100
11/19/91 (2)

Conf-920308--4

GE Industrial & Power Systems
Knolls Atomic Power Laboratory

A Variable Timestep Generalized Runge-Kutta Method for the Numerical Integration of the Space-Time Diffusion Equations

BN Aviles
TM Sutton
DJ Kelly, III

KAPL-4731
September 1991

KAPL--4731

DE92 003010

A Variable Timestep Generalized Runge-Kutta Method for the Numerical Integration of the Space-Time Diffusion Equations

B.N. Aviles, T.M. Sutton, and D.J. Kelly, III

September 1991

Prepared for presentation at the
American Nuclear Society Topical Meeting on Advances in Reactor Physics
March 8-11, 1992

Prepared by
General Electric Company
KNOLLS ATOMIC POWER LABORATORY
Schenectady, New York

Contract No. DE-AC12-76-SN-00052

Disclaimer

CONTENTS

TABLES

FIGURES

Abstract

A generalized Runge-Kutta method has been employed in the numerical integration of the stiff space-time diffusion equations. The method is fourth-order accurate, using an embedded third-order solution to arrive at an estimate of the truncation error for automatic timestep control. The efficiency of the Runge-Kutta method is enhanced by a block-factorization technique that exploits the sparse structure of the matrix system resulting from the space and energy discretized form of the time-dependent neutron diffusion equations. Preliminary numerical evaluation using a one-dimensional finite difference code shows the sparse matrix implementation of the generalized Runge-Kutta method to be highly accurate and efficient when compared to an optimized iterative theta method.

A Variable Timestep Generalized Runge-Kutta Method for the Numerical Integration of the Space-Time Diffusion Equations

B.N. Aviles, T.M. Sutton, and D.J. Kelly, III

I. INTRODUCTION

Several time differencing methods have been developed for the solution of the space-time diffusion equations. Examples include the theta method[1-3], the improved quasi-static method[4-5], the stiffness confinement method[6], and the exponential transform method[7]. Recently, an implicit generalized Runge-Kutta (GRK) method developed by Kaps and Rentrop[8-9] has been applied successfully to the solution of the point kinetics equations[10]. The Kaps-Rentrop GRK method has been shown to be more accurate and efficient than other time differencing techniques for point kinetics transients due to its higher order and systematic variable timestep algorithm[11]. This paper describes a sparse matrix implementation of the Kaps-Rentrop GRK method that is suitable for space-time diffusion applications.

II. KAPS-RENTROP GENERALIZED RUNGE-KUTTA FORMULAS

We begin with the following initial value problem:

$$y'(x) = f(y(x)) \tag{1}$$

where

$y(x)$ = vector of N dependent variables,

$y'(x)$ = vector of $y(x)$ derivatives with respect to x, and

$y(x_0) = y_0$.

To advance the system solution from x_0 to $x_0 + h$, the Kaps-Rentrop GRK method[8-9] forms the following general solution:

$$y(x_0 + h) = y_0 + \sum_{i=1}^{s} c_i k_i \tag{2}$$

where

h = step size,

s = number of stages,

c_i = fixed expansion constants, and

k_i = vector of expansion coefficients.

The k_i vectors are found by solving a system of N linear equations for s different right hand sides[8]:

$$\left[I - \gamma h f'(y_0)\right] k_i = hf\left(y_0 + \sum_{j=1}^{i-1} \alpha_{ij} k_j\right) + hf'(y_0) \sum_{j=1}^{i-1} \gamma_{ij} k_j \text{ , for } i = 1, \ldots, s \tag{3}$$

where

I = identity matrix,

γ, γ_{ij}, α_{ij} = problem independent fixed constants,

$f(\ldots)$ = explicit function evaluation, and

$f'(y_0)$ = N x N Jacobi matrix of partial derivatives, $\left.\frac{\partial f}{\partial y}\right|_{y_0}$.

The Kaps-Rentrop scheme employs a Runge-Kutta-Fehlberg method[12] to obtain an embedded estimate for $y(x_0 + h)$. This is accomplished by computing a lower order estimate, $\hat{y}(x_0 + h)$, using different expansion constants, $\hat{c}_i$, for $i = 1, \ldots, \hat{s}$, where $\hat{s} < s$, but the k_i expansion coefficients remain the same. The smallest values allowing an embedded solution are $\hat{s} = 3$ and $s = 4$, resulting in a fourth-order method[8].

