NUCLEAR ENGINEERING

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

DERIVATION OF THE DIAGONAL θ-WEIGHTING METHOD

MITNE-278

by
Michael L. Zerkle

September 1987

DERIVATION OF THE DIAGONAL θ-WEIGHTING METHOD

MITNE-278

by
Michael L. Zerkle

September 1987

ABSTRACT

A new stiffness decoupled method for solving the point kinetics equations, the Diagonal θ-Weighting Method (DTM), is derived and benchmarked against a traditional θ-weighting method using Crank-Nickolson weighting. It is concluded that the traditional θ-weighting method is superior to the DTM and other stiffness decoupled methods.

ACKNOWLEDGMENTS

The author would like to express his appreciation to Professor Allan F. Henry for his suggestions and support during the course of this work. Many of the original ideas contained in this report are his.

Support for this work was provided by the Sandia National Laboratory.

Table of Contents

	Page
Chapter 1. Introduction	6
Chapter 2. Theory	7
2.1 Derivation	7
2.2 Quadratic $\omega(t)$ Approximation	9
2.3 θ-Weighting Coefficient Selection	11
2.4 Algorithm	12
Chapter 3. Numerical Results	15
3.1 Ramp Reactivity Insertion	15
3.2 Positive Step Reactivity Insertions	16
3.3 Negative Step Reactivity Insertions	16
Chapter 4. Conclusions and Recommendations	23
Chapter 5. References	24
Appendix A. Derivation of the Traditional θ-Weighting Method	25
Appendix B. DTMTST Program	29
B.1 Input Description	29
B.2 Source Listing	30

List of Figures

		Page
2.1	Diagonal θ-Weighting Method Algorithm	13
A.1	Traditional θ-Weighting Method Algorithm	28

List of Tables

		Page
2.1	Effects of DTM Algorithm Options on Accuracy and Speed: 0.1 β/s Ramp Reactivity Insertion	14
3.1	Comparison of DTM vs. θ-Weighting: 0.1 β/s Ramp Reactivity Insertion, Amplitude Function	17
3.2	Comparison of DTM vs. θ-Weighting: 0.1 β/s Ramp Reactivity Insertion, Inverse Period	18
3.3	Comparison of DTM vs. θ-Weighting: Positive Reactivity Steps, Amplitude Function	19
3.4	Comparison of DTM vs. θ-Weightine: Positive Reactivity Steps, Inverse Period	20
3.5	Comparison of DTM vs. θ-Weighting: Negative Reactivity Steps, Amplitude Function	21
3.6	Comparison of DTM vs. θ-Weightine: Negative Reactivity Steps, Inverse Period	22
4.1	Comparison of Traditional θ-Weighting vs. Stiffness Decoupling Methods: 0.1 β/s Ramp Reactivity Insertion, Δt=0.1 s	23

Chapter 1

INTRODUCTION

The point kinetics equations, in general, are a stiff system of nonlinear, ordinary differential equations. This "stiffness" is due to the orders of magnitude difference between the prompt and delayed neutron time constants. As a result, small time steps are required to ensure the accuracy and stability of the numerical method used to solve the point kinetics equations. Chao and Attard [1] observed that the stiffness characteristic is present only in the time response of the prompt neutron density and not in that of the delayed neutron precursors. Their stiffness confinement method permits the use of large time steps while maintaining accuracy and stablity. Several algorithms using this idea have been developed at MIT by Parlos [2] and Tanker [3].

This report presents a new method for solving the point kinetics equations based on the stiffness decoupling concept, the Diagonal θ-Weighting Method (DTM). The DTM is then benchmarked against a traditional θ-weighting method for a wide range of transients.

Chapter 2

THEORY

2.1 Derivation

The goal of all point kinetics methods is to solve the following initial value problem:

$$\dot{T}(t) = \frac{\rho(t)-\beta}{\Lambda} T(t) + \sum_{i=1}^{I} \lambda_i C_i(t) \tag{1}$$

$$\dot{C}_i(t) = \frac{\beta_i}{\Lambda} T(t) - \lambda_i C_i(t) \quad , i = 1,2,\ldots,I \tag{2}$$

$$T(0) = T_0 \tag{3}$$

$$C_i(0) = \frac{\beta_i}{\Lambda\lambda_i} T_0 \quad , i = 1,2,\ldots,I \tag{4}$$

where,

$T(t)$ = the amplitude function.
$C_i(t)$ = the effective delayed neutron precursor population of the ith delayed group.
$\rho(t)$ = the reactivity (assumed to be a known function of time).
β = the effective delayed neutron fraction.
β_i = the effective delayed neutron fraction of the ith delayed group.
Λ = the prompt neutron lifetime.
λ_i = the decay constant of the ith delayed group.
I = the number of delayed groups.

In order to decouple the stiffness from the I delayed neutron precursor equations an auxiliary function $\omega(t)$, the inverse period, is introduced such that

$$\dot{T}(t) \equiv \omega(t)T(t). \tag{5}$$

With equation (5) used to eliminate $T(t)$ the precursor equations become

$$\dot{C}_i(t) = \frac{\beta_i}{\omega(t)\Lambda + \beta - \rho(t)} \sum_{j=1}^{I} \lambda_j C_j(t) - \lambda_i C_i(t) \quad ,i = 1,2,\ldots,I. \tag{6}$$

Equation (6) can be expressed in more compact matrix notation as

$$\dot{\underline{C}} = \underline{\beta}'\underline{\lambda}^T\underline{C} - \underline{\underline{\lambda}}_d\underline{C} \tag{7}$$

where,

$$\underline{C} \equiv \text{Col}\{C_1, C_2, \ldots, C_I\},$$
$$\underline{\beta}' \equiv \frac{1}{\omega\Lambda+\beta-\rho}\,\text{Col}\{\beta_1, \beta_2, \ldots, \beta_I\},$$
$$\underline{\lambda}^T \equiv \text{Row}\{\lambda_1, \lambda_2, \ldots, \lambda_I\},$$
$$\underline{\underline{\lambda}}_d \equiv \text{Diag}\{\lambda_1, \lambda_2, \ldots, \lambda_I\}.$$

An equivalent form of equation (7) is

$$e^{-\underline{\underline{\lambda}}_d t}\frac{d}{dt}\left[\, e^{\underline{\underline{\lambda}}_d t}\underline{C} \,\right] = \underline{\beta}'\underline{\lambda}^T\underline{C} \,. \tag{8}$$

Multiplying (8) by $e^{\underline{\underline{\lambda}}_d t}$ and integrating over $\Lambda_n \in [t_n, t_{n+1}]$ using a diagonal θ-weighting matrix we get

$$e^{\underline{\underline{\lambda}}_d t_{n+1}}\underline{C}^{n+1} - e^{\underline{\underline{\lambda}}_d t_n}\underline{C}^n = \Lambda_n\underline{\underline{\theta}}_d e^{\underline{\underline{\lambda}}_d t_{n+1}}\underline{\beta}'_{n+1}\underline{\lambda}^T\underline{C}^{n+1} + \Lambda_n(\underline{\underline{I}} - \underline{\underline{\theta}}_d)e^{\underline{\underline{\lambda}}_d t_n}\underline{\beta}'_n\underline{\lambda}^T\underline{C}^n \,. \tag{9}$$

