Sparse matrix

```
[christian@octet1 pardiso]$ ls
Makefile  ddtpara.nml  fort.67  fort.97  fort.99    lupara.nml  luresult.txt       lusedfile  luv03.f
a.out     fort.66      fort.96  fort.98  lu_supp.f  lupara.src  luresult.txt.0620  luv02.f    run.sh
[christian@octet1 pardiso]$ vi lupara.nml
```

fort.97

```
[christian@octet1 pardiso]$ cat fort.97
           1   138.134516138841
           2   12.9084315707789
           3  -156.389795135202
           4   151.317426039246
           5   63.6850275068084
           6  -241.381655659478
           7   88.1345161388412
           8   83.6191096894337
           9  -206.389795135202
```

row index

RHS

<NOTE> fort.97 record the RHS

```
[christian@octet1 pardiso]$ cat fort.96
           1  0.508650302437926
           2   3.01890067049449
           3   1.63693388521240
           4  -1.93462507044465
           5  7.765993938798713E-002
           6   2.07728225555734
           7  0.502293343287600
           8  -0.777289771612455
           9  0.677233979695705
```

row index

Solution vector

<NOTE> fort.96 record the solution vector

```
[christian@octet1 pardiso]$ ls
Makefile  ddtpara.nml  fort.67  fort.97  fort.99    lupara.nml  luresult.txt       lusedfile  luv03.f
a.out     fort.66      fort.96  fort.98  lu_supp.f  lupara.src  luresult.txt.0620  luv02.f    run.sh
[christian@octet1 pardiso]$ vi lupara.nml
```

Take dimension by 9

fort.98

```
[christian@octet1 pardiso]$ cat fort.98
           1           1
           2           4
           3           8
           4          11
           5          15
           6          20
           7          24
           8          27
           9          31
          10          34
```

row index

The first nonzero element's index

There are 3 nonzero element in row1
There are 4 nonzero element in row2
There are 3 nonzero element in row3
…
…
…
There are 3 nonzero element in row10

10 exceed the dimension => end of matrix

<NOTE> fort.98 help us to know how many nonzero element in each row

```
[christian@octet1 pardiso]$ ls
Makefile  ddtpara.nml  fort.67  fort.97  fort.99    lupara.nml  luresult.txt       lusedfile  luv03.f
a.out     fort.66      fort.96  fort.98  lu_supp.f  lupara.src  luresult.txt.0620  luv02.f    run.sh
[christian@octet1 pardiso]$ vi lupara.nml
```

```
[christian@octet1 pardiso]$ cat fort.99
          1          1 -64.0000000000000
          1          2  157.421356237309
          1          4  157.421356237309
          2          1 -125.421356237309
          2          2 -64.0000000000000
          2          3  157.421356237309
          2          5  157.421356237309
          3          2 -125.421356237309
          3          3 -64.0000000000000
          3          6  157.421356237309
          4          1 -125.421356237309
          4          4 -64.0000000000000
          4          5  157.421356237309
          4          7  157.421356237309
          5          2 -125.421356237309
          5          4 -125.421356237309
          5          5 -64.0000000000000
          5          6  157.421356237309
          5          8  157.421356237309
          6          3 -125.421356237309
          6          5 -125.421356237309
          6          6 -64.0000000000000
          6          9  157.421356237309
          7          4 -125.421356237309
          7          7 -64.0000000000000
          7          8  157.421356237309
          8          5 -125.421356237309
          8          7 -125.421356237309
          8          8 -64.0000000000000
          8          9  157.421356237309
          9          6 -125.421356237309
          9          8 -125.421356237309
          9          9 -64.0000000000000
```

