OpenATLib and Xabclib

Developer's Manual for Version Alpha

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

September 3rd, 2009

DISCLAIMER

This software, OpenATLib and Xabclib, is provided by the copyright holders and contributors, Information Technology Center, The University of Tokyo and Central Research Laboratory, Hitachi Ltd., "AS IS" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright owner or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence of otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

1. Overview

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

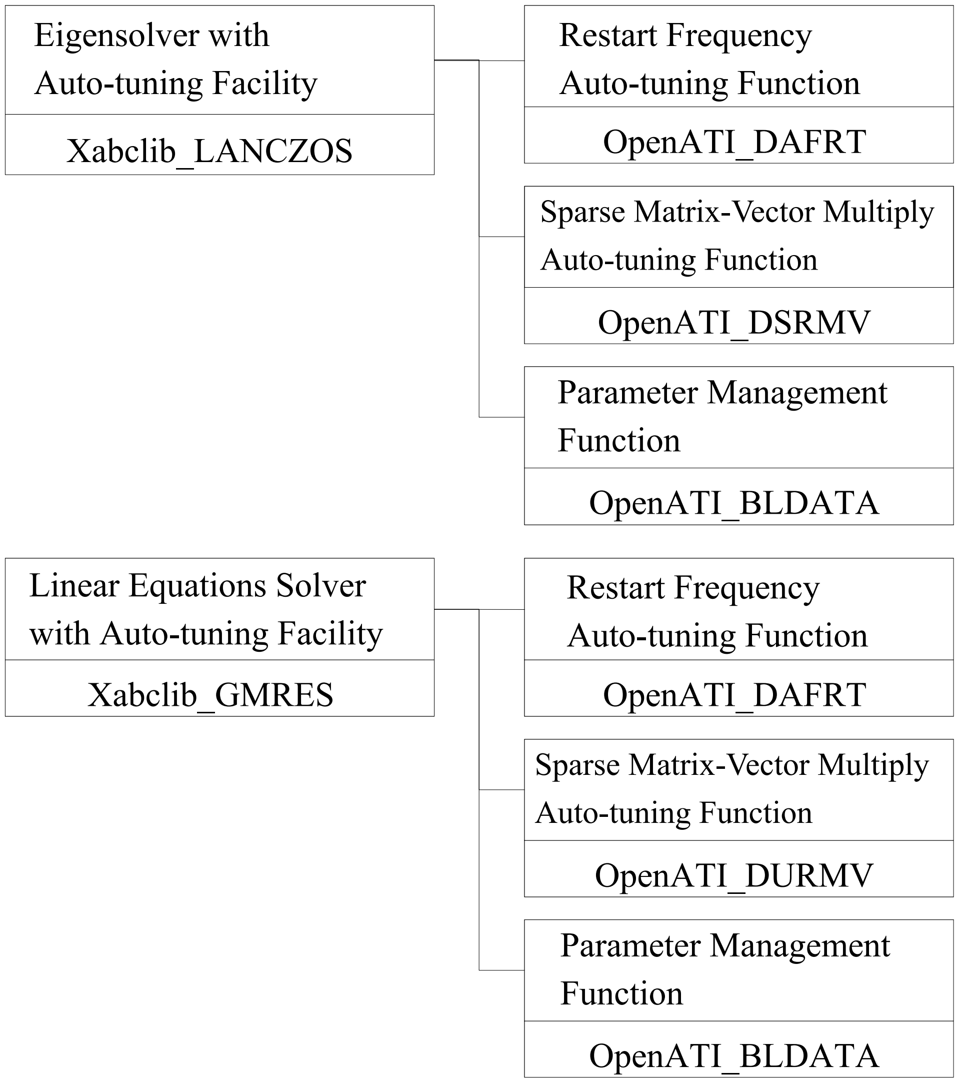

Fig. 1-1 Components of Function on Xabclib.

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_BLDATA	Set default parameters. (Block data format for Fortran.)

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

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

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.

(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:

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

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

Judge incensement of restart frequency 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

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 three kinds of implementation is supplied for OpenATI_DSRMV and OpenATI_DURMV in version alpha.

- OpenATI_DSRMV
  1) Parallelized for the most inner loop, and sequentialized for the outer loop.
  2) Fusion loop for cache optimization, and sequentialized.
  3) Fusion loop for cache optimization, and reduction parallelization (needs workspace allocation at run-time). Each core refers different workspace. Hence, the method requires workspace for: (The number of threads) * (The dimension of vector).