Then, multiplying by $e^{-\underline{\underline{\lambda}}_d t_{n+1}}$ and collecting terms we get,

$$[\underline{\underline{I}} - \Lambda_n\underline{\underline{\theta}}_d\underline{\beta}'_{n+1}\underline{\lambda}^T]\underline{C}^{n+1} = \left[e^{-\underline{\underline{\lambda}}_d\Lambda_n} + \Lambda_n(\underline{\underline{I}} - \underline{\underline{\theta}}_d)e^{-\underline{\underline{\lambda}}_d\Lambda_n}\underline{\beta}'_n\underline{\lambda}^T\right]\underline{C}^n$$
$$= e^{-\underline{\underline{\lambda}}_d\Lambda_n}\left[\underline{\underline{I}} + \Lambda_n(\underline{\underline{I}} - \underline{\underline{\theta}}_d)\underline{\beta}'_n\underline{\lambda}^T\right]\underline{C}^n \,, \tag{10}$$

where, because of its simple structure, the coefficient matrix on the LHS of (10) can be inverted analytically. The resulting expression for $\underline{C}^{n+1}$ is

$$\underline{C}^{n+1} = \left[\underline{\underline{I}} + \frac{\Lambda_n\underline{\underline{\theta}}_d\underline{\beta}'_{n+1}\underline{\lambda}^T}{1 - \Lambda_n\underline{\lambda}^T\underline{\underline{\theta}}_d\underline{\beta}'_{n+1}}\right]e^{-\underline{\underline{\lambda}}_d\Lambda_n}\left[\underline{\underline{I}} + \Lambda_n(\underline{\underline{I}} - \underline{\underline{\theta}}_d)\underline{\beta}'_n\underline{\lambda}^T\right]\underline{C}^n \,. \tag{11}$$

In terms of the previous notation, C_i^{n+1} for i=1,2,...,I, is given by

$$C_i^{n+1} = e^{-\lambda_i \Delta_n} C_i^n + \frac{\Delta_n(1-\theta_i)e^{-\lambda_i\Delta_n}\beta_i}{\omega_{n+1}\Lambda + \beta - \rho_{n+1}} \left[\sum_{j=1}^{I} \lambda_j C_j^n\right]$$

$$+ \frac{\Delta_n \theta_i \beta_i}{(\omega_{n+1}\Lambda+\beta-\rho_{n+1}) - \sum_{j=1}^{I} \Delta_n \lambda_j \theta_j \beta_j} \left[\left[\sum_{j=1}^{I} e^{-\lambda_j\Delta_n}\lambda_j C_j^n\right]\right.$$

$$\left. + \left[\sum_{j=1}^{I} \frac{\Delta_n(1-\theta_j)e^{-\lambda_j\Delta_n}\beta_j\lambda_j}{\omega_n\Lambda + \beta - \rho_n}\right]\left[\sum_{j=1}^{I} \lambda_j C_j^n\right]\right] . \qquad (12)$$

Finally, note that T^{n+1} can be obtained, without approximation, from equations (1) and (5):

$$T^{n+1} = \frac{\Lambda}{\omega_{n+1}\Lambda + \beta - \rho_{n+1}} \sum_{i=1}^{I} \lambda_i C_i^{n+1} . \qquad (13)$$

Two items remain to be addressed: determination of $\omega(t)$ and selection of the θ-weighting coefficients. They will be discussed individually in the next two sections.

2.2 Quadratic $\omega(t)$ Approximation

Assume that over some time interval $\Delta_n \in [t_n, t_{n+1}]$ that $\omega(t)$ can be approximated by a quadratic function of time of the form

$$\omega(t) = \omega_n + b\left[\frac{t-t_n}{\Delta_n}\right] + c\left[\frac{t-t_n}{\Delta_n}\right]^2 \quad , \; t_n \leq t \leq t_{n+1} \qquad (14)$$

where ω_n and the coefficients are given by,

$$\omega_n = \frac{\rho_n - \beta}{\Lambda} + \frac{1}{T^n}\sum_{i=1}^{I} \lambda_i C_i^n , \qquad (15)$$

$$b = \dot{\omega}_n \Delta_n ,$$

$$c = (\omega_{n+1} - \omega_n) - \dot{\omega}_n \Delta_n .$$

From equation (14) we get

$$\dot{\omega}_n + \dot{\omega}_{n+1} = \frac{2}{\Delta_n}(\omega_{n+1} - \omega_n) . \tag{16}$$

A formally exact expression for $\dot{\omega}(t)$ can be derived by differentiating the point kinetics equation [4], the result being

$$\dot{\omega}(t) = -\omega(t)^2 - \left[\lambda_e'(t) + \frac{\beta - \rho(t)}{\Lambda}\right]\omega(t) + \frac{\dot{\rho}(t) + \lambda_e'(t)(\rho(t) - \beta) + \Sigma \lambda_i \beta_i}{\Lambda} \tag{17}$$

where an additional auxiliary function $\lambda_e'(t)$, the instantaneous, effective multi-group decay parameter, is defined to be

$$\lambda_e'(t) \equiv \frac{\sum_{i=1}^{I} \lambda_i^2 C_i(t)}{\sum_{i=1}^{I} \lambda_i C_i(t)} . \tag{18}$$

Combining equation (16) and (17) we get

$$\dot{\omega}_n + \dot{\omega}_{n+1} = \frac{2}{\Delta_n}(\omega_{n+1} - \omega_n) = -\omega_n^2 - \left[\lambda_{e,n}' + \frac{\beta - \rho_n}{\Lambda}\right]\omega_n - \omega_{n+1}^2 - \left[\lambda_{e,n+1}' + \frac{\beta - \rho_{n+1}}{\Lambda}\right]\omega_{n+1} + \frac{\dot{\rho}_n + \dot{\rho}_{n+1} + \lambda_{e,n}'(\rho_n - \beta) + \lambda_{e,n+1}'(\rho_{n+1} - \beta) + 2\Sigma\lambda_i\beta_i}{\Lambda} . \tag{19}$$

Finally, solving for the most positive root of equation (19),

$$\omega_{n+1} = -\left[\frac{\beta-\rho_{n+1}}{2\Lambda} + \frac{\lambda'_{e,n+1}}{2} + \frac{1}{\Delta_n} \right] + \left[\left[\frac{\beta-\rho_{n+1}}{2\Lambda} + \frac{\lambda'_{e,n+1}}{2} + \frac{1}{\Delta_n} \right]^2 + \frac{\dot{\rho}_n + \dot{\rho}_{n+1} + \lambda'_{e,n}(\rho_n-\beta) + \lambda'_{e,n+1}(\rho_{n+1}-\beta) + 2\Sigma\lambda_i\beta_i}{\Lambda} - \omega_n^2 + \left[\frac{2}{\Delta_n} - \lambda'_{e,n} - \frac{\beta - \rho_n}{\Lambda} \right]\omega_n \right]^{1/2} . \tag{20}$$

