OpenATLib and Xabclib

Developer's Manual for Version Beta

Information Technology Center, The University of Tokyo and
Central Research Laboratory, Hitachi Ltd.

January 29, 2010

DISCLAIMER

1. Overview

In this manual, functions for numerical library developers in OpenATLib and Xabclib are explained. Fig. 1-1 and Fig. 1-2 show the components of function on Xabclib and Xabclib.

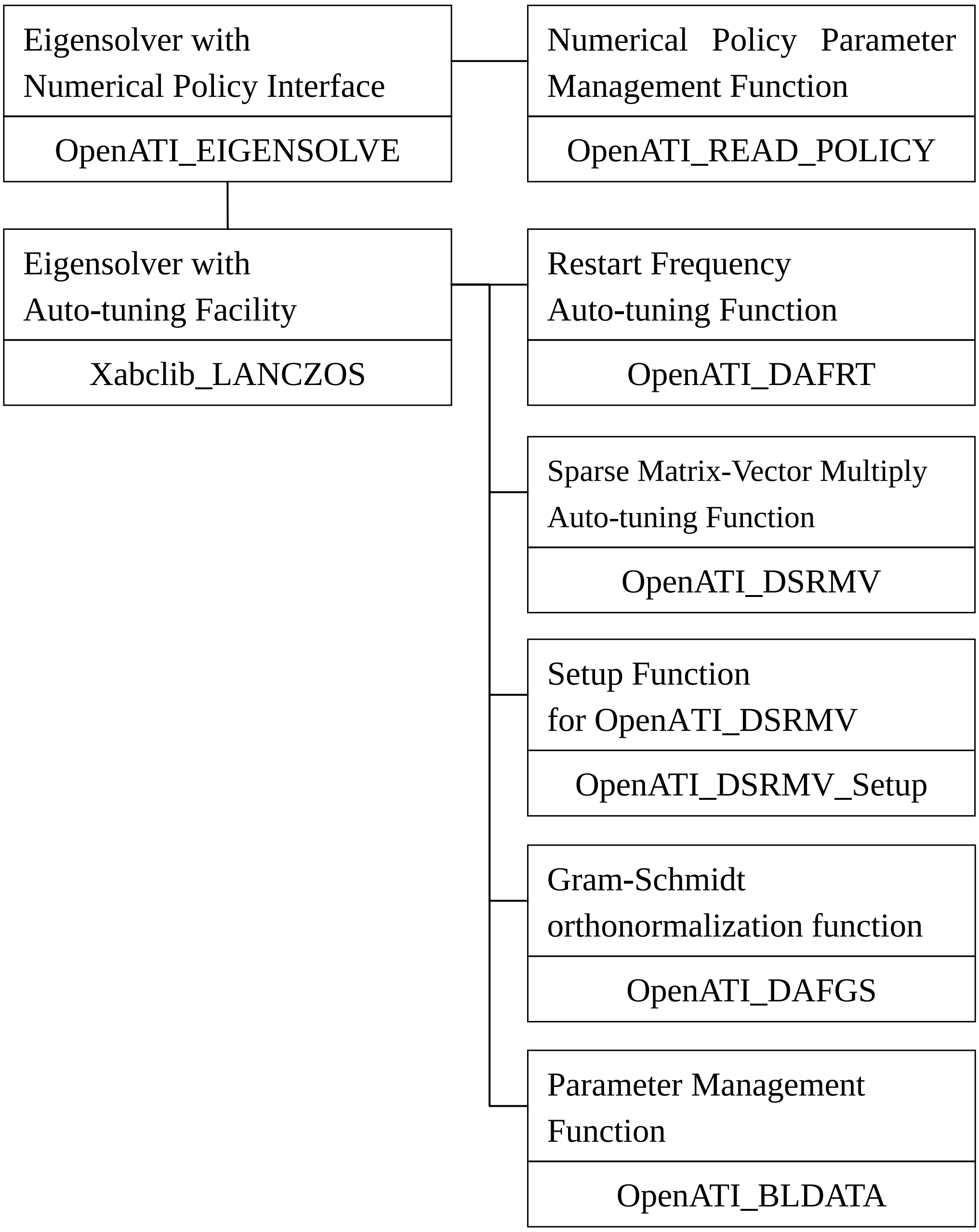

Fig. 1-1 Components of Function on Eigensolver.

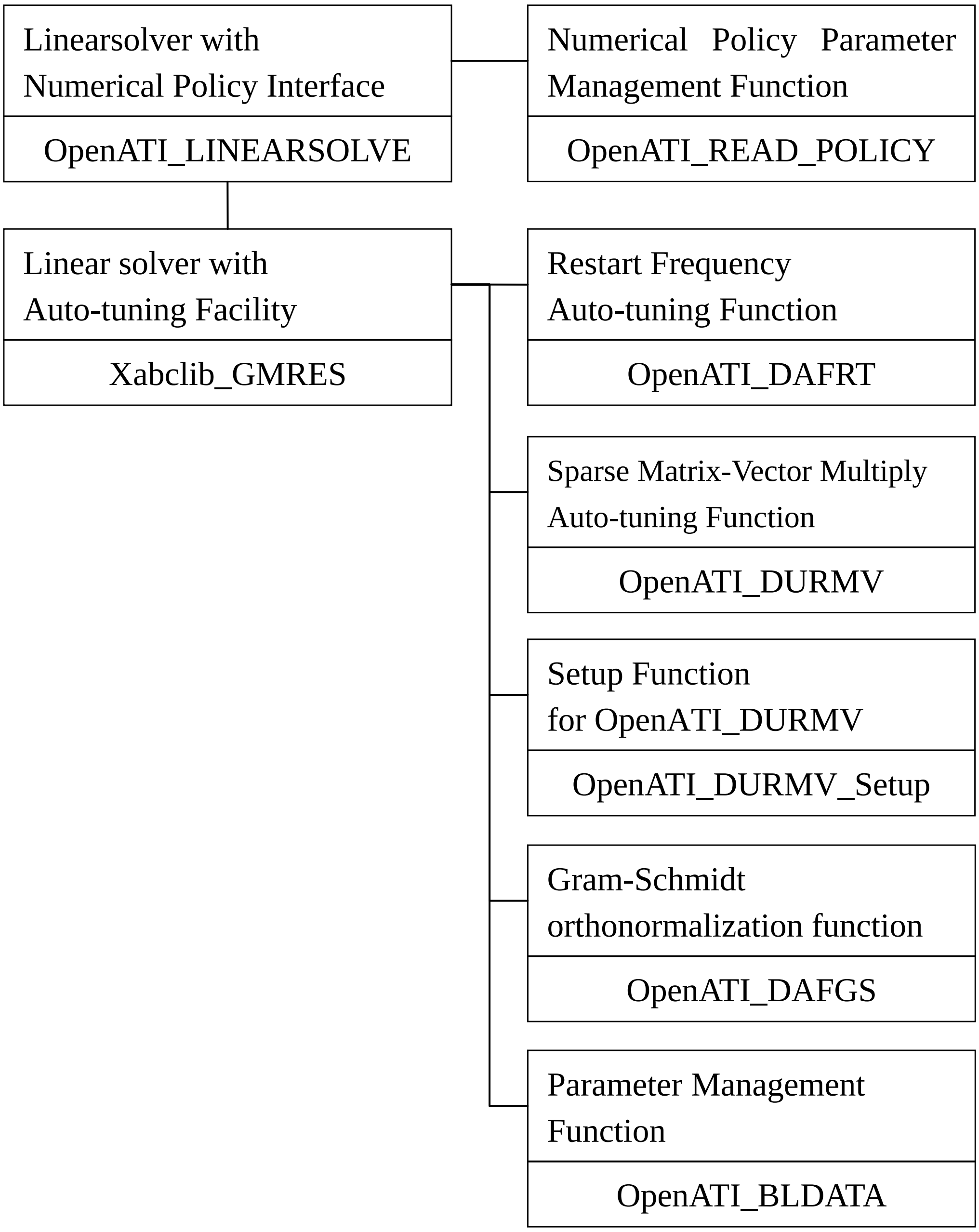

Fig. 1-2 Components of Function on Linearsolver.

