Application of Approximate Minimum Degree Ordering Scheme to Burnup Matrix

Yong Hee Choi[a]*, Jin Young Cho[a]
[a]*Korea Atomic Energy Research Institute, 111, Daedeok-daero 989beon-gil, Yuseong-gu, Daejeon, 34057, Korea*
*Corresponding author: yhchoi@kaeri.re.kr

1. Introduction

Burnup calculation is very important in nuclear core design or radiation shielding design because it provides the material composition changes caused by nuclear reactions or decay. The essential part of the burnup calculation is to obtain the numerical solution of matrix exponential function. Krylov subspace method is one of the most powerful method used in calculating matrix exponential. The key feature of Krylov subspace method is to generate a Hessenburg matrix so that the original large sparse burnup matrix can be approximated by the small but dense matrix having much smaller dimension than original burnup matrix. Once the Hessenburg matrix is found, the matrix exponential can be obtained by using Talyor series. Chebyshev Rational Approximation Method (CRAM) is also widely used in various codes, for instance, SERPENT [1]. In case of CRAM, the matrix exponential is approximated by rational function. Because the rational function always involves the inverse of a matrix, solving Ax=b is essential for CRAM. In this study, a direct solver is chosen as an inverse matrix solver over an iterative solver due to the ill-conditioned nature of the burnup matrix and to take advantage of the sparsity of the burnup matrix. Especially, LU factorization with fill-in reducing ordering scheme is introduced. We will show this direct solver is as efficient as the iterative solver if a proper fill-in reducing scheme is introduced because the burnup matrix is highly sparse.

2. Brief Review of CRAM

Burnup equations make a system of first order differential equations that can be simply expressed by a matrix form as

$$\frac{d\vec{X}(t)}{dt} = \mathbf{A}\cdot\vec{X}(t) \quad , \quad (1)$$

where $\vec{X}(t)$ is nuclide concentration vector and $\mathbf{A}$ is burnup matrix containing the decay and transmutation coefficients of the nuclides under consideration.

The formal solution of the burnup equation can be written by using the matrix exponential function as

$$\vec{X}(t) = e^{\mathrm{A}t}\vec{X}(0) \quad , \quad (2)$$

where $\vec{X}(0)$ represents the nuclide concentration vector at time t=0.

In CRAM, the exponential function can be approximated by rational function with order k as follow.

$$e^{z} \approx r_{k,k}(z) = \alpha_0 + \sum_{j=1}^{k}\frac{\alpha_j}{z-\theta_j} \quad , \quad (3)$$

Because of the fact that the poles always form conjugate pairs, the equation can be rewritten by

$$r_{k,k}(z) = \alpha_0 + 2\,\mathrm{Re}\left(\sum_{j=1}^{k/2}\frac{\alpha_j}{z-\theta_j}\right) \quad , \quad (4)$$

From Eq. (2) and (4), the solution of the burnup equation can be expressed by

$$\vec{X}(t) \approx \alpha_0\vec{X}(0) + 2\,\mathrm{Re}\left(\sum_{j=1}^{k/2}\alpha_j\left(At-\theta_j I\right)^{-1}\right)\vec{X}(0) \quad , \quad (5)$$

As shown in Eq. (5), CRAM repeats solving Ax=b k/2 times in single time step depending on order k. In this study, we introduce a direct solver based on LU factorization with fill-in reducing ordering scheme so called AMD.

3. AMD Ordering Scheme

The AMD method [2] is originally invented as a fill-reducing algorithm of a Cholesky factorization (A=LL^T) because this algorithm assumes that the target matrix is symmetric positive definite. However, it turned out that the AMD is also applicable on an asymmetric case for the purpose of fill-in minimization. Before applying AMD to burnup matrix, in order to understand how AMD method works, let us suppose that we want to decompose a symmetric positive matrix A by using Cholesky factorization. There are many ways of factorizing a symmetric positive matrix A into L and L^T. Among them, a right-looking sparse Cholesky factorization is closely related to AMD. The pseudo code in terms of MATLAB language of the right-looking Cholesky factorization algorithm is given in Fig. 1.

```
For k=1:n
  L(k,k)=srt(A(k,k));
  L(k+1:n,k)=A(k+1:n,k)/L(k,k);
   A(k+1:n,k+1:n)=A(k+1:n,k+1:n
                  -L(k+1:n,k)*L(k+1:n,k)';
end
```

Fig. 1. Pseudo MATLAB code describing how right-looking Cholesky factorization algorithm works.