2.3 θ-Weighting Coefficient Selection

Equation (11) will not hold steady state for an arbitrary choice of $\underline{\underline{\theta}}_d$. Therefore $\underline{\underline{\theta}}_d$ must be defined such that it will hold steady state. Accordingly the weighting coefficient matrix $\underline{\underline{\theta}}_d$ is define to be

$$\underline{\underline{\theta}}_d \equiv \text{Diag}\{\theta_1, \theta_2, \ldots, \theta_I\}$$

where,

$$\theta_i = \frac{1}{\Delta_n\lambda_i} - \frac{e^{-\lambda_i\Delta_n}}{1 - e^{-\lambda_i\Delta_n}} . \tag{21}$$

Substitution of this expression into Equation (12) with $\rho = \omega = 0$ in the definitions (7) of $\underline{\beta}'$ and C_i^n taken as

$$C_i(0) = \frac{\beta_i}{\Lambda\lambda_i} T(0)$$

yields $C_i^{n+1} = C_i(0)$.

2.4 Algorithm

Figure 2.1 presents the DTM algorithm. Note that two options are available. The first, IOMG, controls the method used for the initial estimate of ω_{n+1}. The second, ITER, controls the number of iterations used to converge the solution. The algorithm is discussed in detail below.

The first order of business is bookkeeping. The kinetics parameters from the previous time step are updated and $\dot{\rho}_n$ is determined. If reactivity is a known function of time ρ_{n+1} and $\dot{\rho}_{n+1}$ are calculated. If feedback is significant an initial estimate of ρ_{n+1} and $\dot{\rho}_{n+1}$ is determined.

Next an estimate of ω_{n+1} must be determined. If IOMG = 0, ω_{n+1} is set equal to ω_n. If IOMG = 1, ω_{n+1} is calculated using the quadratic $\omega(t)$ approximation, equation (20), assuming $\lambda'_{e,n+1} = \lambda'_{e,n}$.

The last step is an iteration loop in which C_i^{n+1}, $\lambda'_{e,n+1}$, ω_{n+1}, and T^{n+1} are determined. ITER iterations are performed to converge the solution. Also, if significant feedback is present ρ_{n+1} and $\dot{\rho}_{n+1}$ could be recalculated.

Table 2.1 shows the effects of the options on the algorithm accuracy and speed for a simple problem. If IOMG=0, setting $\omega_{n+1} = \omega_n$, 2-3 iterations are required for convergence. However, for IOMG=1, using the quadratic $\omega(t)$ approximation, no iteration is required. In general, the use of IOMG=1 and ITER=1 will provide the best combination of accuracy and speed. The numerical results referenced in this report were obtained using IOMG=1 and ITER=1.

The FORTRAN code for an implementation of the DTM algoritm, the DTM1D and QOMGD routines, is given in Appendix B.

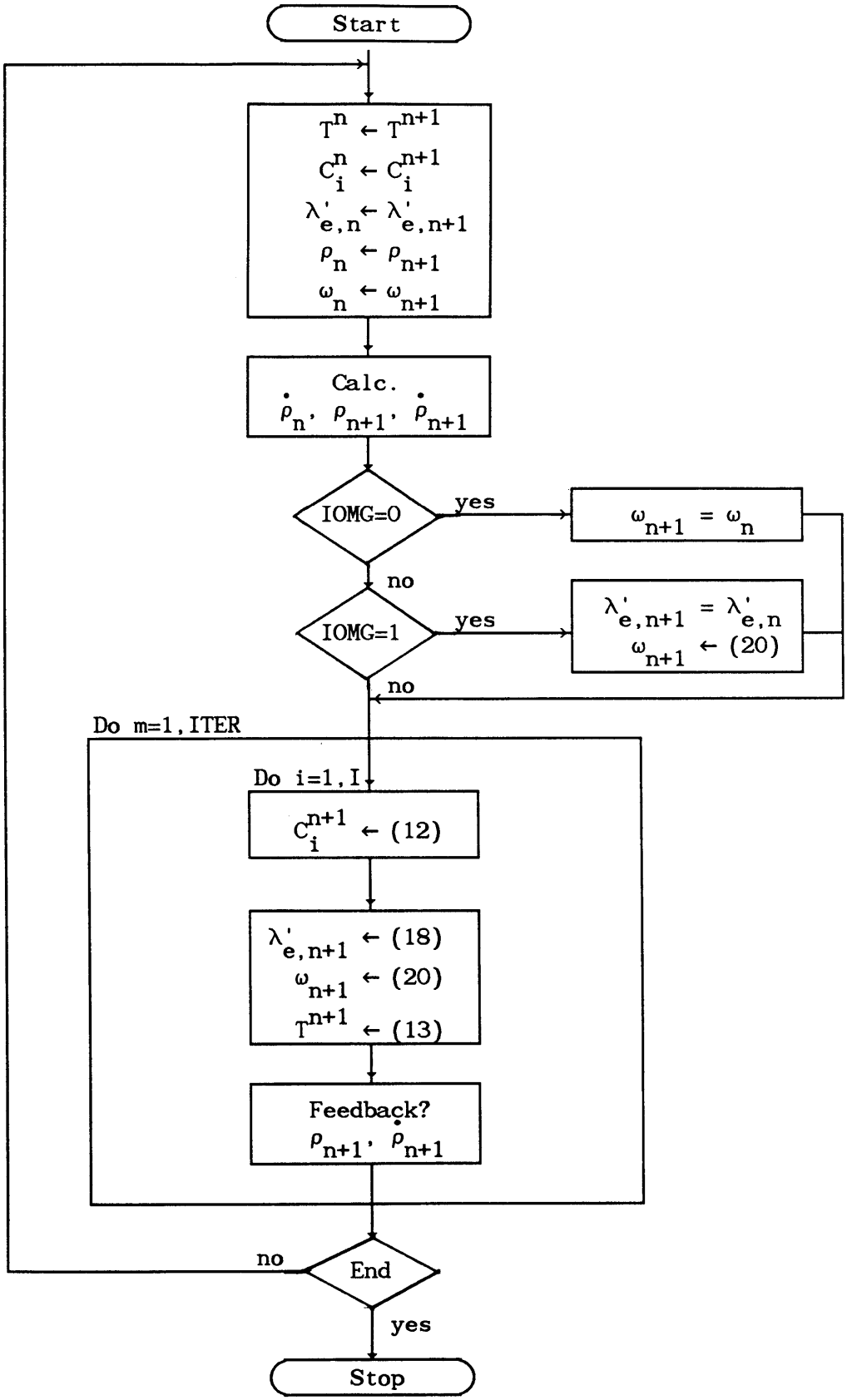

Figure 2.1 Diagonal θ-Weighting Method Algorithm

TABLE 2.1

Effects of DTM Algorithm Options on Accuracy and Speed:
0.1 β/s Ramp Reactivity Insertion