2. OpenATLib : A Common Auto-tuning Interface Library

2.1 Function of OpenATLib and Its Usage

In this section, library for functions and specification on a common auto-tuning interface, named OpenATLib, is explained. OpenATLib is an Application Programming Interface (API) to supply auto-tuning facility on arbitrary matrix computation libraries. For example, estimation function for the best values on algorithmic parameters, and best implementation for sparse matrix-vector multiplication (SpMxV).

(1) The function

Table 2-1 shows auto-tuning functions providing OpenATLib.

Table 2-1 Auto-tuning Function Providing OpenATLib

Function Name	Description
OpenATI_DAFRT	Judge increment for restart frequency on Krylov subspace.
OpenATI_DSRMV	Judge the best implementation for double precision symmetric SpMxV on CRS format.
OpenATI_DURMV	Judge the best implementation for double precision non-symmetric SpMxV on CRS format.
OpenATI_DSRMV_Setup	Setup function for OpenATI_DSRMV.
OpenATI_DURMV_Setup	Setup function for OpenATI_DURMV.
OpenATI_DAFGS	Gram-Schmidt orthonormalization function with 4 implementations.
OpenATI_BLDATA	Set default parameters. (Block data format for Fortran.)
OpenATI_LINEARSOLVE	Over-LinearSolver with numerical policy interface.
OpenATI_EIGENSOLVE	Over-EigenSolver with numerical policy interface.

The functions provided OpenATLib are classified for the following five categories:

a) Computation Function (Ex. SpMxV)
b) Auxiliary Function (Ex. Specified parameter settings.)
c) Management Function (Ex. OpenATI_BLDATA)
d) Setup Function (Ex. OpenATI_DSRMV_Setup)

e) Over-Solver (Ex. OpenATI_LINEARSOLVE)

For a) and b) functions, the function names are named by the manner on Table 2-1, following "OpenATI_" .

Table 2-2 Nomenclature of OpenATLib functions

First Character	The character shows data type. S : Single Precision D : Double Precision
Second and Third Characters	If the function is auxiliary, it comes "AF". If the function is computation, it comes matrix kinds in the second character, and matrix storage format in the third character. ● The second character: S : Symmetric. U : Non-symmetric. D : Diagonal. T : Tridiagonal. ● The third character: R : CRS Format. C : CCS Format.
Fourth and Fifth Characters	Process Kinds. MV: Matrix-vector multiplication. RT: Restart frequency.

(2) Include file “OpenAT.inc”

If you include OpenAT.inc in your program, you can refer and update the following system global variables without definition. After the values are updated, all inner parameters on each OpenATI function are set to the updated values. See each specification for the details of system global variables.

(a) OpenATI_DAFRT_IPARM_1

A flag to perform auto-tuning based on MM ratio.

(b) OpenATI_DAFRT_RPARM_1

The MM ratio.

(c) OpenATI_DSRMV_IPARM_1

A search area parameter for symmetric SpMxV.

(d) OpenATI_DURMV_IPARM_1

A search area parameter for non-symmetric SpMxV.

(e) OpenATI_DURMV_IPARM_2

The number of iteration to evaluate non-symmetric SpMxV.

(f) OpenATI_DAFGS_IPARAM_1

The implementation of Gram-Schmidt orthonormalization.

(3) How to use the OpenATLib.

If you want to develop own library using OpenATLib, you should follow the following processes.

1. Put the include file of "OpenAT.inc", and static library of "libOpenAT.a" to current directory.
2. Include "OpenAT.inc" in program on own library source code, like Fig. 2-1.
3. Call target functions of OpenATLib on own library source code.
4. Describe makefile to link "libOpenAT.a".

```
INCLUDE   "OpenAT.inc"
```

Fig. 2-1 An Example of OpenATLib including.

2.2 OpenATI_DAFRT

2.2.1 Overview of the function

To perform Krylov subspace method, for example, Lanczos method for eigensolvers computation and GMRES method for linear equation solvers, they need to specify the dimension of the inner Krylov subspace to fix available memory space. If the iteration number is over for the fixed dimension, new computation is done with the current calculated approximation as initial vector to make new Krylov subspace. This process is called "restart", and the number of iterations is called "restart frequency". If the restart frequency is too small, it causes stagnation of reduction for residual vector, which is calculated by real solution and approximation vectors, then the number of iterations is increased. On the other hand, if the restart frequency is too big, it causes heave computation to make big Krylov subspaces, hence the execution time is very increased. The best frequency depends on input sparse matrix numerical condition, and it is very tough to estimate the best frequency without execution. Hence in the library point of view, we need on the fly, namely run-time, auto-tuning facility.

OpenATI_DAFRT enables us to judge the incensement of frequency based on the current information of Krylov subspace.

2.2.2 Overview of the auto-tuning method

The previous estimation for the best restart frequency is difficult; it can detect stagnation based on the run-time history of residuals. The method is proposed in [1].

The norm of the stagnation is defined by the value that maximum value divided by minimal vale from t-th time to s-th time. The values called "Ratio of Max-Min in residual". Hereafter, we describe the ratio "**MM ratio**" for simplification.

The MM ratio to past t-th time, namely $Ri(s,t)$, can be described with i-th residual r_i as follows:

$$R_i(s,t) = \frac{\max_z \{r_i(z); z = s-t+1, \cdots, s\}}{\min_z \{r_i(z); z = s-t+1, \cdots, s\}}.$$

If restart frequency is big enough, the residual tends to reduce bigly, hence MM ratio is going to be big. If restart frequency is small, it tends to cause stagnation, hence MM ratio is going to be small. Hence, we can control restart frequency at run-time monitor for the MM ratio. If the MM ratio is going to be small to a fixed value at run-time, the frequency should be increased.

2.2.3 Argument Details and Error Code

(1) Argument Details

Argument	Type	IO	Description
NSAMP	Integer	INPUT	The number of sampling points.
SAMP (NSAMP)	Double	INPUT	The values of sampling points.
IRT	Integer	OUTPUT	0 : Do not need to increase restart frequency. 1 : Need to increase restart frequency.
INFO	Integer	OUTPUT	Error code.

(2) Global Variables Defined on "OpenAT.inc"

Variable Name	Type	Initial Value	Description
OpenATI_DAFRT_IPARM_1	Integer	1	1 : Judge incensement of restart frequency based on MM ratio.
OpenATI_DAFRT_RPARM_1	Double	100.0	Threshold value for MM ratio.

(3) Error Code

Value	Description
0	Normal return.

2.2.4 Usage Example

Judgment of restart frequency is per 5 iterations. If it is needed to increase, the frequency is increased by stridden 1. In this case, you can write the code like Fig. 2-2.

```
//Parameter Definition
INCLUDE "OpenAT.inc"  // Include OpenAT.inc
MSIZE=1               // Initial restart frequency.
I=5                   // Judgment frequency.
                              ～ omission ～
IF RSDID < TOL   RETURN     // Convergence Test

SAMP (K)=RSDID     //Set residual to SAMP(K).

IF (mod (K, I) .eq. 0)   THEN     //Call DAFRT per I times.
        IRT=0
        CALL   OpenATI_ DAFRT (I, SAMP,IRT,INFO)

        IF IRT= 1   MSIZE=MSIZE+1     //Increase restart frequency.
        K=0
END IF

K=K+1
                              ～ omission ～
```

Fig. 2-2 An Example of OpenATI_DAFRT description.

2.3 OpenATI_DSRMV and OpenATI_DURMV, OpenATI_DSRMV_Setup, OpenATI_DURMV_Setup

2.3.1 Overview of the function

Sparse matrix-vector multiplication (SpMxV) is crucial function and widely-used in many iterative methods. Its execution time directly affects total execution time in many cases. There are many implementations to perform SpMxV. The best implementation depends on computer environment and numerical characteristics of input sparse matrix. It is hence difficult to fix the best method. We need auto-tuning method at run-time to adapt user's computer environment and matrices.

OpenATI_DSRMV is designed for double symmetric SpMxV, and OpenATI_DURMV is designed for double non-symmetric SpMxV auto-tuning APIs for their implementations at run-time.

2.3.2 Overview of auto-tuning method

In this function, the API surveys all candidates of SpMxV implementations in the first iteration time, then select the best implementation after that. This method was proposed by [2].