Expanding Eq. (3) for $s = 4$ yields the four linear equation systems to be solved for the k_i vectors[9]:

$$(i) \qquad [I - \gamma h f'(y_0)]\, k_1 = hf(y_0)$$

$$(ii) \qquad [I - \gamma h f'(y_0)]\, (k_2 + \frac{\gamma_{21}}{\gamma} k_1) = hf(y_0 + \alpha_{21} k_1) + \frac{\gamma_{21}}{\gamma} k_1$$

$$(iii) \qquad [I - \gamma h f'(y_0)]\, (k_3 + \frac{\gamma_{31}}{\gamma} k_1 + \frac{\gamma_{32}}{\gamma} k_2) = hf(y_0 + \alpha_{31} k_1 + \alpha_{32} k_2) + \frac{\gamma_{31}}{\gamma} k_1 + \frac{\gamma_{32}}{\gamma} k_2 \tag{4}$$

$$(iv) \qquad [I - \gamma h f'(y_0)]\, (k_4 + \frac{\gamma_{41}}{\gamma} k_1 + \frac{\gamma_{42}}{\gamma} k_2 + \frac{\gamma_{43}}{\gamma} k_3)$$

$$= hf(y_0 + \alpha_{41} k_1 + \alpha_{42} k_2 + \alpha_{43} k_3) + \frac{\gamma_{41}}{\gamma} k_1 + \frac{\gamma_{42}}{\gamma} k_2 + \frac{\gamma_{43}}{\gamma} k_3 \text{ .}$$

Each integration step requires one L-U decomposition of the matrix $[I - \gamma h f'(y_0)]$ followed by four back-substitution steps to determine the four k_i vectors.

The fixed expansion constants, γ, γ_{ij}, and α_{ij}, result from solving the "equations of condition" associated with the Runge-Kutta method. Kaps and Rentrop provide two sets of constants and their associated regions of stability[8]. The set chosen for this study, denoted as GRK4T, is A (89.3°) -stable for the fourth-order method and has been shown to perform better than other constant sets in numerical tests[8]. The constants are listed in Table 1.

III. AUTOMATIC STEPSIZE CONTROL

The Kaps-Rentrop method exploits the availability of both third- and fourth-order estimates of $y(x_0 + h)$ to control step size automatically[8]. If we denote:

$$y_{4th} = y(x_0 + h) = y_0 + \sum_{i=1}^{4} c_i k_i \tag{5}$$

and

$$y_{3rd} = \hat{y}(x_0 + h) = y_0 + \sum_{i=1}^{3} \hat{c}_i k_i ,$$

then the exact solution, y_{exact}, can then be expressed as:

$$y_{exact} = y_{4th} + h^5 g_{5th} + \ldots = y_{3rd} + h^4 g_{4th} + \ldots \tag{6}$$

where g_{5th} and g_{4th} represent higher order terms. Thus, the maximum scaled truncation error for the step is:

$$Err = max\left|\frac{y_{4th} - y_{3rd}}{y_{scal}}\right| = max\left|\frac{h^4 g_{4th} + \ldots}{y_{scal}}\right| , \tag{7}$$

where y_{scal} is an appropriate scaling vector. For a successful integration step, we require the maximum scaled truncation error to be smaller than a specified error tolerance, ε:

$$Err < \varepsilon . \tag{8}$$

In order for the next timestep, h_{next}, to be successful, we also require:

$$\left(\frac{max\left|h_{next}^4 g_{4th,\,next}\right|}{y_{scal}}\right) < \varepsilon . \tag{9}$$

If we assume $g_{4th,next} = g_{4th}$, an estimate of h_{next} can be made:

$$h_{next} < h\left(\frac{\varepsilon}{Err}\right)^{1/4} \tag{10}$$

Finally, to reduce the number of steps that are rejected and to prevent a zig-zag behavior in successful step sizes, a safety factor of 0.9 is included in the determination of h_{next}, and h_{next} is bounded:

$$h_{next} < 0.9 \cdot h\left(\frac{\varepsilon}{Err}\right)^{1/4}, \quad 0.5h < h_{next} < 1.5h . \tag{11}$$

IV. APPLICATION OF THE KAPS-RENTROP METHOD TO THE TIME-DEPENDENT DIFFUSION EQUATIONS

The next two sections describe the implementation of the Kaps-Rentrop method for space-time diffusion theory applications. The few-group, time-dependent diffusion equations without group skipping are written as:

$$\frac{1}{v_g}\frac{\partial \Phi_g}{\partial t} = D_g \nabla^2 \Phi_g - \Sigma_g \Phi_g + \Sigma_{g-1 \to g} \Phi_{g-1} + (1-\bar{\beta})\chi_g^p \sum_{g'} \frac{\nu}{k_{eff}} \Sigma_{fg'} \Phi_{g'} + \chi_g^d \sum_l \lambda_l C_l$$

$$\text{for } g = 1, \ldots, G$$

and (12)

$$\frac{\partial C_l}{\partial t} = \bar{\beta}_l \sum_{g'} \frac{\nu}{k_{eff}} \Sigma_{fg'} \Phi_{g'} - \lambda_l C_l$$

$$\text{for } l = 1, \ldots, L$$

where the spatial indexing has been suppressed for simplicity. In matrix form, Eqs. (12) become:

$$\frac{dy}{dt} = [F]y \tag{13}$$

where

$$y = col\left[\vec{\Phi}_1, \vec{\Phi}_2, \ldots, \vec{\Phi}_G, \vec{C}_1, \vec{C}_2, \ldots, \vec{C}_L\right],$$

$$\vec{\Phi}_g = col\left[\Phi_g^1, \Phi_g^2, \ldots, \Phi_g^M\right],$$

$$\vec{C}_l = col\left[C_l^1, C_l^2, \ldots, C_l^M\right],$$

M = total number of nodes,

G = number of flux energy groups,

L = number of delayed neutron precursor groups, and

$[F]$ = coefficient matrix arising from the time dependent diffusion equations.

