνυσμα val αποτελείται απο όλες τις τιμές των μη μηδενικών μπλοκ εισάγοντάς τις κατά στήλες. Το διάνυσμα $brow\_idx$ περιέχει πληροφορία για την γραμμή μπλοκ του που αρχίζει το κάθε μπλοκ και το διάνυσμα $bcol\_prt$ μας πληροφορεί πόσα μπλοκ υπάρχουν σε κάθε στήλη μπλοκ του .

Matlab Code

```
function [val,brow_idx,bcol_ptr]=sp_mx2bccs(A,nb)
%Author: V. Stamou, AM 1059543, Date: 8/1/2022
dim = ones(1,size(A,1)/nb)*nb;
C = mat2cell(A, dim, dim);
non_zeros=cellfun(@(x) nnz(x)~=0, C); % non-zero blocks
D=C(non_zeros);
v=[];

%Find val
for i=1:size(D,1)
     Dblock = cell2mat(D(i));
         Dblock = Dblock(:)';
         v = [v Dblock];
end
val=full(v);

%Find brow_idx & brow_ptr
[ri,co,values]=cellfun(@(x)find(x),C,'UniformOutput',false);
[brow_idx,J] = find(~cellfun(@isempty,ri));
bcol_ptr = accumarray(J+1,1);
bcol_ptr = cumsum(bcol_ptr)+1;
```

2.4 spmv_bccs

Κατασκευάστηκε συνάρτηση που δέχεται ως είσοδο ένα μητρώο σε αναπαράσταση BCCS (δηλ. τους πίνακες $val, brow\_idx, bcol\_ptr$) με τις τιμές τους) και επιστρέφει το διάνυσμα $y \leftarrow y + Ax$. Χρησιμοποιεί την εν λόγω αραιή αναπαράσταση και τα διανύσματα στην είσοδο και έξοδο είναι όλα σε κανονική (πυκνή) αναπαράσταση.
Στην υλοποίηση του συγκεκριμένου ερωτήματος έπρεπε να αναλογιστούμε τα τρία διανύσματα του BCCS, τι αναπαραστούν και την σύνδεση μεταξύ τους.
Συγκεκριμένα, τα στοιχεία του $bcol\_ptr$ μας δείχουν πόσα μη-μηδενικά υπομητρώα έχουμε συναντήσει συνολικά στις προηγούμενες στήλες, θεωρόντας ως τωρινή στήλη την θέση μέσα στο διάνυσμα $bcol\_ptr$. Έτσι με μία απλή αφαίρεση διαδοχικών τιμών του $bcol\_ptr$, έχουμε το πλήθος των στοιχείων (μητρώων) που χρειαζόμαστε από το διάνυσμα val. Το διάνυσμα val κρατάει την πληροφορία μας και πρέπει να θυμόμαστε ότι δουλέυουμε σε block άρα κάθε στοιχείο των δύο υπόλοιπων διανυσμάτων της BCCS αναπαράστασης αντιστοιχούν σε nb^2 στοιχεία του val. Τέλος, το διάνυσμα $brow\_idx$, δεν είναι τίποτα άλλο από την θέση στον άξονα των γραμμών των υπομητρώων στο A με την σειρά που "ανακαλύφθηκαν" στο προηγούμενο ερώτημα.
Αναλογίζοντας τα προαναφερθέν και συμπληρώνοντας ότι τόσο το διάνυσμα x όσο και το διάνυσμα y θα είναι διαστάσεων length(A)$\times$1 κατανοώντας την πράξη του πολλαπλασιασμού ανάμεσα σε μητρώο και διάνυσμα καταλήξαμε στο παρακάτω κώδικα ο οποίος αποτελείται από μερικούς μετρητές και αναδιάταξεις των διανυσμάτων στον χώρο.

Matlab Code

```
function y=spmv_bccs(y,x,nb,val,brow_idx,bcol_ptr)
%Author: V. Stamou, AM 1059543, Date: 11/1/2022
block=1;
k=0;
col_counter=1;

for reps=2:size(bcol_ptr,1)
   for t=1:bcol_ptr(reps)-bcol_ptr(reps-1)
      for i=1:nb
         y(nb*(brow_idx(block)-1)+i)=y(nb*(brow_idx(block)-1)+i) + val(k+i)*x(col_counter);
      end
   k=k+nb;
   for l=col_counter+1:col_counter+nb-1
      for j=1:nb
         y(nb*(brow_idx(block)-1)+j)=y(nb*(brow_idx(block)-1)+j) + val(k+j)*x(l);
      end
      k=k+nb;
   end
      block=block+1;
   end
   col_counter=col_counter+nb;
end
```

2.5 Προσομοίωση

Θέτοντας $T = toeplitz([4, -1, zeros(1, m-2)$, όπου $m = 32$, χρησιμοποιήθηκε την παραπάνω αναπαράσταση για το μπλοκ τριδιαγώνιο μητρώο $S = blkToeplitzTrid(n, inv(T), T^2, T)$ και να υπολογίστηκε η τιμη του $y = y + A * x$ χρησιμοποιώντας την $spmv\_bccs$ με $nb = m$ για τα διανύσματα $y = eye(n * m, 1)$, $x = ones(n * m, 1)$ και $n = 64$.
Η διαφορά των διανυσμάτων που υπολογίζονται με την εντολή της MATLAB $y = y + A * x$ και από την κλήση της $spmv\_bccs$ ως προς τη νόρμα-2 είναι $d = 0$.

Matlab Code

```
%Author: V. Stamou, AM 1059543, Date: 15/1/2022
clear;clc;
n=64; m=32; nb=m;
y=eye(n*m,1); x=ones(n*m,1);
yy=y;
T=toeplitz([4,-1,zeros(1,m-2)]);
S=blkToeplitzTrid(n,inv(T),T^2,T);

yy=yy+S*x;

[val,brow_idx,bcol_ptr]=sp_mx2bccs(S,nb);

y=spmv_bccs(y,x,nb,val,brow_idx,bcol_ptr);

d=norm(y-yy);
```