The following several implementations are supplied for OpenATI_DSRMV(3 kinds) and OpenATI_DURMV(4 kinds) in version beta.

- OpenATI_DSRMV
  S1) Row Decomposition Method.
  S2) Normalized NZ Method.
  S3) Normalized NZ Method, with vector reduction parallelization.

- OpenATI_DURMV
  U1) Row Decomposition Method.
  U2) Normalized NZ Method (for scalar multi-core processors).
  U3) BSS (Branchless Segmented Scan) (for scalar multi-core processors).
  U4) Original Segmented Scan (for vector processors).

[Row Decomposition Method and Normalized NZ Method]

- Row Decomposition Method

  Input Matrix is divided into the number of threads blocks for balancing the number of row processed by each thread.

- Normalized NZ Method

  Input Matrix is divided into the number of threads blocks for normalizing the number of non-zero element processed by each thread.

Figure 2-3 shows an example of Row Decomposition Method and Normalized NZ Method in case of 6 dimension matrix processed by 4 threads.

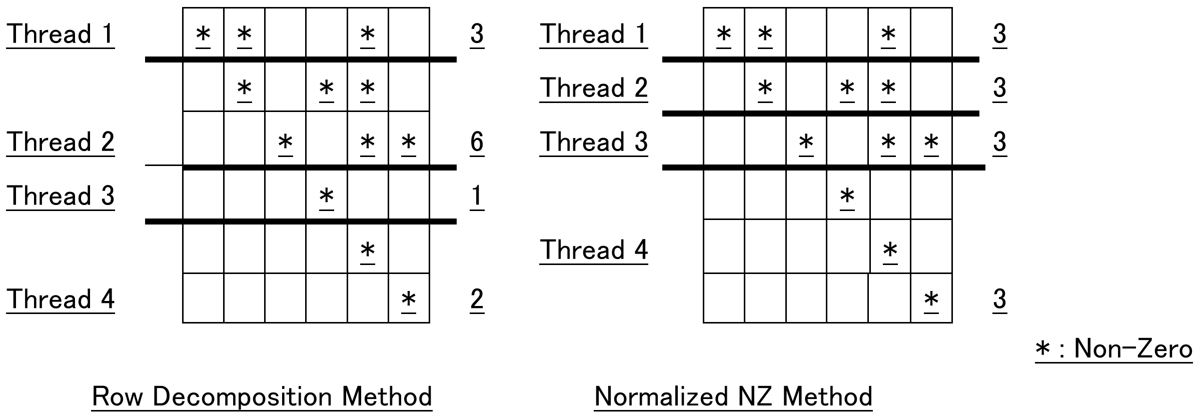

Fig 2-3　An example of Row Decomposition Method and Normalized NZ Method

[Original Segmented Scan method and BSS method]

Original Segmented Scan[5] is designed for sparse matrix multiplication on vector multiprocessors. In this method, input matrix is divided into fixed length of Non-Zero element group. These Non-Zero element group are named segment-vector, In a code of Original Segmented Scan, innermost loop has fixed length of loop and mask process with FLAG representing the beginning of row. (Fig 2-4 shows an example of segment-vector of length 6 processed by 5 threads).

BSS is the method modified for scalar multi-core system by removing IF operator for mask process in innermost loop. In this method, row pointer array in CSR format is extended for segment-vector (In Fig2-4, IRP is expanded　MFLAG）.

【example】

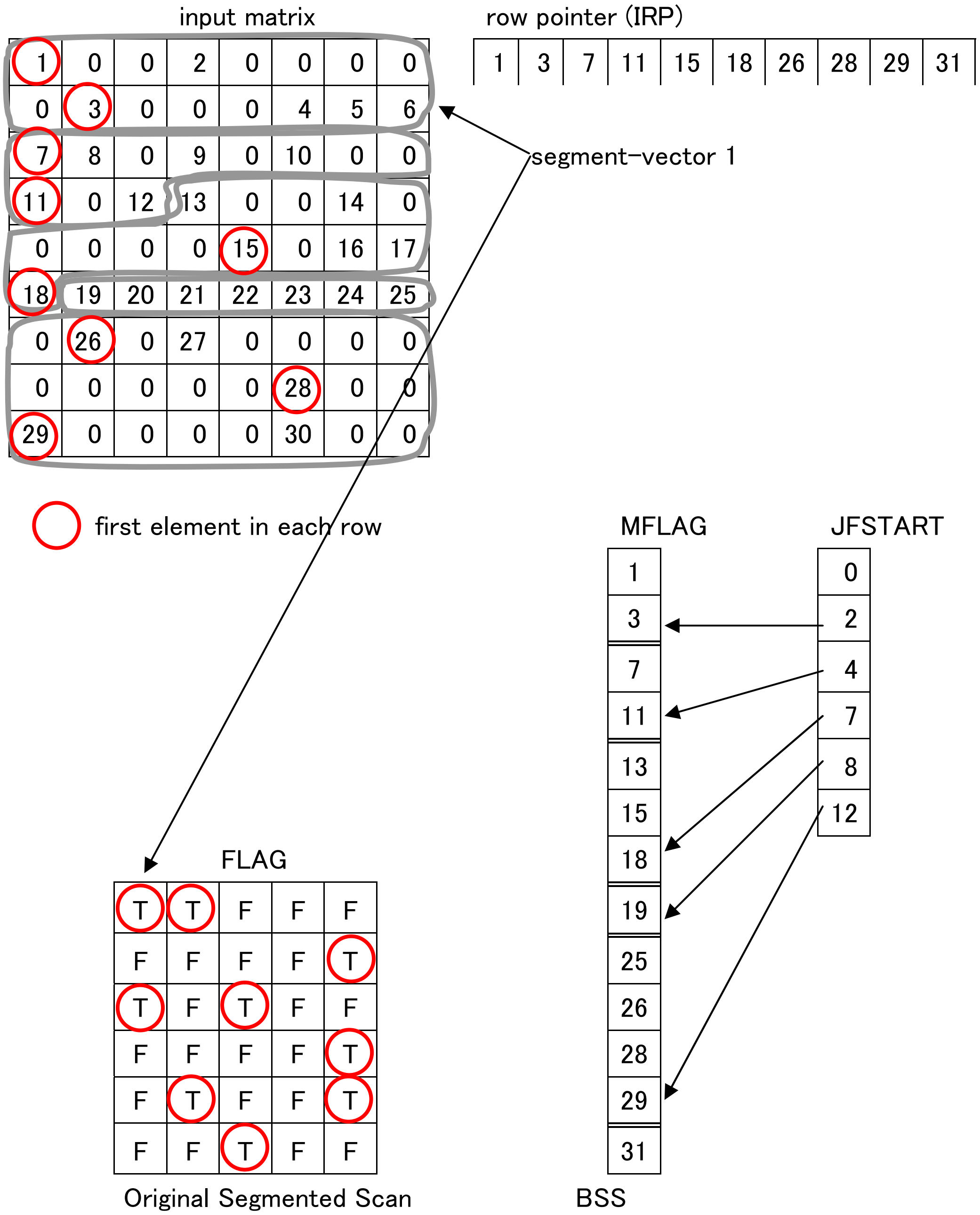

Fig 2-4 An example of Original Segmented Scan and BSS.

If you want to specify SpMxV implementation of OpenATI_DSRMV or OpenATI_DURMV, you need to run setup function before call OpenATI_DSRMV or OpenATI_DURMV.

OpenATI_DSRMV_Setup

(S1) No necessary to run setup function.

(S2) Fix the groups of rows processed by each thread for normalized non-zero elements.

(S3) Fix the groups of rows processed by each thread for normalized non-zero elements, and the start and end point of reduction part of each thread.

OpenATI_DURMV_Setup

(U1) No necessary to run setup function.

(U2) Fix the groups of rows processed by each thread for normalize non-zero elements.

(U3) Set array of MFLAG and JFSTART for BSS.