Figure 1 presents the generic structure of the $[F]$ matrix for a three-dimensional, two flux energy group and six delayed neutron precursor group problem. It is important to note that the difference in the matrix structure for one-, two-, or three-dimensional geometries is manifested in the G three-, five-, or seven-stripe diagonal flux blocks.

The Kaps-Rentrop Jacobi matrix, $\left[I - \gamma h f'(y_0)\right]$, retains the structure of the $\left[F\right]$ coefficient matrix. Therefore, each timestep requires one L-U decomposition of a matrix with the structure of Figure 1, followed by four back-substitutions to obtain the four k_i expansion coefficient vectors. It is obvious that, for a large number of nodes, an L-U decomposition of the full matrix becomes prohibitive. Iterative solution methods are not thought practical in this situation for two reasons. First, four complete iterative solutions must be performed each timestep to obtain the four k_i vectors. Second, the k_i vectors can be numerically very close to zero for mild transients and are identically zero for a null (extended steady state) transient. Iterating on near-zero values leads to slow convergence behavior for standard iterative procedures. Therefore, a direct matrix solution procedure tailored to the sparse matrix structure of Figure 1 has been developed and is described next.

V. SPARSE MATRIX L-U DECOMPOSITION ALGORITHM

The structure of the Jacobi Matrix (Figure 1) lends itself easily to a block factorization L-U decomposition technique. The sparse structure is of a doubly-bordered block diagonal form[13]. This section describes an efficient block factorization technique, referred to in reference 13 as "one-way dissection," that can be used to solve a system having a doubly-bordered block diagonal form. This method reduces the solution computational time when compared to a standard L-U decomposition of the full matrix system. Also, storage reduction techniques that reduce the amount of required memory have been implemented.

We begin by breaking the Jacobi matrix into blocks denoted by $\left[A_{ij}\right]$ in Figure 2. Because the structures of the $\left[A_{12}\right]$, $\left[A_{21}\right]$, and $\left[A_{22}\right]$ matrices are known, they can be stored in vector rather than matrix form:

1) The diagonal matrix $\left[A_{22}\right]$ $(M \cdot L \times M \cdot L)$ is stored in a vector of length $M \cdot L$.

2) The matrix $\left[A_{12}\right]$ $(M \cdot G \times M \cdot L)$ is stored by row in a vector of length $M \cdot G \cdot L$.

3) The matrix $\left[A_{21}\right]$ $(M \cdot L \times M \cdot G)$ is stored by column in a vector of length $M \cdot G \cdot L$.

Currently, storage of the $\left[A_{11}\right]$ matrix remains full $(M \cdot G \times M \cdot G)$, although its sparse structure can also be exploited to reduce storage.

Given the matrix system with multiple right hand sides, $\left[A\right]k_i = b_i$, the one-way dissection proceeds as follows:

1) Decompose the system matrix, $\left[A\right]$, into blocks as described above. Note that the constant vector and the unknown vector will each be split into two pieces, b_{1i},b_{2i},k_{1i}, and k_{2i}, respectively.

2) Compute: $\left[D_{11}\right] = \left[A_{11}\right] - \left[A_{12}\right]\left[A_{22}\right]^{-1}\left[A_{21}\right]$. (14)

Note that this operation can be performed efficiently because inverting the $\left[A_{22}\right]$ matrix is trivial and the product operations involve only vector multiplication. The $\left[D_{11}\right]$ matrix retains the structure of the original $\left[A_{11}\right]$ block.

3) Perform an L-U decomposition on the $[D_{11}]$ matrix. Currently, an L-U decomposition is performed on the full $[D_{11}]$ matrix. For large problems or many neutron energy groups, the sparse structure of this matrix must also be exploited with an appropriate decomposition technique.

For each of the four k_i vectors and associated constant vectors, b_i, from the Kaps-Rentrop equations (Eq. 5):

4) Compute $q_{2i} = [A_{22}]^{-1} b_{2i}$. (15)

5) Solve: $[D_{11}]q_{1i} = b_{1i} - [A_{12}]q_{2i}$ for q_{1i} by back substitution. (16)

6) Then $k_{1i} = q_{1i}$ and $k_{2i} = q_{2i} - [A_{22}]^{-1}[A_{21}]k_{1i}$. (17)

In summary, for each Kaps-Rentrop integration step, steps (1) through (3) must be performed once to arrive at the L-U form of the $[D_{11}]$ matrix. Steps (4) through (6) must then be performed four times to compute the k_i vectors. The most CPU intensive step of the sparse matrix algorithm is the L-U decomposition in step (3).