For example, the initial 3 steps of Cholesky factorization of 5x5 symmetric matrix can be described schematically in Fig. 2.

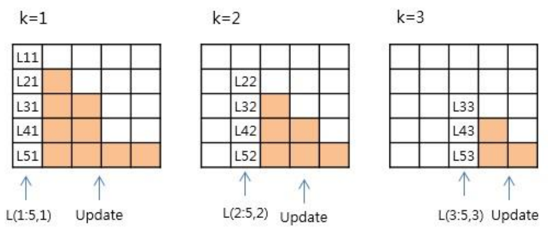


Fig. 2. Schematic illustration of right-looking Cholesky factorization process for 5x5 matrix

In the right-looking factorization algorithm, at every kth step, A(k+1:n,k+1:n) is newly updated by the outer product L(k+1:n,k)*L(k+1:n,k)' as shown in the pseudo code. While doing this outer product, the fill-in of the original matrix happens. In AMD method, this fill-in process is described graphically by using the elimination graph. The elimination graph is a graphical tool that can display the non-zero pattern of a given matrix. Each vertex in a graph represents the row or column index of a matrix. If a given matrix has a non-zero entry in A_{ij}, the graph has an edge connecting between the ith and jth vertex. Because the matrix is assumed to be symmetric, the edge of the two vertices is undirected. As an example, non-zero pattern of the 10x10 matrix and its graph is shown in Fig. 3.

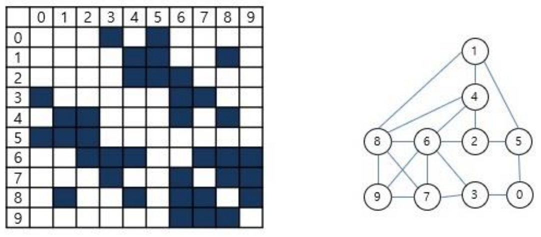


Fig. 3. An example of non-zero pattern of a 10x10 matrix and the corresponding elimination graph

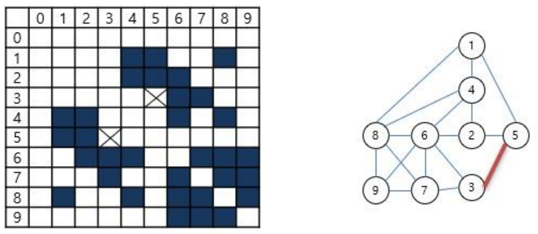


Fig. 4. Non-zero pattern of the matrix after performing the first factorization step and the corresponding elimination graph

Suppose that k=1. Then, the first column of L can be obtained and the remaining part of the matrix, A(2:10,2:10), is newly updated by the outer product. In this process, new fill-in entries appear in the matrix. The newly created fill-in entries are illustrated in Fig. 4. The graph shows that new fill-in entries corresponds to newly created edges connecting originally neighbor vertices of the removed vertex. In general, removing a vertex and adding a dense submatrix to the graph (in graph theory this is called clique) corresponds to creation of fill-in in the matrix at each factorization step. In order to minimize this kind of fill-in, AMD scheme calculates the approximate degree of each vertex by using Eq. (6).

$$\bar{d}_i = |A_i| + |L_k \setminus \{i\}| + \sum_{e\in\varepsilon_i\setminus\{k\}} |L_e \setminus L_k| \quad , \qquad (6)$$

where $|A_i|$ denotes the degree of ith vertex, L_k is the nonzero pattern of L(:,k) and A\B denotes subtracting A from B. The exact degree of the vertex is defined by the number of neighbor vertices connected by the edges.

Actually, Eq. (6) is the definition of the approximate degree of the vertex. The approximate degree has some benefits against exact degree. For example, it requires much less calculation efforts to obtain it. Furthermore, it is known that its performance is much better in terms of fill-in minimization. If approximate degrees of all vertices are calculated, AMD scheme selects the vertex whose approximate degree is minimum as a pivot column.