(U4) Set array of FALG for Original Segmented Scan

2.3.3 Argument Details and Error Code of OpenATI_DSRMV_Setup

(1) Argument Details

Argument	Type	IO	Description
N	Integer	INPUT	The number of dimension for the matrix. (N>=1)
NNZ	Integer	INPUT	The number of non-zero elements for the matrix.
IRP(N+1)	Integer	INPUT	Pointers to first elements on each row for the matrix.
ICOL(NNZ)	Integer	INPUT	The non-zero row indexes for the matrix.
ICASE	Integer	INPUT	Set the number corresponding implementation of SpMxV in OpenATI_DSRMV. 11: No necessary to run this function. 12: Create information for Normalized NZ Method. 13: Create information for Normalized NZ Method with vector reduction parallelization
SINF (LSINF)	Double	OUTPUT	If ICASE=11 No returns. If ICASE=12,13 Returns the groups of rows processed each thread for OpenATI_DSRMV.
LSINF	Integer	INPUT	The size of SINF ICASE=11: LSINF >= 0 ICASE=12: LSINF >= int(0.5*NUM_SMP)+1 ICASE=13: LSINF >= N+NUM_SMP+3
NUM_SMP	Integer	INPUT	Set the number of threads to the argument.
INFO	Integer	OUTPUT	Error Code

(2)Error Code

Value	Description
0	Successful exit.
100	Invalid ICASE value is inputted.

200	Invalid LSINF value is inputted. (ICASE=12 or 13)

2.3.4 Argument Details and Error Code of OpenATI_DURMV_Setup

(1) Argument Details

Argument	Type	IO	Description
N	Integer	INPUT	The number of dimension for the matrix. (N>=1)
NNZ	Integer	INPUT	The number of non-zero elements for the matrix.
IRP(N+1)	Integer	INPUT	Pointers to first elements on each row for the matrix.
ICASE	Integer	INPUT	Set the number corresponding implementation of SpMxV in OpenATI_DURMV. 11: No necessary to run this function. 12: Create information for Normalized NZ Method. 13: Create information for BSS. 21: Create information for Original Segmented Scan
UINF (LUINF)	Double	OUTPUT	ICASE=11: No returns. ICASE=12,13,21: Returns the groups of rows processed each thread or information array for segmented scan.
LUINF	Integer	INPUT	The size of UINF ICASE=11: LUINF >= 0 ICASE=12: LUINF >= int(0.5*NUM_SMP)+1 ICASE=13: LUINF >= int(1.5*N)+546 ICASE=21: LUINF >= int(1.125*NNZ)+273
NUM_SMP	Integer	INPUT	Set the number of threads to the argument.
INFO	Integer	OUTPUT	Error Code

(2)Error Code

Value	Description
0	Successful exit.
100	Invalid ICASE value.
200	LUINF value exceeds upper limit of Integer.
300	Invalid LUINF value (ICASE=12,13,21).

2.3.5 Argument Details and Error Code for OpenATI_DSRMV

(1) Argument Details

Argument	Type	IO	Description
N	Integer	INPUT	The number of dimension for the matrix.（N>=1）
NNZ	Integer	INPUT	The number of non-zero elements for the matrix.
IRP(N+1)	Integer	INPUT	Pointers to diagonal elements on each row for the matrix.
ICOL(NNZ)	Integer	INPUT	The non-zero row indexes for the matrix.
VAL(NNZ)	Double	INPUT	The non-zero elements for the matrix.
X(N)	Double	INPUT	Right hand side vector elements.
Y(N)	Double	OUTPUT	Solution vector elements for SpMxV.
ICASE	Integer	INPUT/ OUTPUT	If OpenATI_DSRMV_IPARM_1=1, then set the number of implementations. If OpenATI_DSRMV_IPARM_1=2 or 3, the best number of implementations returns. The numbers of implementations are: 11: Row Decomposition Method. 12: Normalized NZ Method. 13: Normalized NZ Method, with vector reduction parallelization.
NUM_SMP	Integer	INPUT	If OpenATI_DSRMV_IPARM_1=1 and ICASE=13, or OpenATI_DSRMV_IPARM_1=3, then set the number of threads to the argument.
WK(N, NUM_SMP)	Double	WORK	If OpenATI_DSRMV_IPARM_1=1 and ICASE=13, or OpenATI_DSRMV_IPARM_1=3, then set workspace to the argument.
SINF (LSINF)	Double	INPUT/ OUTPUT	If OpenATI_DSRMV_IPARM_1=1 (INPUT) ICASE=11 : Not necessary to set. ICASE=12,13 : Set SINF retuned by OpenATI_DSRMV_Setup. If OpenATI_DSRMV_IPARM_1=2,3 (INPUT) Not necessary to set.

			(OUTPUT) Returns setup information for best implementation.
LSINF	Integer	INPUT	The size of SINF If OpenATI_DSRMV_IPARM_1=1 ICASE=11: LSINF >= 0 ICASE=12: LSINF >= int(0.5*NUM_SMP)+1 ICASE=13: LSINF >= N+NUM_DMP+3 If OpenATI_DSRMV_IPARM_1=2 LSINF >= int(0.5*NUM_SMP)+1 If OpenATI_DSRMV_IPARM_1=3 LSINF >= N+NUM_SMP+3
INFO	Integer	OUTPUT	Error code.

(2) Global Variables Defined On "OpenAT.inc"

Variable Name	Type	Initial Value	Description
OpenATI_DSRMV_IPARM_1	Integer	1	1 : Perform SpMxV specified by ICASE. 2 : Perform SpMxV to judge the best methods between two methods, except for reduction parallel implementation. 3 : Perform SpMxV to judge the best method among three methods. Note that workspace according to the number of threads is needed.

(3) Error Code

Value	Description
0	Successful exit.
100	The value of ICASE is illegal. (If OpenATI_DSRMV_IPARM_1=1.)
200	The value of OpenATI_DSRMV_IPARM_1 is illegal.

2.3.6 Argument Details and Error Code for OpenATI_DURMV

(1) Argument Details

Argument	Type	IO	Description
N	Integer	INPUT	The number of dimension for the matrix. (N>=1)
NNZ	Integer	INPUT	The number of non-zero elements for the matrix.
IRP(N+1)	Integer	INPUT	Pointers to first elements on each row for the matrix.
ICOL(NNZ)	Integer	INPUT	The non-zero row indexes for the matrix.
VAL(NNZ)	Double	INPUT	The non-zero elements for the matrix.
X(N)	Double	INPUT	Right hand side vector elements.
Y(N)	Double	OUTPUT	Results vector elements for SpMxV.
ICASE	Integer	INPUT/ OUTPUT	If OpenATI_DURMV_IPARM_1=1, then set the number of implementations. If OpenATI_DURMV_IPARM_1=2 or 3, the best number of implementations returns. The numbers of implementations are: 11: Row Decomposition Method. 12: Normalized NZ Method (for scalar multi-core processors). 13: BSS (for scalar multi-core processors). 21: Original Segmented Scan (for vector processors).
UINF (LUINF)	Double	INPUT/ OUTPUT	If OpenATI_DURMV_IPARM_1=1 (INPUT) ICASE=11 : Not necessary to set ICASE=12,13,21 :Set UINF returned by OpenATI_DURMV_Setup. If OpenATI_DURMV_IPARM_1=2,3 (INPUT) Not necessary to set. (OUTPUT) Returns setup information for best implementation.
LUINF	Integer	INPUT	The size of UINF

			If OpenATI_DURMV_IPARM_1=1 ICASE=11: LUINF >= 0 ICASE=12: LUINF >= int(0.5*NUM_SMP)+1 ICASE=13: LUINF >= int(1.5*N)+546 ICASE=21: LUINF >= int(1.125*NNZ)+273 If OpenATI_DURMV_IPARM_1=2. LUINF >= int(0.5*NUM_SMP)+1 If OpenATI_DURMV_IPARM_1=3, LUINF >= int(1.5*N)+546
NUM_SMP	Integer	INPUT	Set the number of threads to the argument.
INFO	Integer	OUTPUT	Error Code.

(2) Global Variables Defined on "OpenAT.inc".

Variable Name	Type	Initial Value	Description
OpenATI_DURMV_IPARM_1	Integer	1	1 : Perform SpMxV specified by ICASE. 2 and 3 : Perform SpMxV to judge the best method among three implementations.
OpenATI_DURMV_IPARM_2	Integer	1	The number of iterations for non-symmetric SpMxV in performance evaluation.