The row index

The column index

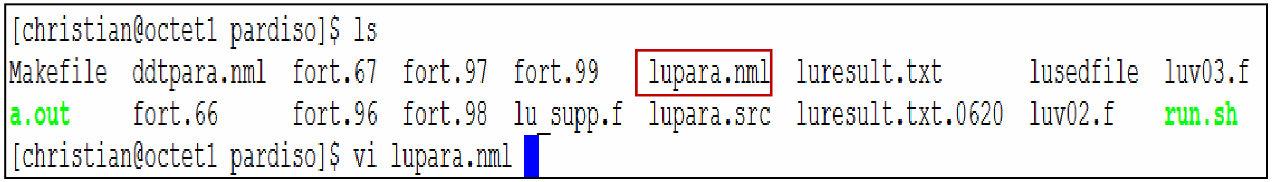
The entity of corresponding index

<NOTE> fort.99 actually record the matrix

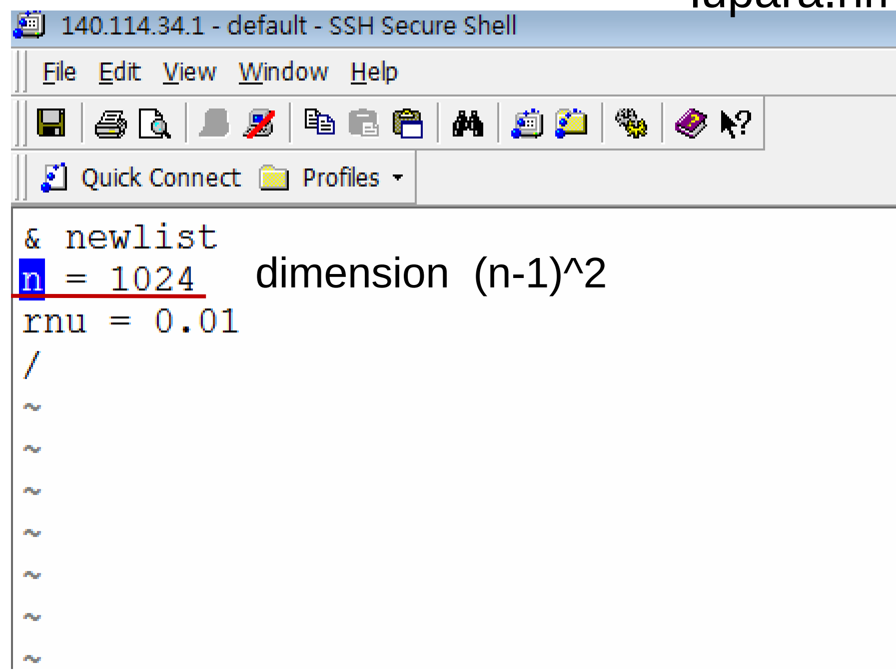
procedure

- Step I : Modify parameter n to change the dimension of matrix A
- Step II : Execute the programming in fortran to get the solution vector
- Step III : Load fort.96, fort.97, fort.99 into matlab to get data we need
- Step IV : Construct a sparse matrix then get solution vector
- Step V : Compute the SupNorm of solution vector in fortran and in matlab

Step I : set the dimension of sparse matrix



lupara.nml



dimension (n-1)^2

In this experiment

We use n =128, 256, 512, 1024 consecutively

<NOTE> we use lipara.nml to change the dimension of our matrix

Step II : make project and execute

```
[christian@octet1 pardiso]$ make
ifort -O2 -r8 -ipo \
        -L/usr/lib64 -lgfortran -lgomp /opt/pardiso/libpardiso_GNU42_INTEL64_INT
lapack -lptcblas -lptf77blas -latlas -L/usr/lib64 -lpthread \
        luv02.f lu_supp.f
ipo: remark #11000: performing multi-file optimizations
ipo: remark #11005: generating object file /tmp/ipo_iforte2WQit.o
luv02.f(262): (col. 7) remark: LOOP WAS VECTORIZED.
luv02.f(265): (col. 7) remark: LOOP WAS VECTORIZED.
luv02.f(270): (col. 7) remark: LOOP WAS VECTORIZED.
luv02.f(278): (col. 7) remark: LOOP WAS VECTORIZED.
luv02.f(286): (col. 7) remark: LOOP WAS VECTORIZED.
luv02.f(294): (col. 7) remark: LOOP WAS VECTORIZED.
luv02.f(302): (col. 7) remark: LOOP WAS VECTORIZED.
```

```
[christian@octet1 pardiso]$ ./a.out
 rnu =   1.000000000000000E-002

================  PARDISO: solving a real nonsymmetric system  ================


Summary PARDISO: ( reorder to reorder )
==================

Times:
======

      Time fulladj: 0.000000 s
      Time reorder: 0.000000 s
      Time symbfct: 0.000000 s
      Time parlist: 0.000000 s
      Time malloc : 0.000000 s
      Time total  : 0.000000 s total - sum: 0.000000 s
```