VI. METHOD EVALUATION USING A ONE-DIMENSIONAL FINITE-DIFFERENCE CODE

The sparse matrix block factorization implementation of the Kaps-Rentrop method (denoted as K-R Sparse) was evaluated using a one-dimensional finite-difference code. The K-R Sparse algorithm was compared to the Kaps-Rentrop method with an L-U decomposition performed on the full Kaps-Rentrop Jacobi matrix (hereafter referred to as K-R Full). The K-R Sparse algorithm was also compared to an optimized Theta method employing a block successive over-relaxation (block-SOR) technique[14]. The Theta method was optimized for the one-dimensional finite-difference code by (1) using an explicit solution method for the (1-D) tridiagonal flux blocks[15], (2) computing an optimum over-relaxation parameter from the spectral radius of the Jacobi iteration matrix, (3) choosing the implicit method ($\theta = 1$) for both fluxes and precursors and (4) employing a step-doubling technique for error control and automatic timestep selection[15].

Several transient comparisons were made for a one-dimensional geometry based on the ANL benchmark problem BSS-6[16]. The geometry and group constant data are presented in Figure 3 and Table 2, respectively. All transients were initiated from a tightly converged solution obtained using the finite-difference option of the KAPL Nodal Expansion Method code, NODEX[17]. All reported CPU times are for a single processor on a CRAY YMP/8.

VI.A. Transient 1: Ramp Reactivity Increase

The first transient involved a 1% linear decrease of Region 1 thermal absorption cross section in 1.0 second (ANL BSS-6-A2)[16]. Each model consisted of 120 nodes. The initial timestep size was 1.0E-3 seconds and the relative truncation error tolerance, ε, was chosen to be 1.0E-2 for both the Kaps-Rentrop and Theta method to ensure accurate solutions. Table 3 summarizes the results and Figure 4 is a plot of power versus time. From Table 3, it can be seen that both the Kaps-Rentrop

and Theta method are very accurate (error in total power at t=4 seconds is less than 0.4%), but the K-R Sparse method required fewer timesteps due to its higher order accuracy and required significantly less CPU time than did the Theta method. The effects of the block factorization technique and the improved vector storage can be seen in the K-R Sparse versus K-R Full results. The two implementations of the Kaps-Rentrop method produce identical results, but the sparse matrix implementation executes the transient in 1/40th of the CPU time and requires 1/25th of the storage.

VI.B. Transient 2: Sinusoidal Reactivity Perturbation

The second transient was a sinusoidal reactivity perturbation due to a sinusoidal variation of Region 1 thermal absorption cross section by 1% with a period of 1.0 second continuously for four seconds. Node size, error tolerance, and initial timestep size remained the same as in Transient 1. Table 4 and Figure 5 present the results of this simulation. Both the Kaps-Rentrop and Theta method exhibit similar accuracy, forced by the truncation error tolerance requirements. The results of this transient highlight the benefit of the fourth-order Runge-Kutta method over the first-order Theta method. The Kaps-Rentrop method required only 92 timesteps and 8.8 seconds to complete the transient, whereas the Theta method required 2.7 times as many steps and over forty times more CPU time.

VII. EXTENSION OF KAPS-RENTROP METHOD TO OTHER PROBLEM SIZES

The results presented in this paper involve a one-dimensional geometry with 120 nodes. A question arises as to how well the Kaps-Rentrop method performs for larger problems. For the 1-D finite difference cases, a parametric study was performed comparing CPU time per timestep versus problem size for the Kaps-Rentrop method and optimized 1-D Theta method. Because both Kaps-Rentrop methods use a direct L-U decomposition, the CPU time per timestep scales roughly as N^3, where $N = M \cdot (G+L)$ for K-R Full, and $N = M \cdot G$ for K-R Sparse. The iterative block-SOR Theta method scales approximately linearly with N, where $N = M \cdot (G+L)$. Therefore, at some problem size (roughly 600 nodes for the sinusoid problem), the K-R Sparse algorithm requires more CPU time per timestep than does the Theta method. However, recall that Kaps-Rentrop took three times fewer steps to complete the transient, indicating that the K-R Sparse method would still require less total CPU time for a 960 node problem.

For three-dimensional geometries, the Kaps-Rentrop CPU time per timestep curve would remain basically the same because we are performing an L-U decomposition on the full $[D_{11}]$ flux matrix $(M \cdot G \times M \cdot G)$. The Theta method will necessarily become more expensive because the flux blocks will lose the desirable tridiagonal structure. Therefore, the CPU time per timestep comparison would favor the Kaps-Rentrop Sparse method for 3-D geometries. For very large three-dimensional problems, a more efficient L-U decomposition algorithm exploiting the structure of the $[D_{11}]$ matrix must be employed.

VIII. CONCLUSIONS

This paper has described a variable timestep Generalized Runge-Kutta method suitable for time-dependent diffusion theory applications. The basic method, timestep control, and sparse matrix

implementation are derived from sound mathematical theory. The method utilizes a direct matrix solution algorithm, therefore its CPU time per timestep behavior is invariant to the type of transient and depends only on the problem size. The sparse matrix implementation of the Kaps-Rentrop GRK method is fourth-order accurate and very efficient for the range of 1-D test problems investigated. The Kaps-Rentrop sparse matrix method took consistently fewer timesteps and less CPU time when compared to an optimized 1-D Theta method.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. M.R. Mendelson and Dr. R.G. Gamino for their suggestions during this research.