(3)Error Code

Value	Description
0	Successful exit.
100	The value of ICASE is illegal. (If OpenATI_DURMV_IPARM_1=1.)
200	The value of OpenATI_DURMV_IPARM_1 is illegal.

2.3.5 Usage Example

Search the best implementation of SpMxV in the first iteration time, then the best implementation is used after that based on the run-time searching. To implement this, see the code of Fig. 2-5.

```
//Parameter definition.
INCLUDE "OpenAT.inc"              // Include OpenAT.inc
OpenATI_DSRMV_IPARM_1=3           //Initialize DSRMV parameter.
ICASE=0                           //Initialize DSRMV parameter.
LSINF= N+NUM_SMP+3
ALLOCATE(SINF(LSINF))
                                  ～ omission ～

//The first SpMxV.
CALL   OpenATI_DSRMV (N, NNZ, IRP, ICOL, VAL, X, Y, ICASE,
                          NUM_SMP, WK, SINF, LSINF, INFO)
OpenATI_DSRMV_IPARM_1=1   //Hereafter, we select the best one.

                                  ～ omission ～

// SpMxV after run-time searching.
// We can use the best implantation based on previous information.
CALL   OpenATI_DSRMV (N, NNZ, IRP, ICOL, VAL, X, Y, ICASE,
                          NUM_SMP, WK, SINF, LSINF, INFO)

                                  ～ omission ～
```

Fig. 2-5 An Example of OpenATI_DSRMV Description.

If you want to specify SpMxV implementation in OpenATI_DSRMV, implement the code like Fig.2-6.

```
// Parameter definition.
INCLUDE "OpenAT.inc"              // Include OpenAT.inc
OpenATI_DSRMV_IPARM_1=1           // Initialize DSRMV parameter.
ICASE= 13                         // Initialize DSRMV parameter.

                        ~ omission ~

// The first SpMxV.
LSINF=N+NUM_SMP+3                //Allocate memory for setup
ALLOCATE(SINF(LSINF))
CALL   OpenATI_DSRMV_Setup(N,NNZ,IRP,ICOL,ICASE,
                          SINF, LSINF, NUM_SMP,INFO)
CALL   OpenATI_ DSRMV (N,NNZ,IRP,ICOL,VAL,X,Y,ICASE,
                          NUM_SMP, WK, SINF, LSINF, INFO)
                        ~ omission ~

// SpMxV after run-time searching.
// We can use the best implantation based on previous information.
CALL  OpenATI_ DSRMV (N,NNZ,IRP,ICOL,VAL,VEC,JPARM,
                      IPARM,RPARM,SINF,LSINF,INFO)

                        ~ omission ~
```

Fig.2-6 An example of OpenATI_DSRMV Description with specified SpMxV implementation.

2.4 OpenATI_ DAFGS

2.4.1 Overview of the function

Vector Reorthonormalization spends a lot of CPU time in many Krylov Subspace methods. Gram-Schmidt Reorthonormalization method is typcal Reorthonormalization method. There are many implementations to perform Gram-Schmidt method, and trade-offs must be made between computational complexity and accracy. Hence, It is difficult to fix the best implementation.

OpenATI_DAFGS is API that supplies selectable from 4 kinds Gram-Schmidt Reorthonormalization implementation.

2.4.2 Overview of Reorthonormalization method

In this function, the API has 4 kinds Gram-Schmidt Reorthonormalization method. Selected method is indicated by value of Global Variables 'OpenATI_DAFGS_IPARM_1'. By default , Modified Gram-Schmidt method is selected.

(1) Classical Gram-Schmidt method (CGS)

When Krylov Subspace size is large, accuracy of reorthonormalization is lowering. Acceleration performance by parallelization is excellent.

(2) DGKS method

This method supplies improved accuracy by running CGS 2 times. DGKS method computational complexity needs twice as many as CGS' one.

(3) Modified Gram-Schmidt method (MGS)

MGS is most popular Gram-Schmidt method. This method is most effective performance and accuracy.

(4) Blocked Classical Gram-Schmidt method (BCGS)

BCGS method is orthonormalized by intra-block with CGS, by inter-block with MGS. Block length is 4.

2.4.3 Argument Details and Error Code

(1) Argument Details

Argument	Type	IO	Description
NORMALFLG	Integer	INPUT	Normalization of Output vector 0 : not normalized 1 : normalized
N	Integer	INPUT	Vector length（N>=1）
X(N)	Double	INPUT	Vector for normalization
Q(LQ,MM)	Double	INPUT	Orthonormalized vectors Q(1:N,MM)
LQ	Integer	INPUT	Leading Dimension of Q
MM	Integer	INPUT	The number of vector of Q
HR(MM)	Double	OUTPUT	Inner product X by Q(1:N,M)
IDGKS	Integer	OUTPUT	Iterative refinement of DGKS 0 : no Iterative refinement 1 : Iterative refinement

(2) Global Variables Defined on "OpenAT.inc"

Variable Name	Type	Initial Value	Description
OpenATI_DAFGS_IPARM_1	Integer	2	0 : Classical Gram-Schmidt 1 : DGKS 2 : Modified Gram-Schmidt 3 : Blocked Gram-Schmidt

2.5 OpenATI_LINEARSOLVE and OpenATI_EIGENSOLVE : Sparse iterative solvers with Numerical policy

2.5.1 Overview of the function

Numerical policy is requirement and priority of memory, CPU time, accuracy and others specified by library user. OpenATI supplies OpenATI_LINEARSOLVE is designed for unsymmetric liner problem, and OpenATI_EIGENSOLVE is designed for symmetric eigenvalue problem as sparse iterative solvers with numerical policy.

OpenATI_LINEARSOLVE and OpenATI_EIGENSOLVE are Over-Solvers that call Xabclib and set optimized arguments automatically on user's numerical policy.

2.5.2 Overview of numerical policy

If you want to use Over-Solvers, you make numerical policy file with following format, and input numerical policy file path into global variable "OPENATI_POLICY".

Policy file's format is as follow.

```
<keyword> = <value>
```

There is `POLICY/CPU/RESIDUAL/MAXMEMORY/MAXTIME/PRECONDITIONER` as configurable keywords. Unregistered `<keyword>` in policy file is inputted the default value. The explanation of all `<keyword>` is as follow.

```
POLICY = <value>
  <value> : TIME / ACCURACY / MEMORY
  "TIME" is selected by default.
  1. If POLICY = TIME, Over-Solvers preference for execution time over
     accuracy and saving memory. Therefore, algorithms for high
     performance are positively selected.
  2. If POLICY = ACCURACY, Over-Solvers recalculation solution of
     solvers. If false convergence occurs, Over-Solvers continue to
     re-execute with more exact convergence test until true
     convergence.
  3. If POLICY = MEMORY, Over-Solvers set arguments with less memory
     usage.
```

```
CPU = <value>
```

<value> : entry OMP_NUM_THREADS at run-time.
OMP_GET_NUM_THREADS is selected by default.
Note) 1 <= <value> <= OMP_GET_MAX_THREADS()

RESIDUAL = <value>

<value> : entry require accuracy by real value.
The default value is 1.0D-8.
In case of "POLICY = ACCURACY" is set and false convergence occur, solver continue to re-execute with more exact convergence test until true convergence.

MAXMEMORY = <value>

<value> : entry require memory usage in [Gbyte].
The default value is "memfree" in /proc/meminfo (Linux).
If fails to get property in /proc/meminfo, search and allocate free memory dynamically.
Note) The maximum limit of MAXMEMORY is 16Gbyte.

MAXTIME = <value>

<value> : entry time tolerance in [sec].
The default value is infinite.
When execution time exceeds time tolerance, computation is stopped.

PRECONDITIONER = <value>

<value> : NO / JACOBI / SSOR / ILU0
ILU0 is selected by default. This keyword is used by only OpenATI_LINEARSOLVE.