In execution , showing some information to us

Step III : load the file into matlab [1]

Load fort.96 and fort.97, then get the RHS and solution vector

```
>> load fort.96
>> Y = fort(:,2)

Y =

    0.5087
    3.0189
    1.6369
   -1.9346
    0.0777
    2.0773
    0.5023
   -0.7773
    0.6772
```

Solution vector

```
>> load fort.97
>> F = fort(:,2)

F =

  138.1345
   12.9084
 -156.3898
  151.3174
   63.6850
 -241.3817
   88.1345
   83.6191
 -206.3898
```

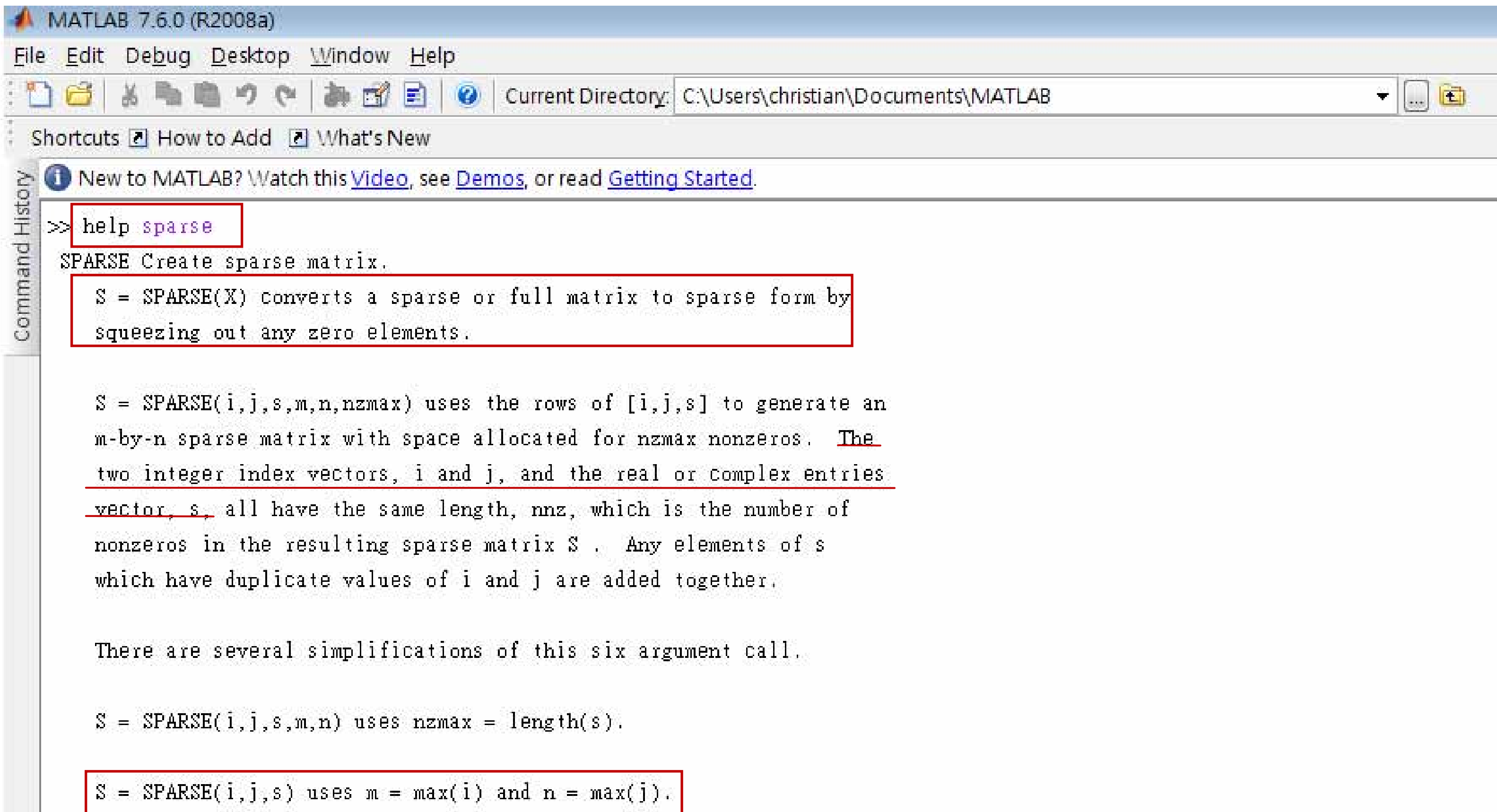
RHS

<NOTE> we ignore the first column, because it is index

Step III : load the file into matlab [2]

Load fort.99, then get the parameter I, J, S

```
>> load fort.99
>> I = fort(:,1);
>> J = fort(:,2);
>> S = fort(:,3);
>> load fort.99
>> I = fort(:,1)
```



Step IV : Construct a sparse matrix then get solution vector

Construct sparse A

```
>> A = sparse(I,J,S)

A =

   (1,1)      -64.0000
   (2,1)     -125.4214
   (4,1)     -125.4214
   (1,2)      157.4214
   (2,2)      -64.0000
   (3,2)     -125.4214
   (5,2)     -125.4214
   (2,3)      157.4214
   (3,3)      -64.0000
   (6,3)     -125.4214
   (1,4)      157.4214
   (4,4)      -64.0000
   (5,4)     -125.4214
   (7,4)     -125.4214
   (2,5)      157.4214
   (4,5)      157.4214
   (5,5)      -64.0000
   (6,5)     -125.4214
   (8,5)     -125.4214
   (3,6)      157.4214
   (5,6)      157.4214
   (6,6)      -64.0000
   (9,6)     -125.4214
```

```
   (4,7)      157.4214
   (7,7)      -64.0000
   (8,7)     -125.4214
   (5,8)      157.4214
   (7,8)      157.4214
   (8,8)      -64.0000
   (9,8)     -125.4214
```

Do solute