- OpenATI_DURMV
  1) Vectorized loops through compiler optimization.
  2) Explicit 8*2 unrolling description for the outer loop with compiler directive.
  3) Explicit non-vectorized description through compiler optimization.

2.3.3 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: Parallelized for the most inner loop, and sequentialized for the outer loop. 12: Fusion loop for cache optimization, and sequentialized. 13: Fusion loop for cache optimization, and reduction parallelization (needs workspace allocation at run-time). Each core refers different workspace. Hence, the method requires the workspace for (The number of threads) * (The dimension of vector).
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.
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	Normal return.
100	The value of ICASE is illegal. (If OpenATI_DSRMV_IPARM_1=1.)
200	The value of OpenATI_DSRMV_IPARM_1 is illegal.

2.3.4 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: Vectorized loops through compiler optimization. 12: Explicit 8*2 unrolling description for the outer loop with compiler directive. 13: Explicit non-vectorized description through compiler optimization.
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	Normal return.
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. 2-3.

```
//Parameter definition.
INCLUDE "OpenAT.inc"            // Include OpenAT.inc
OpenATI_DSRMV_IPARM_1=3         //Initialize DSRMV parameter.
ICASE=0                         //Initialize DSRMV parameter.

                                ～ omission ～

//The first SpMxV.
CALL  OpenATI_DSRMV (N, NNZ, IRP, ICOL, VAL, X, Y, ICASE,
                             NUM_SMP, WK, 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, VEC, JPARM,
                         IPARM, RPARM, INFO)

                                ～ omission ～
```

Fig. 2-3 An Example of OpenATI_DSRMV Description.

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.

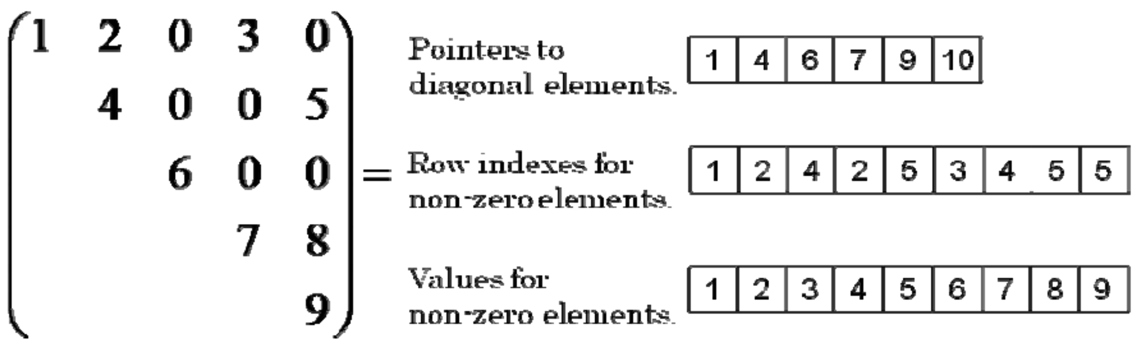

Fig. 3-1 Compressed Row Storage (CRS) for Symmetric Matrices.

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. $\quad\quad Compute\ v_j := r / \beta_0$
5. $\quad\quad r := Av_j$
6. $\quad\quad \alpha_{\mathrm{j}} := (r, v_j)$
7. $\quad\quad if\ \ (j=1)\ then\quad r := r - \alpha_{\mathrm{j}} v_j$
8. $\quad\quad if\ \ (j \neq 1)\ then\quad r := r - \alpha_{\mathrm{j}} v_j - \beta_{\mathrm{j-1}} v_{j-1}$
9. $\quad\quad r \perp V_{j-1}\ \ by\ modified\ Gram\text{-}Schmidt$
10. $\quad\quad \beta_j := \|r\|_2$
11. $\quad EndDo$
12. $\quad Eigen\ solve\ T = S\Theta S^T \ \ , T = \begin{bmatrix} \alpha_{\mathrm{lock+1}} & & & & \\ \beta_{lock+1} & \alpha_{\mathrm{lock+2}} & & & \\ & \cdots & \cdots & & \\ & & \cdots & \cdots & \\ & & & \beta_{m-1} & \alpha_m \end{bmatrix}$
13. $\quad k\text{-}th\ residual\ estimate\ with\ |\beta_m S_{m,k}| / |\Theta_k|\ for\ k = lock+1, NEV$
14. $\quad creat\ Ritz\ vectors\ \mathrm{Q_k} = V_m S_k$
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 Return the actual restart frequency. ・IPARM(4)～IPARM(10) For future extension.
RPARM(10)	Double	INPUT	Library patameters for the Lanczos method. (Double) ● RPARM(1)