REFERENCES

1. A.V. Vota, N.J. Curlee, and A.F. Henry, "WIGL3 - A Program for the Steady-State and Transient Solution of the One-Dimensional, Two-Group, Space Time Diffusion Equations Accounting for Temperature, Xenon, and Control Feedback," WAPD-TM-788, February 1969.

2. K.S. Smith, "An Analytic Nodal Method for Solving the Two-Group, Multidimensional Static and Transient Neutron Diffusion Equations," S.M. and N.E. Thesis, Massachusetts Institute of Technology, Cambridge, MA, March 1979.

3. T.A. Taiwo and H.S. Khalil, "The DIF3D Nodal Kinetics Capability in Hex-Z Geometry -- Formulation and Preliminary Tests," Proc. Adv. in Math., Comp., and Reactor Phys., Vol. 5, 23.2 2-1, Pittsburgh, PA, April 28 - May 2 1991.

4. P.W. Kao, "Application of Supernodal Methods to Transient Analysis," Ph.D. Thesis, Massachusetts Institute of Technology Department of Nuclear Engineering, Cambridge, MA, August 1988.

5. H. Sano, Y. Ishii, J. Koyama, and Y. Bessho, "Three-Dimensional, Multi-Group Nuclear Reactor Kinetic Program Based on a Vectorized Nodal Expansion Method," Proc. Adv. in Math., Comp., and Reactor Phys., Vol. 5, 30.2 5-1, Pittsburgh, PA, April 28 - May 2 1991.

6. Y.A. Chao and P. Huang, "Theory and Performance of the Fast-Running Multi-Dimensional PWR Kinetic Code, SPNOVA-K," Proc. ANS Int. Reac. Phys. Conf., IV-153, Jackson Hole, WY, September 18-22, 1988.

7. P.K. Hutt and M.P. Knight, "The Development of a Transient Neutron Flux Solution in the PANTHER Code," Trans. Amer. Nucl. Soc., Vol. 61, p. 348, Nashville, TN, June 1990.

8. P. Kaps and P. Rentrop, "Generalized Runge-Kutta Methods of Order Four with Stepsize Control for Stiff Ordinary Differential Equations," Numerische Mathematik, Vol. 33, pp. 55-68, 1979.

9. W.H. Press and S.A. Teukolsky, "Integrating Stiff Ordinary Differential Equations," Computers in Physics, May/June 1989, pp. 88-91.

10. J. Sanchez, "On the Numerical Solution of the Point Reactor Kinetics Equations by Generalized Runge-Kutta Methods," Nucl. Sci. and Eng., Vol. 103, pp. 94-99, 1989.

11. D.J. Kelly and D.R. Harris, "Comparison of Variable Timestep Differencing Methods Applied to the Point Reactor Kinetic Equations," to be published in Trans. Amer. Nucl. Soc., Vol. 64, San Francisco, CA, Nov. 1991.

12. J. Stoer and R. Bulirsch, Introduction to Numerical Analysis, Springer-Verlag, New York, 1980.

13. I.S. Duff, A.M. Erisman and J.K. Reid, Direct Methods for Sparse Matrices, Oxford University Press, New YOrk, 1986.

14. J.R. Westlake, A Handbook of Numerical Matrix Inversion and Solution of Linear Equations, John Wiley and Sons, New York, 1968.

15. W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, Numerical Recipes: The Art of Scientific Computing, Cambridge University Press, New York, 1986.

16. National Energy Software Center, Benchmark Problem Book, ANL-7416-Supplement 3, December 1985.

17. T.M. Sutton, "NODEX: A High Order NEM-Based Multigroup Nodal Code," Proc. Adv. Nucl. Eng. Comp. and Rad. Shield., Sante Fe, NM, April 1989.

TABLE 1. GRK4T Expansion Constants (reference 8)

$\gamma = 0.231$	$\gamma_{21} = -0.270629667752$
$\gamma_{31} = 0.311254483294$	$\gamma_{32} = 0.852445628482E-2$
$\gamma_{41} = 0.282816832044$	$\gamma_{42} = -0.457959483281$
$\gamma_{43} = -0.111208333333$	
$\alpha_{21} = 0.462$	
$\alpha_{31} = -0.815668168327E-1$	$\alpha_{32} = 0.961775150166$
$\hat{c}_1 = -0.717088504499$	$\hat{c}_2 = 0.177617912176E-1$
$\hat{c}_3 = -0.590906172617E-1$	
$c_1 = 0.217487371653$	$c_2 = 0.486229037990$
$c_3 = 0.0$	$c_4 = 0.296283590357$

TABLE 2. ANL BSS-6 Initial Two Group Constants (reference 16)

Constant		Region 1,3	Region 2
D_1	(cm)	1.5	1.0
D_2	(cm)	0.5	0.5
Σ_{r1}	(cm^{-1})	0.026	0.02
Σ_{r2}	(cm^{-1})	0.18	0.08
$\Sigma_{1\rightarrow 2}$	(cm^{-1})	0.015	0.01
$\nu\Sigma_{f1}$	(cm^{-1})	0.010	0.005
$\nu\Sigma_{f2}$	(cm^{-1})	0.2	0.099
χ_1		1.0	1.0
χ_2		0.0	0.0
v_1	(cm/s)	1.0E+7	1.0E+7
v_2	(cm/s)	3.0E+5	3.0E+5

Removal cross section includes capture, fission, and downscatter.