```
>> format long
>> A \ F

ans =

   0.50865030243793
   3.01890067049449
   1.63693388521240
  -1.93462507044465
   0.07765993938798
   2.07728225555734
   0.50229334328760
  -0.77728977161245
   0.67723397969570
```

Step IV : Compute the SupNorm of solution vector in fortran and in matlab

```
>> X = A \ F

X =

   0.50865030243793
   3.01890067049450
   1.63693388521241
  -1.93462507044465
   0.07765993938799
   2.07728225555734
   0.50229334328760
  -0.77728977161245
   0.67723397969571

>> max (abs(X-Y))

ans =

     8.659739592076221e-15
```

Compute the SupNorm

```
>> help umfpack
 UMFPACK  A sparse LU factorization package
    UMFPACK is the sparse LU factorization and solve package that is used
    by the sparse LU factorization [L,U,P,Q] = lu(A) and sparse general
    solve X = A \ B when A is sparse, square but not diagonal, banded as
    defined by spparms('bandden'), triangular, a permutation of a
    triangular matrix or symmetric positive definite.
    Note: if UMPACK is used within sparse general backslash, it may use
    either AMD or a modified COLAMD reordering routine first.

    UMFPACK Version 5.0 is written and copyrighted by Timothy A. Davis.

    Availability:

      http://www.cise.ufl.edu/research/sparse/umfpack

    See also lu, slash, amd, colamd.
```

N = 128
SupNorm = 8.049116928532385e-16

N = 256
SupNorm = 6.383782391594650e-15

N = 512
SupNorm = 1.434963259328015e-14

N = 1024
SupNorm = 1.975294927625271e-13

When n increase 2 times, dimension increase 4 times roughly.

Conclusion
When n become double , we lose one point accuracy