Δt (s)	IOMG	ITER	Amplitude Function, T(t)				Exec.[a] Time (s)
			t=6 s	t=8 s	t=9 s	t=10 s	
10^{-4}	0	1	5.5821	4.2786E+1	4.8752E+2	4.5126E+5	565.6
		2	5.5821	4.2786E+1	4.8752E+2	4.5116E+5	956.4
	1	1	5.5821	4.2786E+1	4.8752E+2	4.5116E+5	658.5
		2	5.5821	4.2786E+1	4.8752E+2	4.5116E+5	1049.2
10^{-2}	0	1	5.5822	4.2793E+1	4.8801E+2	4.6377E+5	5.5
		2	5.5821	4.2789E+1	4.8768E+2	4.5357E+5	9.4
	1	1	5.5821	4.2789E+1	4.8767E+2	4.5354E+5	6.6
		2	5.5821	4.2789E+1	4.8768E+2	4.5357E+5	10.5
10^{-1}	0	1	5.5888	4.3131E+1	5.0836E+2	1.4837E+6	0.5
		2	5.5884	4.3091E+1	5.0423E+2	8.4655E+5	0.8
	1	1	5.5882	4.3083E+1	5.0390E+2	8.4172E+5	0.5
		2	5.5884	4.3091E+1	5.0423E+2	8.4540E+5	1.0

[a]Execution times (cpu sec) for 8 MHz 8086/8087 PC.

Chapter 3

NUMERICAL RESULTS

The DTM has been tested with three types of problems and the results compared against those obtained using the traditional θ-weighting method (see Appendix A for a derivation of the traditional θ-weighting method). The three types of problems examined were a ramp reactivity insertion, positive step reactivity insertions, and negative step reactivity insertions. For all three problems the following kinetics parameters were used:

$\Lambda = 2\text{x}10^{-5}$ s
$\beta = 0.007$
$\beta_i = 0.000266,\ 0.001491,\ 0.001316,\ 0.002849,\ 0.000896,\ 0.000182$
$\lambda_i = 0.0127,\ 0.0317,\ 0.115,\ 0.311,\ 1.4,\ 3.87$ 1/s.

All runs were made using a constant timestep to prevent ambiguity when comparing the accuracy and speed of the numerical methods. The DTM runs were made using the suggested option (IOMG=1 and ITER=1) and the traditional θ-weighting method runs were made using $\theta_p = \theta_d = 0.5$ (Crank-Nickolson). The calculations were performed in double precision on a 8 MHz 8086/8087 PC using MS-DOS 3.1 and IBM Professional FORTRAN.

3.1 Ramp Reactivity Insertion

The first case tested was a 0.1 β/s ramp reactivity insertion covering the range of reactivities from prompt subcritical to prompt supercritical. Tables 3.1 and 3.2 present the amplitude function and inverse period obtained using several timestep sizes. Table 3.1 shows that the accuracy of the DTM is comparable to that of the traditional θ-weighting method, however it is approximately 30% slower. In addition, it can be infered from Table 3.2 that the quadratic ω(t) approximation is both valid and accurate.

3.2 Positive Step Reactivity Insertions

Four positive step reactivity insertions were considered: two prompt subcritical ρ=0.003 and ρ=0.0055, one prompt critical ρ=0.007, and one prompt supercritical ρ=0.008. Tables 3.3 and 3.4 show the amplitude function and inverse period obtained using various timesteps. The DTM has difficulty modeling positive step reactivity insertions, requiring and order of magnitude smaller timestep to provide accuracy comparable to the θ-weighting method. This is due to the need to accurately model rapid exponential decay of the inverse period during the prompt jump. In addition, if a large timestep is used, the quadratic $\omega(t)$ approximation encounters a numerical instability during the initial timestep. Since for the initial timestep

$$\omega_n = \rho(0^+)/\Lambda >> 1,$$

it can clearly be seen from equation (20) that the square root term will become imaginary unless a small timestep is used. For subsequent timesteps this restriction is relaxed due to the rapid decay of ω. Note that this limitation is only present for positive step reactivity insertions.

3.3 Negative Step Reactivity Insertion

Four negative step reactivity insertions were considered: ρ=-0.003, ρ=-0.0055, ρ=-0.007, and ρ=-0.014. Tables 3.5 and 3.6 give the values of the amplitude function and inverse period for various timesteps. In general, the accuracy of the DTM is comparable for the smaller negative steps and superior for the larger negative steps. Note that the use of $\theta_p = 1$ (a fully implicit prompt term) would have eliminated the instability problem that appears at larger timesteps with the traditional θ-weighting method.

TABLE 3.1

Comparison of DTM vs. θ-Weighting: 0.1 β/s Ramp Reactivity Insertion, Amplitude Function

Numerical Method	t (s)	Amplitude Function, T(t)			
		$\Delta t=10^{-4}$s	$\Delta t=10^{-3}$s	$\Delta t=10^{-2}$s	$\Delta t=10^{-1}$s
θ-Weighting	2	1.3382	1.3382	1.3382	1.3382
	4	2.2284	2.2284	2.2284	2.2286 (.01)[a]
	6	5.5821	5.5821	5.5821	5.5837 (.03)
	8	4.2786E+1	4.2786E+1	4.2788E+1	4.2914E+1 (.30)
	9	4.8752E+2	4.8752E+2	4.8762E+2 (.02)	4.9723E+2 (1.99)
	10	4.5116E+5	4.5119E+5 (.01)	4.5331E+5 (.48)	7.9076E+5 (75.27)
	11	1.7922E+16	1.7954E+16 (.18)	2.1464E+16 (19.76)	-5.8778E+14
DTM	2	1.3382	1.3382	1.3382	1.3384 (.01)
	4	2.2284	2.2284	2.2284	2.2292 (.04)
	6	5.5821	5.5821	5.5821	5.5882 (.11)
	8	4.2786E+1	4.2786E+1	4.2789E+1 (.01)	4.3083E+1 (.71)
	9	4.8752E+2	4.8752E+2	4.8767E+2 (.03)	5.0390E+2 (3.43)
	10	4.5116E+5	4.5119E+5 (.01)	4.5354E+5 (.53)	8.4172E+5 (87.38)
	11	1.7922E+16	1.7955E+16 (.18)	2.1534E+16 (20.15)	-5.3993E+14
θ-Weighting[b]		499.5 sec	49.9 sec	4.9 sec	0.4 sec
DTM		658.5 sec	65.9 sec	6.6 sec	0.5 sec

[a]Values in parentheses are % relative error using $\Delta t=10^{-4}$s as reference.

[b]Execution times (cpu sec) for 8 MHz 8086/8087 PC.