1. PRECONDITIONER = NO : No preconditioner
2. PRECONDITIONER = JACOBI :JACOBI
3. PRECONDITIONER = SSOR :SSOR
4. PRECONDITIONER = ILU0 :ILU(0)

2.5.3 Argument Details and Error Code of OpenATI_LINEARSOLVE

```
CALL OpenATI_LINEARSOLVE (N,NNZ,IRP,ICOL,VAL,B,X,INFO)
```

(1) Argument Details

Argument	Type	IO	Description
N	Integer	INPUT	The number of dimension for the matrix. (N>=1)
NNZ	Integer	INPUT	The number of non-zero elements for the matrix.
IRP(N+1)	Integer	INPUT	Pointes to first position on each row for the matrix. Note: Satisfy IRP(1)=1, IRP(N+1)=NNZ+1.
ICOL(NNZ)	Integer	INPUT	The row indexes for non-zero elements for the matrix.
VAL(NNZ)	Double	INPUT	The non-zero elements for the matrix.
B(N)	Double	INPUT	The elements for right hand size vector *b*.
X(N)	Double	INPUT / OUTPUT	INPUT: Set the elements of initial guess for solution vector *x_0*. OUTPUT: Return the elements of solution vector *x*.
INFO	Integer	OUTPUT	Error Code

(2) Error Code

Value	Description
0	Normal return.
-100	”=” in POLICY FILE is illegal.
-200	The value of OpenATI_DURMV_IPARM_1 is illegal
-300	”POLICY” in POLICY FILE is illegal
-310	”PRECONDITIONER” in POLICY FILE is illegal
-400	The value of ”MAXMEMORY” in POLICY FILE is greater than free size of memory
-500	Failing to allocate work area
>0	Error code from Xabclib_GMRES. For more detaile, refer 3.1.3.

2.5.4 Argument Details and Error Code of OpenATI_EIGENSOLVE

```
CALL OpenATI_EIGENSOLVE(N,NNZ,IRP,ICOL,VAL,IORDER, NEV,EV,EVEC,INFO)
```

(1) Argument Details

Argument	Type	IO	Description
N	Integer	INPUT	The number of dimension for the matrix. （N>=1）
NNZ	Integer	INPUT	The number of non-zero elements for the upper triangle part.
IRP(N+1)	Integer	INPUT	Pointes to diagonal elements on each row. Note: Satisfy IRP(1)=1, IRP(N+1)=NNZ+1.
ICOL(NNZ)	Integer	INPUT	The row indexes for non-zero elements on the upper triangle part.
VAL(NNZ)	Double	INPUT	The values for non-zero elements on the upper triangle part.
IORDER	Integer	INPUT	Option parameter for eigensolve 1 : Compute eigenvalues and eigenvectors from the raw value, that means including minus. 2 : Compute eigenvalues and eigenvectors from the absolute value
NEV	Integer	INPUT	The number of eigenvalues you need.
EV(NEV)	Double	OUTPUT	The eigenvalues. The k-th eigenvalue is set to EV(k).
EVEC (N,NEV)	Double	OUTPUT	The eigenvectors. The k-the eigenvector corresponding to the eigenvalue EV(k) is set to the k-th column.
INFO	Integer	OUTPUT	Error Code

(2) Error Code

Value	Description
0	Normal return.
-100	”=” in POLICY FILE is illegal.
-200	The value of OpenATI_DURMV_IPARM_1 is illegal
-300	”POLICY” in POLICY FILE is illegal
-310	”PRECONDITIONER” in POLICY FILE is illegal
-400	The value of ”MAXMEMORY” in POLICY FILE is greater than free size of

	memory
-500	Failing to allocate work area
>0	Error code from Xabclib_LANCZOS. For more detail, refer 3.1.4.

2.5.5 Usage Example

An example of policy file

```
POLICY          =      ACCURACY
RESIDUAL        =      1.0D-10
CPU             =      16
PRECONDITIONER  =      ILU0
MAXMEMORY =1.0
MAXTIME   =      500.0
```

Before running, set global variables "OPENATI_POLICY" as follow.
(In case of file name is "input_policy.data")

```
OPENATI_POLICY   =    input_policy.data
```

When OpenATI_LINEARSOLVE running is complete, computation result and input parameters are recorded in "OPENATI_POLICY_REPORT.txt".

An example of "OPENATI_POLICY_REPORT.txt" as follow.

```
****************************************************
*****  OpenATI  LINEAR SOLVER POLICY REPORT    *****
*****                  2010.0114   11:30    *****      <- report date / time
****************************************************

 [Environment variables]                                | input parameters
    OPENATI_DEBUG  =                                     v
    OPENATI_POLICY = ./input_policy.dat
 [Policy Definitions]
   POLICY         = ACCURACY
   SMPs           =           16
   SOLVER         = XABCLIB_GMRES
   PRECONDITIONER = ILU0
   REQUIREMENT WORKING MEMORY =    16.0000000000000
     <<< Upper Bound 16GBYTE >>>
   REQUIREMENT RESIDUAL      =   1.000000000000000E-008
   REQUIREMENT MAX. TIME     =    500.000000000000

   MAX. SUBSPACE SIZE   =       14214
   RUNTIME MEMORY USE   =      3.24 [GBYTE]

   KRYLOV SUBSPACE EXPAND AT =  1   ,MATVEC AT =  1
   Initial Gram-Schmidt Strategy = BCGS

 ====== OPENATI_LINEARSOLVE SUCCESSFULY ENDED ======       | successfully exit
                                                           v
 [OPENATI_LINEARSOLVE RESULT]                              |result report
   MATRIX DATA : N=       14214  NNZ=       259688         v
   FASTEST MATVEC NO. =           11                       <- fastest OpenATI_DURMV case
   FINAL KRYLOV SUBSPACE SIZE =          42                <- Msize for convergence
   FINAL Gram-Schmidt Strategy = DGKS
   2-Norm of RHS =   25.2388589282479                      <- initial norm of RHS
   NUMBER OF RETRYED GMRES =           6                   <- retried iterations
   TOTAL RESTARTS of GMRES =         197
   RESIDUAL NORM          =  3.005885687924543E-010
   SET-UP TIME            =  1.126790046691895E-002  [SEC]
   SOLVER TIME            =   1.32032704353333       [SEC]

   TOTAL  TIME            =   1.33159494400024       [SEC]
```

3. Xabclib : A Numerical Library with Auto-tuning Facility on OpenATLib

3.1 Xabclib_LANCZOS

3.1.1 Overview of the function

Xabclib_LANCZOS can compute several eigenvalues from the absolutely largest value for large-scale symmetric matrices in the standard eigenproblem.

3.1.2 Target problem formularization and data format

(1) Target problem

The target problem is the standard eigenproblem $A\ v = \lambda\ v$ for computing eigenvalues and eigenvectors on large-scale sparse matrices, where A is a large-scale sparse matrix, λ is an eigenvalue, and v is an eigenvector.

(2) Input data format

The data format for input symmetric sparse matrix A is Compressed Row Storage (CRS) shown in Fig.3-1. Please note that the format is dedicated for symmetric matrices, hence we do not need lower elements.

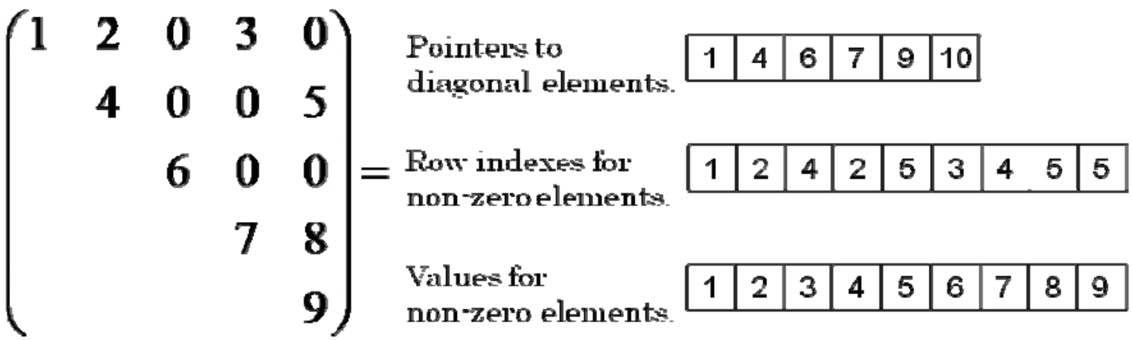

Fig. 3-1 Compressed Row Storage (CRS) for Symmetric Matrices.

3.1.3 The Lanczos Method

The Lanczos method using this library is shown in Fig. 3-2. The algorithm is based on the algorithm referred by [3].