Delayed Neutron Parameters

Group	$\bar{\beta}$	λ (s^{-1})
1	0.00025	0.0124
2	0.00164	0.0305
3	0.00147	0.1110
4	0.00296	0.3010
5	0.00086	1.1400
6	0.00032	3.0100

TABLE 3. Comparison of Kaps-Rentrop and Theta Method Performance: 1-D Ramp Transient

Method	Reference[1]	K-R Sparse	K-R Full	Theta Method
Number of Nodes	120	120	120	120
Error Tolerance[2]	---	0.01	0.01	0.01
Initial Timestep Size (s)	0.001	0.001	0.001	0.001
Total Number of Timesteps	4000	28	28	41
Total CPU Time[3] (s)	NA	2.7	109.1	81.5
Required Storage (MWords)	NA	0.075	1.86	---[4]
Relative Power				
t = 0.0 s	1.000	1.000	1.000	1.000
t = 0.1 s	1.028	1.028	1.028	1.028
t = 0.2 s	1.063	1.062	1.062	1.061
t = 0.5 s	1.205	1.204	1.204	1.202
t = 1.0 s	1.740	1.738	1.738	1.736
t = 1.5 s	1.959	1.957	1.957	1.956
t = 2.0 s	2.166	2.163	2.163	2.163
t = 3.0 s	2.606	2.602	2.602	2.599
t = 4.0 s	3.108	3.102	3.102	3.096
Regional Power Fractions at t = 4.0 s				
Region 1	0.4424	0.4425	0.4425	0.4424
Region 2	0.4306	0.4303	0.4303	0.4303
Region 3	0.1272	0.1271	0.1271	0.1272

[1]reference for ramp is RAUMZEIT with a fixed 0.001 second timestep (reference 16)

[2]relative error tolerance used in automatic timestep control

[3]single processor on CRAY YMP/8

[4]Theta method not optimized for storage

TABLE 4. Comparison of Kaps-Rentrop and Theta Method Performance: 1-D Sinusoid Transient

Method	Reference[1]	K-R Sparse	K-R Full	Theta Method
Number of Nodes	120	120	120	120
Error Tolerance[2]	---	0.01	0.01	0.01
Initial Timestep Size (s)	0.0001	0.001	0.001	0.001
Total Number of Timesteps	40,000	92	92	252
Total CPU Time[3] (s)	15,910	8.8	357.7	373.3

[1]reference for sinusoid is Theta method with a fixed 0.0001 second timestep

[2]relative error tolerance used in automatic timestep control

[3]single processor on CRAY YMP/8

FIGURE 1. Generic Structure of the Time-Dependent Diffusion Equation Coefficient Matrix