TABLE 3.2

Comparison of DTM vs. θ-Weighting: 0.1 β/s Ramp Reactivity Insertion, Inverse Period

Numerical Method	t (s)	Inverse Period, ω(t)			
		$\Delta t=10^{-4}$s	$\Delta t=10^{-3}$s	$\Delta t=10^{-2}$s	$\Delta t=10^{-1}$s
θ-Weighting	2	1.9261E-1	1.9261E-1	1.9261E-1	1.8676E-1 (-3.04)[a]
	4	3.3145E-1	3.3145E-1	3.3145E-1	3.3122E-1 (-.07)
	6	6.3269E-1	6.3269E-1	6.3268E-1	6.3226E-1 (-.07)
	8	1.6508E+0	1.6508E+0	1.6508E+0	1.6465E+0 (-.26)
	9	3.6049E+0	3.6049E+0	3.6046E+1 (-.01)	3.5771E+0 (-.77)
	10	1.2346E+1	1.2346E+1	1.2344E+1 (-.02)	1.2101E+1 (-1.98)
	11	3.8852E+1	3.8851E+1	3.8849E+1 (-.01)	3.8579E+1 (-.70)
DTM	2	1.9261E-1	1.9261E-1	1.9261E-1	1.8275E-1 (-5.12)
	4	3.3145E-1	3.3145E-1	3.3145E-1	3.3071E-1 (-.22)
	6	6.3269E-1	6.3269E-1	6.3269E-1	6.3203E-1 (-.10)
	8	1.6508E+0	1.6508E+0	1.6508E+0	1.6483E+0 (-.15)
	9	3.6049E+0	3.6049E+0	3.6049E+0	3.5984E+0 (-.18)
	10	1.2346E+1	1.2346E+1	1.2346E+1	1.2330E+1 (-.13)
	11	3.8852E+1	3.8852E+1	3.8851E+1	3.8841E+1 (-.03)

[a]Values in parentheses are % relative error using $\Delta t=10^{-4}$s as reference.

TABLE 3.3

Comparison of DTM vs. θ-Weighting:
Positive Reactivity Steps, Amplitude Function

ρ	Method	Δt	Amplitude Function, T(t)		
			t=1 s	t=10 s	t=20 s
.003	θ-Weighting	.001	2.2098	8.0192	2.8297E+1
		.01	2.2098	8.0192	2.8297E+1
		.1	2.2091	8.0193	2.8298E+1
	DTM	.0001	2.2098	8.0192	2.8297E+1
		.001	2.2099	8.0192	2.8298E+1
		.01	2.2155	8.0332	2.8345E+1
			t=.1 s	t=1 s	t=10 s
.0055	θ-Weighting	.001	5.2100	1.5183E+1	1.3886E+5
		.01	5.2106	1.5183E+1	1.3887E+5
		.1	7.2564	1.5180E+1	1.4005E+5
	DTM	.0001	5.2100	1.5183E+1	1.3886E+5
		.001	5.2108	1.5184E+1	1.3888E+5
			t=.01 s	t=.5 s	t=2 s
.007	θ-Weighting	.0001	4.5089	5.3459E+3	2.0592E+11
		.001	4.5089	5.3462E+3	2.0597E+11
		.01	4.5133	5.3813E+3	2.1142E+11
	DTM	.00005	4.5090	5.3459E+3	2.0592E+11
		.0001	4.5093	5.3460E+3	2.0592E+11
		.001	4.5582	5.3563E+3	2.0637E+11
			t=.01 s	t=.1 s	t=1 s
.008	θ-Weighting	.00001	6.2029	1.4104E+3	6.1633E+23
		.0001	6.2029	1.4104E+3	6.1641E+23
		.001	6.2044	1.4122E+3	6.2394E+23
		.01	6.3603	1.6038E+3	2.2187E+24
	DTM	.00001	6.2029	1.4104E+3	6.1633E+23
		.0001	6.2038	1.4106E+3	6.1647E+23
		.001	6.3052	1.4254E+3	6.3017E+23

TABLE 3.4

Comparison of DTM vs. θ-Weighting:
Positive Reactivity Steps, Inverse Period

ρ	Method	Δt	Inverse Period, $\omega(t)$		
			t=1 s	t=10 s	t=20 s
.003	θ-Weighting	.001	1.9099E-1	1.2926E-1	1.2443E-1
		.01	1.9099E-1	1.2926E-1	1.2443E-1
		.1	9.7978E+0	1.2926E-1	1.2443E-1
	DTM	.0001	1.9099E-1	1.2926E-1	1.2443E-1
		.001	1.9098E-1	1.2926E-1	1.2443E-1
		.01	1.9035E-1	1.2925E-1	1.2443E-1
			t=.1 s	t=1 s	t=10 s
.0055	θ-Weighting	.001	1.4692	1.0739	1.0085
		.01	1.4612	1.0739	1.0085
		.1	-2.0654E+1	1.1617	1.0085
	DTM	.0001	1.4693	1.0739	1.0085
		.001	1.4688	1.0739	1.0085
			t=.01 s	t=.5 s	t=2 s
.007	θ-Weighting	.0001	7.8214E+1	1.1648E+1	1.1644E+1
		.001	7.8213E+1	1.1648E+1	1.1644E+1
		.01	7.8137E+1	1.1648E+1	1.1644E+1
	DTM	.00005	7.8212E+1	1.1648E+1	1.1644E+1
		.0001	7.8206E+1	1.1648E+1	1.1644E+1
		.001	7.7380E+1	1.1648E+1	1.1644E+1
			t=.01 s	t=.1 s	t=1 s
.008	θ-Weighting	.00001	1.0701E+2	5.2985E+1	5.2804E+1
		.0001	1.0701E+2	5.2985E+1	5.2804E+1
		.001	1.0699E+2	5.2985E+1	5.2804E+1
		.01	1.0567E+2	5.2963E+1	5.2804E+1
	DTM	.00001	1.0701E+2	5.2985E+1	5.2804E+1
		.0001	1.0700E+2	5.2985E+1	5.2804E+1
		.001	1.0610E+2	5.2983E+1	5.2804E+1

TABLE 3.5

Comparison of DTM vs. θ-Weighting:
Negative Reactivity Steps, Amplitude Function

ρ	Method	Δt	Amplitude Function, T(t)		
			t=1 s	t=10 s	t=20 s
-.003	θ-Weighting	.01	.63896	.43441	.33486
		.1	.77358	.43451	.33486
	DTM	.001	.63896	.43441	.33486
		.01	.63905	.43444	.33487
		.1	.71754	.43646	.33596
			t=.1 s	t=2 s	t=10 s
-.0055	θ-Weighting	.001	.55023	.44781	.28522
		.01	.55081	.44781	.28522
		.1	.13781	.57005	.28595
	DTM	.001	.55024	.44781	.28522
		.01	.55063	.44792	.28526
		.1	.46533	.48706	.28722
			t=.01 s	t=.5 s	t=2 s
-.007	θ-Weighting	.0001	.49968	.45839	.38820
		.001	.49956	.45839	.38820
		.01	.22157	.45839	.38820
	DTM	.0001	.49968	.45839	.38820
		.001	.49962	.45839	.38820
		.01	.46316	.45860	.38831
			t=.01 s	t=.5 s	t=2 s
-.014	θ-Weighting	.0001	.33257	.29687	.23871
		.001	.33256	.29687	.23871
		.01	-.12068	.29687	.23871
	DTM	.0001	.33257	.29687	.23871
		.001	.33257	.29688	.23871
		.01	.30961	.29710	.23882