1. $Start\ with\ v_0 \equiv r, \beta_0 := \|r\|_2, lock = 0$
2. $For\ IR = 1,2,\cdots, maxrestart\ Do:$
3. $\quad For\ j = lock+1,\cdots,m\ Do:$
4. $\qquad Compute\ v_j := r/\beta_0$
5. $\qquad r{:=}Av_j$
6. $\qquad \alpha_{\rm j} := (r, v_j)$
7. $\qquad if\ \ (j=1)\ then \quad r{:=}r - \alpha_{\rm j} v_j$
8. $\qquad if\ \ (j \neq 1)\ then \quad r{:=}r - \alpha_{\rm j} v_j - \beta_{\rm j\text{-}1} v_{j-1}$
9. $\qquad r \perp V_{j-1}\ \ by\ modified\ Gram\text{-}Schmidt$
10. $\qquad \beta_j := \|r\|_2$
11. $\quad EndDo$
12. $\quad Eigen\ solve\ T = S\Theta S^T\ \ ,\ T = \begin{bmatrix} \alpha_{\rm lock+1} & & & & \\ \beta_{lock+1} & \alpha_{\rm lock+2} & & & \\ & \cdots & \cdots & & \\ & & \cdots & \cdots & \\ & & & \beta_{m-1} & \alpha_m \end{bmatrix}$
13. $\quad k\text{-}th\ residual\ estimate\ with\ \left|\beta_m S_{m,k}\right| / \left|\Theta_k\right|\ for\ k = lock+1, NEV$
14. $\quad creat\ Ritz\ vectors\ {\rm Q_k} = V_m S_k$
15. $\quad count-up\ \ 'new\ locked'\ Ritz\ pair$
16. $\quad if\ \ (lock + 'new\ lock'\ \ \geq NEV)\ \ goto\ exit$
17. $\quad create\ new\ starting\ Shur\ vector\ r = V_m S_{'new\ locked'+1}$
18. $\quad deflation\ V_{lock+L} = Q_{lock+L}\ \ for\ L = 1, 'new\ lock', then\ lock = 'new\ lock'$
19. $EndDo$

Fig. 3-2 The Lanczos Method.

3.1.4 Argument Details and Error Code

(1) Argument Details

Argument	Type	IO	Description
N	Integer	INPUT	The number of dimension for the matrix. (N>=1)
NNZ	Integer	INPUT	The number of non-zero elements for the upper triangle part.
IRP(N+1)	Integer	INPUT	Pointes to diagonal elements on each row. Note: Satisfy IRP(1)=1, IRP(N+1)=NNZ+1.
ICOL(NNZ)	Integer	INPUT	The row indexes for non-zero elements on the upper triangle part.
VAL(NNZ)	Double	INPUT	The values for non-zero elements on the upper triangle part.
NEV	Integer	INPUT	The number of eigenvalues you need. The execution time increases according to the NEV. If NEV>100, the execution time will be enormous, hence it may not solve in practical time.
EV(NEV)	Double	OUTPUT	The eigenvalues. The k-th eigenvalue is set to EV(k).
EVEC (LDE,NEV)	Double	OUTPUT	The eigenvectors. The k-the eigenvector corresponding to the eigenvalue EV(k) is set to the k-th column.
LDE	Integer	INPUT	The dimension of EVEC array (LDE < N)
MSIZE	Integer	INPUT	The restart frequency. Set MSIZE > NEV.
IPARM(10)	Integer	INPUT / OUTPUT	Library patameters for the Lanczos method. (Integer) ● IPARM(1) : INPUT 1 : Compute eigenvalues and eigenbectors from the raw value, that means including minus. 2 : Compute eigenvalues and eigenbectors from the absolute value. ● IPARM(2) : INPUT Set maximum restart frequency for Lanczos method. ● IPARM(3) : OUTPUT Returns the actual restart frequency. ● IPARM(4) If IAT(1)=1, (INPUT) Set initial restart frequency. If IPARM(4) < NEV,

			then IPRAM(4)=NEV. OUTPUT) Returns the actual restart frequency. ・IPARM(5) -- IPARM(10) For future extension.
RPARM(10)	Double	INPUT/ OUTPUT	Library patameters for the Lanczos method. (Double) ● RPARM(1):INPUT Set the convergence test value for eigenvalue and eigenvector computation. The test norm in this solver is as follows: $$\frac{\|Ax-\lambda x\|}{\|\lambda\|}.$$ ● RPARM(2) (INPUT) Tolerance maximum execution time in second. (OUTPUT) Returns the actual execution time in second. ● RPARM(3) :INPUT The threshold value for MM ratio to judge restart frequency. It is same as OpenATI_DAFRT_RPARM_1 on OpenATI_DAFRT. ● RPARM(4) -- RPARM(10) For future extension.
IAT(10)	Integer	INPUT/ OUTPUT	Auto-tuning control parameters. ● If IAT(1)＝1, the best restart frequency is set by using auto-tuning facility. ● IAT(2) :INPUT 1 : Perform SpMxV with the best method using auto-tuning facility. 2 : Perform SpMxV with taking into account avairable memory space at run-time using auto-tuning facility. -11 : Perform SpMxV with Row Decomposition method. -12 : Perform SpMxV with Normalized NZ method. -13 : Perform SpMxV with Normalized NZ method with parallel vector reduction.

			● IAT(3) :OUTPUT Retuens the number indicating performed SpMvx implementation. ● IAT(4) -- IAT(10) For future extension.
WK (LWK)	Double	WORK	Workspace.
LWK	Integer	INPUT	The size of the double precision workspace WK. Satisfy LWK >= (1+MSIZE)*N + 2*MSIZE*MSIZE + 7*MSIZE + 5*NEV +2.
IWK (LIWK)	Integer	WORK	Workspace.
LIWK	Integer	INPUT	The size of the integer workspace IWK. Satisfy LIWK >= 5*MSIZE + 3.
INFO	Integer	OUTPUT	Error code.

(2) Error Code

Value	Description
0	Normal return.
Less than 0	If -i returns, the value of i-th argument is illegal.
100	Computation was stopped by breakdown for zero vector division.
200	Computation was stopped by abnormal computation of eigenvalues in part of tridiagonal matrix computation.
300	Computation was stopped by exceeding the maximum number of restart.
400	Computation was stopped by exceeding the execution time tolerance.
500	Computation was stopped by failing to allocate memory in case of IAT(2)=-12,-13.

3.2 Xabclib_GMRES

3.2.1 Overview of the function

Xabclib_GMRES can solve large-scale non-symmetric sparse matrices in the linear equations problem.

3.2.2 Target problem and data format

(1) Target problem

The problem to be solved in the library is the linear equations problem $A\ x = b$, where A is a large-scale sparse matrix, x is a solution vector, and b is a right hand side vector.

(2) Input data format

The non-symmetric sparse matrix format is Compressed Row Storage (CRS) for non-symmetric matrices shown in Fig. 3-3.

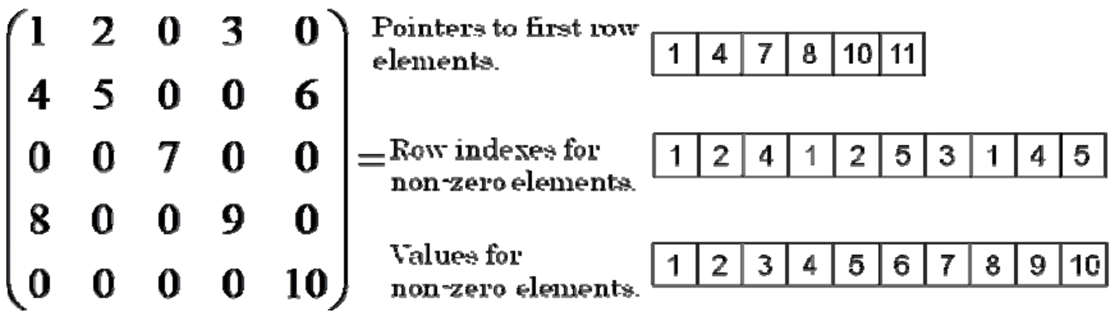

Fig. 3-3 Compressed Row Storage (CRS) for Non-symmetric Matrices.