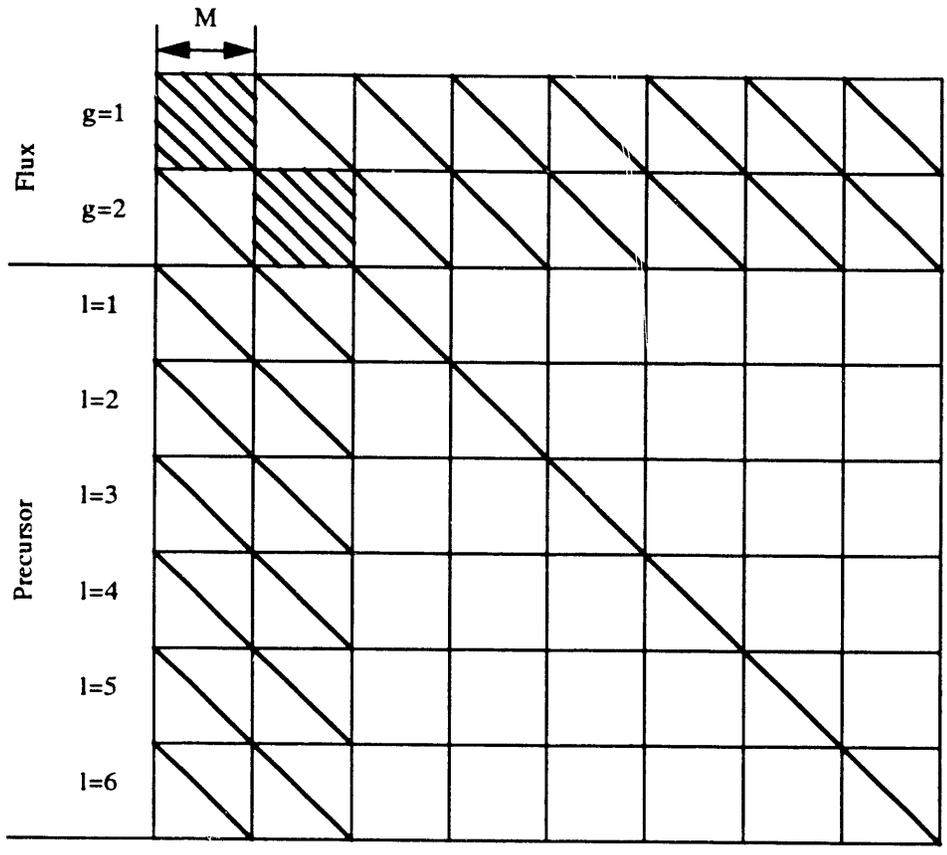


FIGURE 2. Kaps-Rentrop Method Jacobi Matrix in Block Form

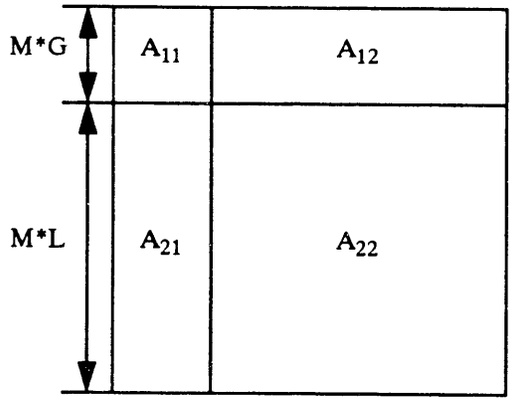


FIGURE 3. ANL-BSS-6 Geometry (reference 16)