TABLE 3.6

Comparison of DTM vs. θ-Weighting:
Negative Reactivity Steps, Inverse Period

ρ	Method	Δt	Inverse Period, ω(t)		
			t=1 s	t=10 s	t=20 s
-.003	θ-Weighting	.01	-7.0854E-2	-3.1211E-2	-2.2480E-2
		.1	-8.7137E+1	-1.4701E-1	-2.2530E-2
	DTM	.001	-7.0855E-2	-3.1211E-2	-2.2480E-2
		.01	-7.0905E-2	-3.1214E-2	-2.2481E-2
		.1	-5.0213E+1	-6.4822E-2	-2.2557E-2
			t=.1 s	t=2 s	t=10 s
-.0055	θ-Weighting	.001	-1.7283E-1	-8.0956E-2	-4.2051E-2
		.01	-8.3139E-1	-8.0956E-2	-4.2051E-2
		.1	1.8704E+3	-1.3415E+2	-1.6397E+0
	DTM	.001	-1.7283E-1	-8.0957E-2	-4.2051E-2
		.01	-3.0459E-1	-8.0997E-2	-4.2057E-2
		.1	1.1863E+2	-4.2573E+1	-2.7811E-1
			t=.01 s	t=.5 s	t=2 s
-.007	θ-Weighting	.0001	-8.5093E-1	-1.4503E-1	-9.1133E-2
		.001	-6.8347E-1	-1.4503E-1	-9.1133E-2
		.01	8.7699E+2	-1.4503E-1	-9.1133E-2
	DTM	.0001	-8.5171E-1	-1.4503E-1	-9.1133E-2
		.001	-7.5026E-1	-1.4504E-1	-9.1134E-2
		.01	5.4802E+1	-1.4530E-1	-9.1184E-2
			t=.01 s	t=.5 s	t=2 s
-.014	θ-Weighting	.0001	-3.4479E-1	-1.9223E-1	-1.1832E-1
		.001	-3.0554E-1	-1.9223E-1	-1.1832E-1
		.01	-3.9433E+3	-1.9224E-1	-1.1832E-1
	DTM	.0001	-3.4500E-1	-1.9223E-1	-1.1832E-1
		.001	-3.1581E-1	-1.9224E-1	-1.1832E-1
		.01	7.8767E+1	-1.9268E-1	-1.1840E-1

Chapter 4

CONCLUSIONS AND RECOMMENDATIONS

Stiffness decoupling methods do not provide a significant improvement in accuracy, speed or stability compared to the traditional θ-weighting method using Crank-Nickolson weighting. This can be clearly seen in Table 4.1 where three stiffness decoupling methods developed at MIT are compared to the traditional θ-weighting method. Even Parlos' [2] implementation of the Stiffness Confinement Method (SCM) [3] is outperformed by traditional θ-weighting (note that the SCM is 3 times slower than traditional θ-weighting).

As a result, further development of stiffness decoupling methods should be discontinued. Development of timestep selection rules for the traditional θ-weighting method should be pursued instead.

TABLE 4.1

Comparison of Traditional θ-Weighting vs Stiffness Decoupling Methods: 0.1 β/s Ramp Reactivity Insertion, Δt = 0.1 s

t (s)	Amplitude Function, T(t)			
	θ-Weighting	DTM	Parlos' SDM[a]	Parlos' SCM[b]
2	1.3382	1.3384(.015)[c]	1.3386(.030)	1.3382
4	2.2286(.013)	2.2292(.036)	2.2311(.126)	2.2286(.013)
6	5.5837(.039)	5.5882(.120)	5.6157(.613)	5.5846(.056)
8	42.914 (.311)	43.083 (.706)	45.024 (5.24)	42.996 (.503)
$\frac{\text{CPU sec}}{\text{Real sec}}$[d]	0.036	0.045	0.045	0.11

[a] A stiffness decoupling method (Method #1) developed by Parlos [2].

[b] Parlos' implementation of the Stiffness Confinement Method (Method #2) [2].

[c] Values in parentheses are % relative error using $\Delta t = 10^{-4}$ s as reference.

[d] Equivalent execution time: double precision, 8 MHz 8086/8087.

Chapter 5

REFERENCES

1. Y. A. Chao and A. Attard, "A Resolution of the Stiffness Problem of Reactor Kinetics," *Nuclear Science and Engineering*, **90**, 40 (1985).

2. A. G. Parlos, "Non-Linear Control of the Power Overshoot and Steady State Error for a Tightly Coupled Core with no Significant T-H Feedback," Special Problem, MIT Course 22.902, August, 1985.

3. A. Z. Tanker, "Comparison of Three Non-Stiff Numerical Methods to Solve Point Kinetics Equations," Special Problem, MIT Course 22.902, July, 1986.

4. J. A. Bernard, A. F. Henry and D. D. Lanning, "The Design and Experimental Evaluation of a Closed-Loop Digital Controller Based on an Alternate Formulation of the Dynamic Period Equation," *Proceedings of the American Nuclear Society Topical Meeting on Reactor Physics and Safety*, Saratoga Springs, NY, September, 1986, pp. 610-621.

5. Prof. Allan F. Henry, personal communications.

6. T. A. Porsching and A. F. Henry, "Some Numerical Methods for Problems in Reactor Kinetics," WAPD-T-2130 (February 1968).

7. T. A. Porsching,"The Numerical Solution of the Reactor Kinetics Equations by Difference Analogs: A Comparison of Methods," WAPD-TM-564 (March 1966).

Appendix A

DERIVATION OF THE TRADITIONAL θ-WEIGHTING METHOD

The traditional or standard θ-weighting method is a single-step, semi-implicit numerical method for solving the point kinetics equations [5,6]. The θ's are weighting coefficients used to specify the contribution of the explicit and implicit terms to be used in the solution. Two θ's are used: one for the prompt response associated with the amplitude function (θ_p) and the second for the delayed response associated with the delayed neutron precursors (θ_d). The value of the θ's range from zero, indicating a fully explicit solution, to unity, indicating a fully implicit solution.