3.2.3 Overview of the algorithm

The algorithm used in this solver is the GMRES method, which is shown in Fig. 3-4. The algorithm was presented in [4].

1. *Compute* $r_0 = b - Ax_0, \beta := \|r_0\|_2$, *and* $v_1 := r_0 / \beta$
2. *Define the* $(m+1) \times m$ *matrix* $\overline{H}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$, *Set* $\overline{H}_m = 0$
3. *For* $j = 1, 2, \cdots, m$ *Do*:
4. *Compute* $\omega_j := Av_j$
5. *For* $i = 1, \cdots, j$ *Do*:
6. $h_{ij} := (\omega_j, v_j)$
7. $\omega_j := \omega_j - h_{ij} v_j$
8. *EndDo*
9. $h_{j+1,j} = \|\omega_j\|_2$. *If* $h_{j+1,j} = 0$ *Set* $m := j$ *and go to* 12
10. $v_{j+1} = \omega_j / h_{j+1,j}$
11. *EndDo*
12. *Compute* y_m *the minimizer of* $\|\beta e_1 - \overline{H}_m y\|_2$ *and* $x_m = x_0 + V_m y_m$.

Fig. 3-4 The GMRES Method.

3.2.4 Argument Details and Error Code

(1) Argument Details

Argument	Type	IO	Description
N	Integer	INPUT	The number of dimension for the matrix. （N>=1）
NNZ	Integer	INPUT	The number of non-zero elements for the matrix.
IRP(N+1)	Integer	INPUT	Pointes to first position on each row for the matrix. Note: Satisfy IRP(1)=1, IRP(N+1)=NNZ+1.
ICOL(NNZ)	Integer	INPUT	The row indexes for non-zero elements for the matrix.
VAL(NNZ)	Double	INPUT	The non-zero elements for the matrix.
B(N)	Double	INPUT	The elements for right hand size vector *b*.
X(N)	Double	INPUT / OUTPUT	INPUT: Set the elements of initial guess for solution vector *x_0*. OUTPUT: Return the elements of solution vector *x*.
KIND_PRECOND	Integer	INPUT	Set preconditioner kinds. 1 : None. 2 : Jacobi. 3 : SSOR. 4 : ILU(0).
PRECOND (NPRE)	Double	INPUT / OUTPUT	INPUT: ● If IPCPARM(1)=1, then none to be set. ● If IPCPARM(1)=2, then set preconditioner kind of M already specified. OUTPUT: ● If IPCPARM(1)=1, then the preconditioner kind of M returns. ● If IPCPARM(1)=2, then no modification.
NPRE	Integer	INPUT	The size of PRECOND array. If KIND_PRECOND is 2, then NPRE≧0. If KIND_PRECOND is 3 or 4, then NPRE≧N.
IPCPARM (10)	Integer	INPUT	Preconditioner Parameters（Integer） ● IPCPARM(1)

			1：Compute Preconditioner M. 2：Use precondition M inputed by user. ● IPCPARM(2) -- IPCPARM(10) For future extension.

RPCPARM (10)	Double	INPUT	Preconditioner parameters (Double)
			● RPCPARM(1)
			If KIND_PRECOND=3, then
			Set parameter ω for SSOR preconditioner.
			If KIND_PRECOND=4, then
			Set threathold value to judge breakdown when computing ILU(0) preconditioner.
			● RPCPARM(2) -- RPCPARM(10)
			For future extension.

MSIZE	Integer	INPUT	Restart Frequency.
IGRPARM (10)	Integer	INPUT/ OUTPUT	Library parameters for GMRES Method.（Integer） ● IGRPARM(1)：INPUT Set maximum restart frequency for GMRES method. ● IGRPARM(2)：OUTPUT Final restart frequency returns. ● IGRPARM(3) If IAT(1)=1, (INPUT) Set initial restart frequency. If initial value is not positive, then it's set 2. (OUTPUT) Returnss the actual restart frequency. ● IGRPARM(4) -- IGRPARM(10) For future extension.
RGRPARM (10)	Double	INPUT	Library parameters for GMRES Method.（Double） ● RGRPARM(1) :INPUT Set the threthold value of convergence test. The convergence test is done with the following formula: $\dfrac{\left\Vert M^{-1}(b-Ax)\right\Vert}{\left\Vert M^{-1}b\right\Vert}$. ● RGRPARM(2)

			(INPUT) Set maximum tolerance execution time in second. (OUTPUT) Returns the actual execution time in second. ● RGRPARM(3) :INPUT Set threthold value of MM ratio to judge restart frequency. It is same as OpenATI_DAFRT_RPARM_1 on OpenATI_DAFRT. ● RGPARAM(4) :OUTPUT Returns the residual value. ● RGRPARM(5) -- RGRPARM(10) For future extension.
IAT(10)	Integer	INPUT/ OUTPUT	Auto-tuning parameters. ● If IAT(1)＝1, set the best restart frequency with auto-tuning facility. ● IAT(2) : INPUT 1: set the best implementation of SpMxV with auto-tuning facility. -11 : Perform SpMxV with Row Decomposition Method. -12 : Perform SpMxV with Normalized NZ Method. -13 : Perform SpMxV with BSS. -21 : Perform SpMxV with Original Segmented Scan. ● IAT(3) :OUTPUT Retuens the number indicating performed SpMvx implementation. ● IAT(4) -- IAT(10) For future extension.
WK (LWK)	Double	WORK	Workspace.
LWK	Integer	INPUT	The size of the workspace for double precision WK. Satisfy LWK >= (MSIZE+2)*N + (MSIZE+1)*(MSIZE+1) + (N-1)/2+1.
INFO	Integer	OUTPUT	Error code.

(2) Error Code

Value	Description
0	Normal return.
Less than 0	If -i returns, the value of i-th argument is illegal.
100	Computation was stopped by failing to make preconditioner.
200	Computation was stopped by breakdown.
300	Computation was stopped by that the value of OpenATI_DAFRT is illegal.
400	Computation was stopped by exceeding the execution time tolerance.
500	Computation was stopped by exceeding the maximum number of restart.
600	Computation was stopped by failing to allocate memory in case of IAT(2)=-12,-13.
700	Computation was stopped by the value of LUINF exceeds Integer max in case of ICASE=21.

4. References

[1] T. Sakurai, K. Naono, M. Egi, M. Igai, and H. Kidachi: Proposal on Runtime Parameter Auto Tuning Approach for Restarted Lanczos Method, IPSJ SIG Notes, 2007-HPC-111, pp.173-178, (2007)(in Japanese).

[2] M. Kudo, H. Kuroda, T. Katagiri, and Y. Kanada: The Effect of Optimal Algorithm Selection of Parallel Sparse Matrix-Vector Multiplication, IPSJ SIG Notes, 2002-ARC-147, pp.151-156 (2002)(in Japanese).

[3]V. Hernandez, J. E. Roman, and A. Tomas: Evaluation of Several Variants of Explicitly Restarted Lanczos Eigensolvers and Their Parallel Implementations, High Performance Computing for Computational Science - VECPAR 2006, pp.403-416 (2007).

[4] Y. Saad: Iterative methods for sparse linear systems, SIAM, (1996).

[5]Guy E. Blelloch, Michael A. Heroux, and Marco Zagha: Segmented Operations for Sparse Matrix Computation on Vector Multiprocessors, Carnegie Mellon University, Pittsburgh, PA, (1993).

[6] K. Naono, M. Igai and H. Kidachi: Performance Evaluation of the Gram-Schmidt Orthogonalization Library with Numerical Policy Interface on Heterogeneous Platforms, IPSJ Tran. on Advanced computing systems, 46(SIG_12(ACS_11)) , pp.279-288 (2005)(in Japanese).

[7]Daniel, J., Gragg, W.B., Kaufman, L. And Stewart, G.W.: Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization, Math. of Computation, Vol.30, pp.772-795 (1976).

[8] K. Naono, M. Igai and H. Kidachi: Performance Evaluation of the Gram-Schmidt Orthogonalization Library with Numerical Policy Interface on Heterogeneous Platforms, Transaction on Advanced Computing Systems, Vol. 46 No. SIG12 (ACS11), pp. 279-288 (2005) (in Japanese).