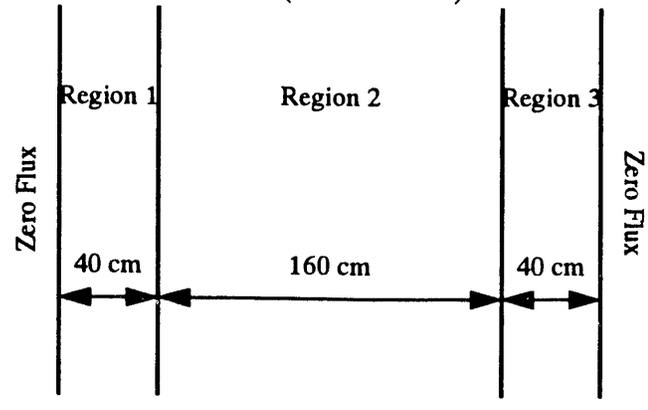


FIGURE 4: ANL BSS-6-A2, RAMP REACTIVITY
120 NODES, 2 FLUX, 6 PRECURSOR GROUPS

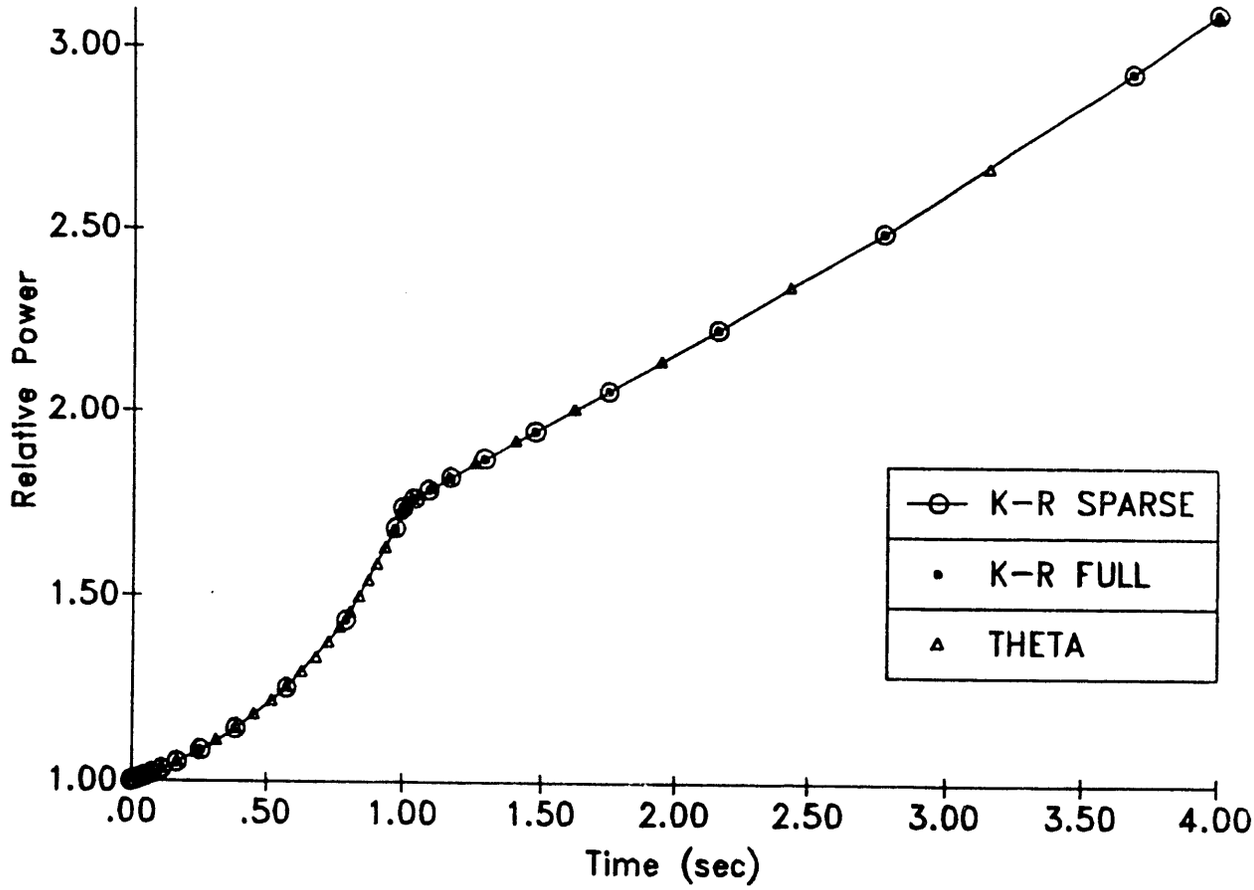
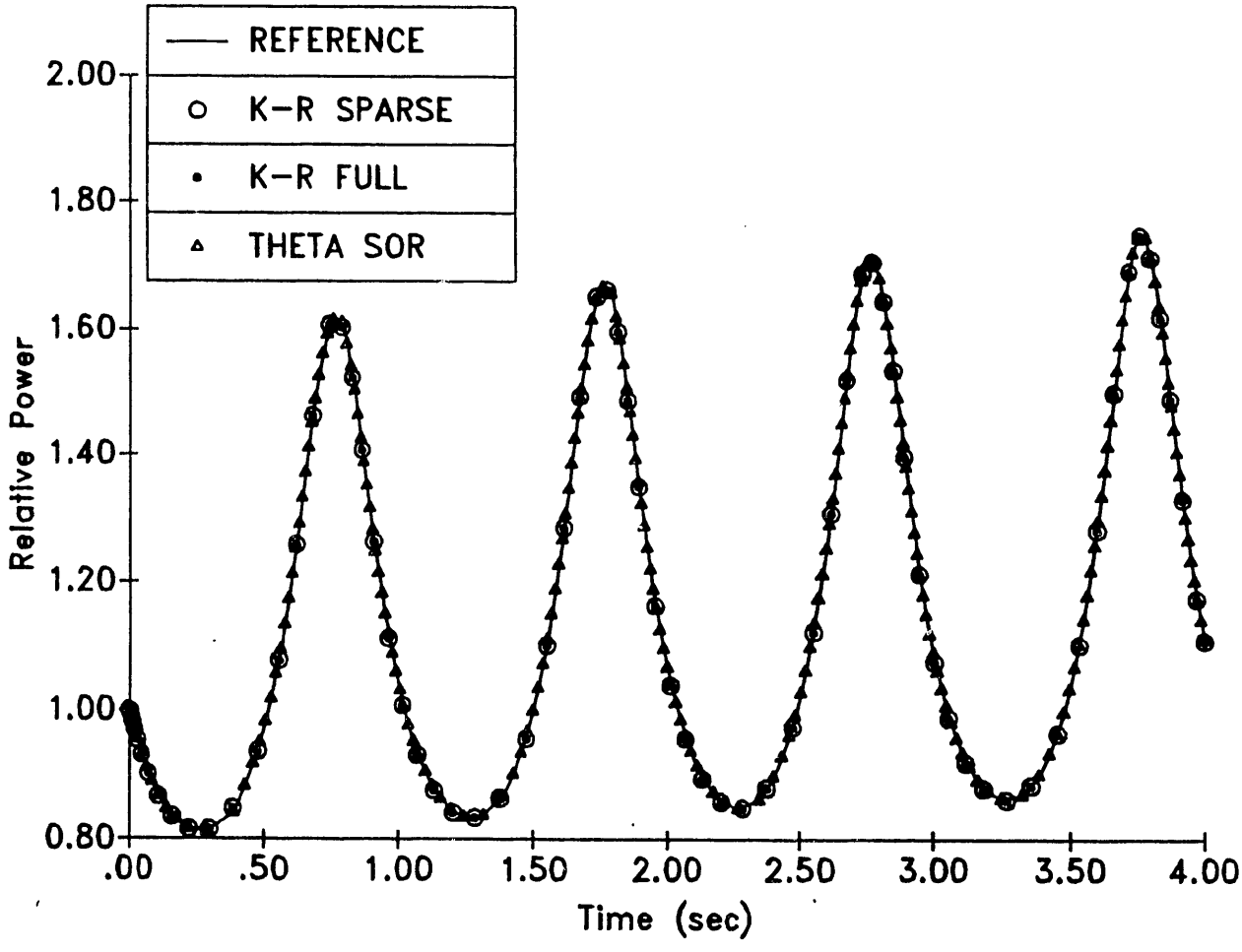


FIGURE 5: ANL BSS-6, SINUSOIDAL REACTIVITY
120 NODES, 2 FLUX, 6 PRECURSOR GROUPS



END

DATE
FILMED
2/10/92