The derivation of the traditional θ-weighting method starts with the point kinetics equations:

$$\frac{dT(t)}{dt} = \frac{\rho(t)-\beta}{\Lambda} T(t) + \sum_{i=1}^{I} \lambda_i C_i(t) \qquad \text{(A-1)}$$

$$\frac{dC_i(t)}{dt} = \frac{\beta_i}{\Lambda} T(t) - \lambda_i C_i(t) \qquad ,i=1,2,\ldots,I \qquad \text{(A-2)}$$

where the kinetics parameters are defined in section 2.1. The dynamic behavior of the delayed neutron precursors is determined by approximating equation (A-2) using the following difference analog over some small time interval Δ_n ($\Delta_n = t_{n+1} - t_n$),

$$\frac{C_i^{n+1} - C_i^n}{\Delta_n} = \theta_d \left[\frac{\beta_i}{\Lambda} T^{n+1} - \lambda_i C_i^{n+1}\right] + (1-\theta_d)\left[\frac{\beta_i}{\Lambda} T^n - \lambda_i C_i^n\right] \qquad ,i=1,2,\ldots,I \; . \qquad \text{(A-3)}$$

Solving for C_i^{n+1} we get,

$$C_i^{n+1} = \frac{\frac{\beta_i}{\Lambda}\Delta_n\left[\theta_d T^{n+1} + (1-\theta_d)T^n\right] + \left[1 - (1-\theta_d)\Delta_n\lambda_i\right]C_i^n}{1 + \theta_d\Delta_n\lambda_i} . \qquad \text{(A-4)}$$

The dynamic behavior of the amplitude function is obtained using a similar difference analog to approximate equation (A-1),

$$\frac{T^{n+1} - T^n}{\Delta_n} = \theta_p\left[\left[\frac{\rho_{n+1}-\beta}{\Lambda}\right]T^{n+1} + \sum_{i=1}^{I}\lambda_i C_i^{n+1}\right] + (1-\theta_p)\left[\left[\frac{\rho_n-\beta}{\Lambda}\right]T^n + \sum_{i=1}^{I}\lambda_i C_i^n\right] . \qquad \text{(A-5)}$$

The expression for T^{n+1} is obtained by substituting equation (A-4) into (A-5) and rearranging:

$$T^{n+1} = \left[\left[\Lambda + (1-\theta_p)\Delta_n(\rho_n-\beta) + \theta_p\Delta_n\sum_{i=1}^{I}\frac{(1-\theta_d)\Delta_n\beta_i\lambda_i}{1+\theta_d\Delta_n\lambda_i}\right]T^n + \theta_p\Delta_n\Lambda\sum_{i=1}^{I}\frac{[1-(1-\theta_d)\Delta_n\lambda_i]\lambda_i C_i^n}{1+\theta_d\Delta_n\lambda_i} + (1-\theta_p)\Delta_n\Lambda\sum_{i=1}^{I}\lambda_i C_i^n\right] \cdot \left[\Lambda - \theta_p\Delta_n(\rho_{n+1}-\beta) - \theta_p\Delta_n\sum_{i=1}^{I}\frac{\theta_d\Delta_n\beta_i\lambda_i}{1+\theta_d\Delta_n\lambda_i}\right]^{-1} . \qquad \text{(A-6)}$$

Equations (A-6) and (A-4) form a complete system of equations assuming ρ is a known function of time.

A flowchart for the traditional θ-weighting method algorithm is given in Figure A.1. One should immediately note the simple structure of the algorithm. Following some simple bookkeeping the first order of business is to determine the reactivity at the end of the timestep (ρ_{n+1}). Next the amplitude function is updated using equation (A-6). Finally, the delayed neutron precursors are updated using equation (A-4) and the calculation proceeds to the next timestep. An implementation of

the traditional θ-weighting algorithm is given by the THETAM routine in Appendix B.

Finally, comment on θ-weighting coefficient selection would be appropriate. In general three strategies are recommended:

1. $\theta_p = 0.5,\ \theta_d = 0.5$
2. $\theta_p = 1.0,\ \theta_d = 0.5$
3. $\theta_p = f(\omega_p,\ \Delta t),\ \theta_d = (\omega_d,\ \Delta t)$

In the first case both the prompt and delayed equations receive equal contributions from their explicit and implicit terms (Crank-Nickolson weighting). In the second case the amplitude function is determined using a fully-implicit scheme, which has the benefit of being unconditionally stable, however the accuracy of the scheme tends to be inferior. In the final case the weighting coefficients are determined using some complicated function, as Porsching [7] suggests in his XITE-4 code. Crank-Nickolson weighting ($\theta_p = \theta_d = 0.5$) appears to provide the best combination of accuracy, speed, and stability for positive reactivity insertions.

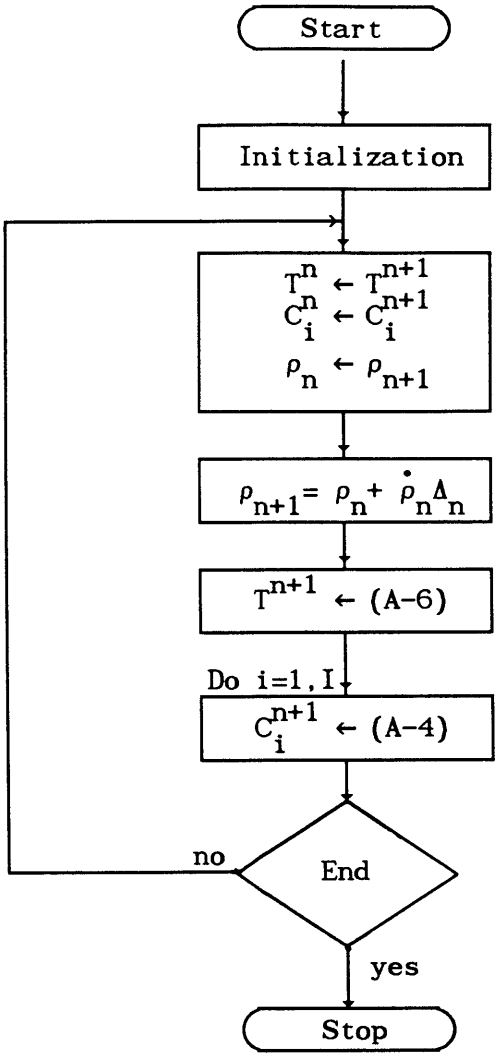

Figure A.1 Traditional θ-Weighting Method Algorithm

Appendix B

DTMTST PROGRAM

B.1 Input Description

The input for a DTMTST case consists of 1 Title Card and NSTEP Data Cards. The Title Card (Card Type 1) consists of a integer variable, which indicates the number of Data Cards in the case, and a descriptive title. The Data Card (Card Type 2) contains the data needed to specify the numerical method and reactivity profile during a portion of the case. There are NSTEP Data Cards in a case, and cases may be stacked. Execution will continue until the last case in the input deck is executed (until a EOF is encountered).

Card Type 1: Title Card

Name	Column	Format	Description
NSTEP	1	I1	Number of steps (Data Cards) in this case.
TITLE	2-80	19A4,A3	Descriptive case title.