4. Numerical Results

As stated before, AMD is available only for symmetric matrix. Since the burnup matrix is asymmetric, the AMD cannot be applied directly to the problem. Instead, we apply AMD to $A+A^T$ or A^T*A to determine permutation of a matrix A. The steps of solving Ax=b for calculating Eq. (5) is follow:

1. Find a permutation to reduce fill-in by using AMD scheme described in the previous chapter for the matrix $A+A^T$ or A^T*A. This step is called symbolic

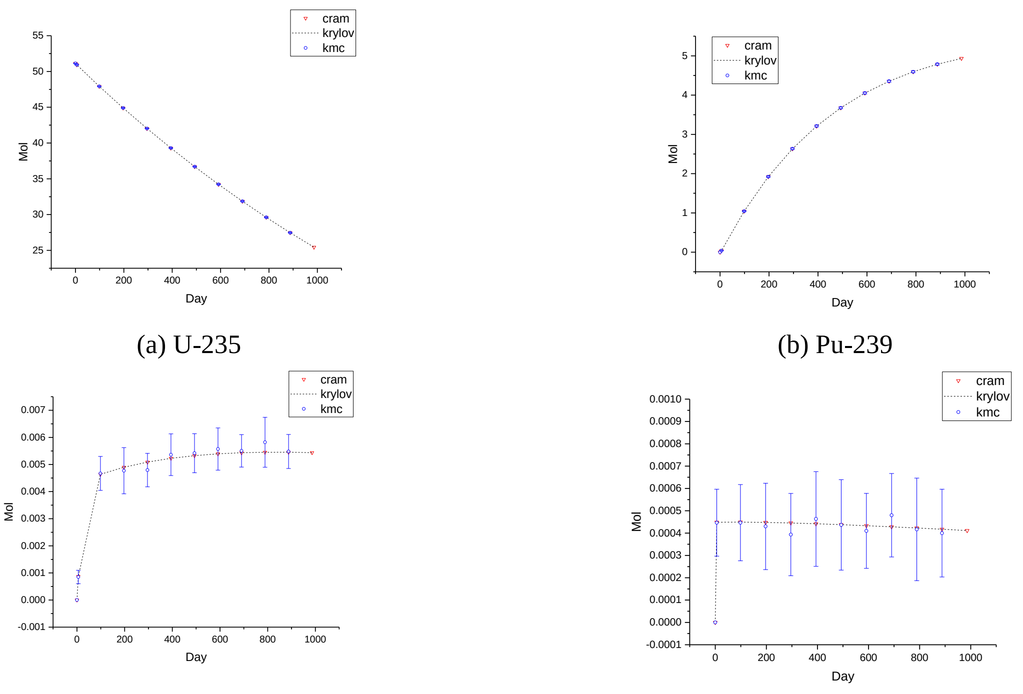
factorization because it depends only on the nonzero pattern of the matrix, not the numerical values.

2. Perform LU factorization for the permuted matrix obtained in step 1
3. Find the solution vector x through backward and forward substitutions.

In order to verify the exactness of the solution and performance of the AMD ordering scheme, a code is developed to solve the depletion equation based on CRAM which includes LU factorization solver with AMD ordering scheme. One group ORIGEN library is used as a depletion library. Table I shows the initial condition of the test problem at a specified region of a core. While burn proceeds, the power level is assumed to be constant as 6.4MWt. For verification of exactness of solution, the results are compared with other burnup calculation codes employing different methodology other than CRAM.

Table I: Initial conditions for the test problem

Isotope	**Mol**	**Isotope**	**Mol**	**Isotope**	**Mol**
H-1	4.18E+03	B-5	3.47E-01	B-11	1.40E+00
C-12	5.47E-01	C-13	5.91E-03	O-16	4.22E+03
U-235	5.11E+01	U-238	1.01E+03		



(a) U-235 (b) Pu-239

(c) Sm-149 (d) Xe-135

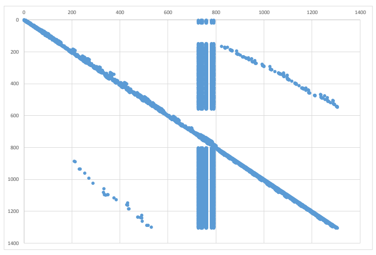
Fig. 5. Comparison of time evolution of concentration of various nuclides for the test problem.