			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) Tolerance maximum execution time in second. ● RPARM(3) 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	Auto-tuning control parameters. ● If IAT(1)＝1, the best restart frequency is set by using auto-tuning facility. ● IAT(2) 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. ● IAT(3)～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.

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.

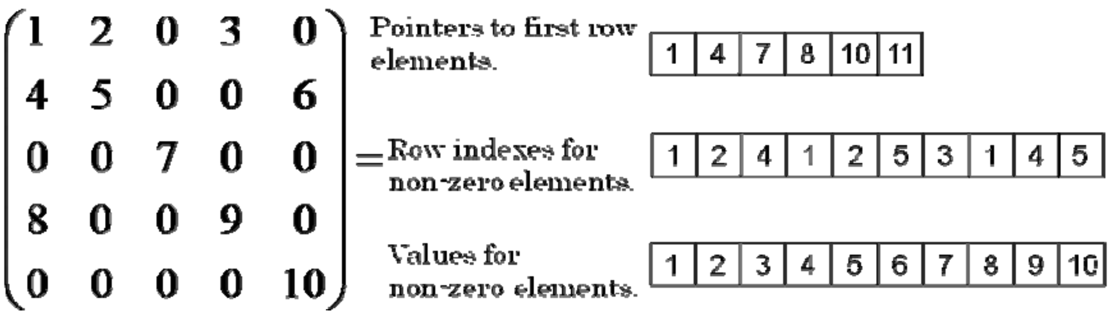

Fig. 3-3 Compressed Row Storage (CRS) for Non-symmetric Matrices.

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. 0 : None. 1 : Jacobi. 2 : SSOR. 3 : ILU(0).
PRECOND (NPRE)	Double	INPUT / OUTPUT	INPUT: ● If IPCPARM(1)=0, then none to be set. ● If IPCPARM(1)=1, then set preconditioner kind of M already specified. OUTPUT: ● If IPCPARM(1)=0, then the preconditioner kind of M returns. ● If IPCPARM(1)=1, then no modification.
NPRE	Integer	INPUT	The size of PRECOND array. If KIND_PRECOND is 1, then NPRE≧0. If KIND_PRECOND is 2 or 3, then NPRE≧N.
IPCPARM (10)	Integer	INPUT	Preconditioner Parameters (Integer) ● IPCPARM(1)

			0：Compute Preconditioner M. 1：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=2, then
			Set parameter ω for SSOR preconditioner.
			If KIND_PRECOND=3, 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)～IGRPARM(10) For future extension.
RGRPARM (10)	Double	INPUT	Library parameters for GMRES Method.（Double） ● RGRPARM(1) Set the threthold value of convergence test. The convergence test is done with the following formula: $\frac{\lVert M^{-1}(b-Ax)\rVert}{\lVert M^{-1}b\rVert}$. ● RGRPARM(2) Set maximum tolerance execution time in second. ● RGRPARM(3) Set threthold value of MM ratio to judge restart frequency. It is same as OpenATI_DAFRT_RPARM_1 on OpenATI_DAFRT. ● RGRPARM(4)～RGRPARM(10) For future extension.

IAT(10)	Integer	INPUT	Auto-tuning parameters. ● If IAT(1) = 1, set the best restart frequency with auto-tuning facility. ● If IAT(2) = 1, set the best implementation of SpMxV with auto-tuning facility. ● IAT(3)～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.

4. References

[1] Takao Sakurai, Ken Naono, Masashi Egi, Mitsuyoshi Igai, and Kidachi Hiroyuki: Proposal on Runtime Parameter Auto Tuning Approach for Restarted Lanczos Method, IPSJ SIG Notes, 2007-HPC-111, pp.173-178, 2007. In Japanese.

[2] Makoto Kudo, Hisayasu Kuroda, Takahiro Katagiri, and Yasumasa 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.