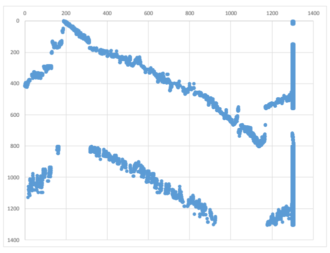
The comparisons of the results are performed with MEDEAC code [3] and a Monte Carlo burnup solver developed in KAERI which employs KMC algorithm [4]. Fig. 5 shows that the time evolution of each code is well consistent with each other. So it ensures the CRAM method is realized in a proper way.

To see the performance of the AMD ordering scheme, the execution times are measured and compared between codes by varying the ordering scheme as shown in Table II. The reference is set by the execution time of the MEDEAC code. Note that the order of CRAM is k=14 and the number of sub-steps is determined by 4 in those calculations. It shows that the execution time of CRAM14 is ~4 times larger than the MEDEAC code when Ax=b is solved by using LU factorization without ordering scheme. In this case, non-zero pattern of the burnup matrix is appeared as given in Fig. 6. (a).

Table II. Execution time of each ordering scheme recorded for the test problem

Methodology	SEC
Krylov subspace method	0.28
CRAM14 (Natural ordering)	1.21
CRAM14 (AMD for $A+A^T$)	0.29
CRAM14 (AMD for A^T*A)	0.30


(a) Non-zero Pattern of Burnup Matrix Without Ordering Scheme


(b) Non-zero Pattern of Burnup Matrix When AMD Ordering Scheme is Applied to A+A^T

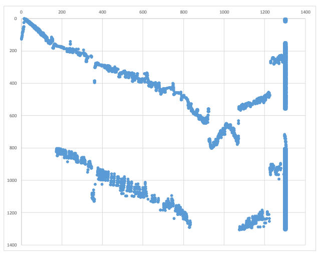

(c) Non-zero Pattern of Burnup Matrix When AMD Ordering Scheme is Applied to A^T*A

Fig. 6. Non-zero Patterns of Burnup Matrix Depending On Ordering Scheme

Table II shows that the execution time decreases to 1/4 times smaller than the case without ordering scheme and it is comparable to that of MEDEAC code. It appears that the difference between ordering scheme A+A^T and A^T*A is not large. Non-zero patterns of the burnup matrix when AMD ordering scheme is used are shown in Fig. 6 (b) and (c).

5. Conclusions

Solving Ax=b is essential for CRAM. In this study, a direct solver is chosen as an inverse matrix solver of CRAM. In order to reduce execution time of the solver, fill-in reducing algorithm is introduced called AMD. The AMD uses the elimination graph to predict fill-in of the matrix graphically. It calculates the approximate degree of all vertices in elimination tree and selects the vertex as a pivot row or column when it has the minimum approximate degree out of all vertices. The results show that the execution time of CRAM is improved about 4 times faster by AMD ordering scheme.

ACKNOWLEDGMENT

This study was supported by the National Research foundation of Korea (NRF) grant funded by the Korea Government (MSIT). (No. 2017M2A8A1092448)

REFERENCES

[1] Pusa, M. and Leppanen, J. (2010) "Computing the matrix exponential in burnup calculations." Nucl. Sci. Eng., 164 (2010) 140-150.
[2] Patric R. Amestoy, T. A. Davis, I. S. Duff. (1996) “An Approximate Minimum Degree Ordering Algorithm.” SIAM J. Matrix Analysis & Applic., Vol 17, no 4, pp. 886-905.
[3] J. Y. Cho et al., MEDEAC: Matrix Exponential based isotope DEpletion and Analysis Code, Transactions of the Korean Nuclear Society Autumn Meeting, Gyeongju, Korea, October 26–27, 2017.
[4] H. J. Shim, Stochastic Perturbation Algorithm for Kinetic Monte Carlo Simulations, Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA+MC2013), Paris, France, October 27–31, 2013.