Card Type 2: Data Card (Format Free Input)

Name	Item	Description
INOP(1)	1	Number of neutronic timesteps per print edit ($\geq$ 1).
INOP(2)	2	Number of print edits in this step ($\geq$ 1).
INOP(3)	3	Method to be used. =0 Theta Method (THETAM). =1 Diagonal Theta Method, Version 1.0 (DTM1D).
INOP(4)	4	ω_{n+1} option in the DTM1D routine (IOMG). =0 $\omega_{n+1} = \omega_n$. =1 ω_{n+1} determined using the quadratic $\omega(t)$ approximation assuming $\lambda'_{e,n+1} = \lambda'_{e,n}$.
INOP(5)	5	Number of iterations used in DTM1D (ITER).
INOP(6)	6	Reactivity insertion mode flag. =0 Ramp reactivity insertion. =1 Step reactivity insertion.
INOP(7)	7	Number of lines printed per page.
DELN	8	Neutronic timestep [sec].
RAMP	9	Reactivity Insertion. INOP(6) = 0 Ramp rate [1/sec]. INOP(6) = 1 Step (final value of reactivity after step).
THETAP	10	Prompt θ-weighting coefficient for THETAM routine ($0.0 \leq \theta \leq 1.0$).
THETAD	11	Delayed θ-weighting coefficient for THETAM routine. ($0.0 \leq \theta \leq 1.0$).

B.2 Source Listing

This section lists the FORTRAN-77 source code for the DTMTST program.

```
      PROGRAM DTMTST
C*******************************************************************************
C
C TITLE:
C
C      Diagonal Theta-weighting Method Test (DTMTST)
C
C      Copyright (c) 1987 by Massachusetts Institute of Technology
C
C AUTHOR:
C
C      Michael L. Zerkle
C      MIT, Room 38-181
C      (617)253-0945
C
C PURPOSE:
C
C      The DTMTST program was developed to test the diagonal theta-weighting
C method algorithm, a new numerical method for solving the point kinetics
C equations, and benchmark it against a traditional theta-weighting method
C algorithm.
C
C INPUT DESCRIPTION:
C
C      A DTMTST case consists of:  1 Title Card (Card Type 1) and NSTEP
C Data Cards (Card Type 2).  The contents of each card type are given below.
C Several cases may be stacked.  Execution will continue until an EOF is
C encountered.
C
C   CARD TYPE 1:  Title Card
C
C      COLUMN   FORMAT   NAME
C
C      1        I1       NSTEP
C      2-80     A79      TITLE
C
C   CARD TYPE 2:  Data Card (Format Free Input)
C
C      ITEM     NAME             NOTE
C
C       1       INOP(1)
C       2       INOP(2)
C       3       INOP(3)
C       4       INOP(4)
C       5       INOP(5)
C       6       INOP(6)
C       7       INOP(7)
C       8       DELN
C       9       RAMP
C      10       THETAP           If INOP(3)=1 enter any value.
C      11       THETAD           If INOP(3)=1 enter any value.
C
C FILE(S):
C      UNIT     NAME             [R/W]   USAGE
C
C      5        D:DTMTST.DAT     R       System Input.
C      6        D:DTMTST.OUT     W       System Output.
C
C KEY LOCAL VARIABLES:
C      NAME     T*L      A       DESCRIPTION
C
C      B        R*8      25      Effective delayed neutron fraction of the
C                                 ith delayed group.
C      BT       R*8              Total effective delayed neutron fraction.
C      CN       R*8      25      Effective delayed neutron precursor population
C                                 at time n (beginning of timestep).
C      CN1      R*8      25      Effective delayed neutron precursor population.
C      DELN     R*8              Timestep size, DELN=t(n+1)-t(n) [s].
C      I        I*4              Delayed group index.
C      INOP     I*4      7       Input options.
C                                 INOP(1) # of neutronic timesteps per print
C                                         edit (>=1).
```

```
C                                  INOP(2) # of print edits in this step (>=1).
C                                  INOP(3) Point kinetics method to be used.
C                                    =0    Theta Method (THETAM).
C                                    =1    Diagonal Theta Method, (DTM1D).
C                                  INOP(4) Omega option for DTM1D routine (IOMG).
C                                    =0    OMGN1 = OMGN.
C                                    =1    OMGN1 determined using the quadratic
C                                          omega approximation, assuming
C                                          LEN1 = LEN.
C                                  INOP(5) Number of iterations to be used in
C                                          DTM1D, INOP(5)=ITER (>=1).
C                                  INOP(6) Reactivity insertion mode flag.
C                                    =0    Ramp reactivity insertion.
C                                    =1    Step reactivity insertion.
C                                  INOP(7) # lines/page, edit control option.
C      IOMG     I*4               IOMG = INOP(4).
C      ITER     I*4               ITER = INOP(5).
C      J        I*4               Timestep index.
C      K        I*4               Print edit index.
C      L        R*8      25       Decay constant of the ith delayed group [1/s].
C      LEN      R*8               Effective, multi-group decay parameter
C                                  @ t(n) [1/s].
C      LEN1     R*8               Effective, multi-group decay parameter
C                                  @ t(n+1) [1/s].
C      LP       R*8               Prompt neutron lifetime [s].
C      N        I*4               Line counter.
C      NDNPG    I*4               Number of delayed neutron precursor groups to
C                                  be used in this calculation.
C      NN       I*4               Step index.
C      NSTEP    I*4               Number of steps (Data Cards to be read) in
C                                  this case.  Each step corresponds to a
C                                  different reactivity profile.
C      OMGN     R*8               Inverse period @ t(n).
C      OMGN1    R*8               Inverse period @ t(n+1).
C      PN       R*8               Amplitude function @ t(n).
C      PN1      R*8               Amplitude function @ t(n+1).
C      RAMP     R*8               Reactivity Insertion.
C                                  INOP(6)=0  Ramp rate [1/s].
C                                  INOP(6)=1  Step (final value of reactivity
C                                             after step change).
C      RHODN    R*8               Differential reactivity @ t(n) [1/s].
C      RHODN1   R*8               Differential reactivity @ t(n+1) [1/s].
C      RHON     R*8               Reactivity @ t(n).
C      RHON1    R*8               Reactivity @ t(n+1).
C      SBL      R*8               Sum B(i)*L(i), i=1,NDNPG [1/s].
C      SL2CN    R*8               Weighted delayed neutron source @ t(n)
C                                  [s**-2].
C      SL2CN1   R*8               Weighted delayed neutron source @ t(n+1)
C                                  [s**-2].
C      SLCN     R*8               Delayed neutron source @ t(n) [1/s].
C      SLCN1    R*8               Delayed neutron source @ t(n+1) [1/s].
C      THETAD   R*8               Delayed weighting coefficient used in THETAM.
C                                  THETAD=0 fully explicit.  THETAD=1 fully
C                                  implicit. (0 <= THETAD <= 1).
C      THETAP   R*8               Prompt weighting coefficient used in THETAM.
C                                  THETAP=0 fully explicit.  THETAP=1 fully
C                                  implicit.  (0 <= THETAP <= 1).
C      THETD    R*8      25       Diagonal theta-weighting matrix used in DTM1D.
C                                  THETD(i)=0 fully explicit.  THETD(i)=1 fully
C                                  implicit.  (0 <= THETD(i) <= 1).
C      TIME     R*8               Simulated time [s].
C      TITLE    C*79              Case title printed at top of each page.
C
C ROUTINES/FUNCTIONS CALLED:
C      NAME     DESCRIPTION
C
C      DTM1D    Solves the point kinetics equations using the Diagonal
C                Theta-weighting Method, (DTM).
C      THETAM   Solves the point kinetics equations using the traditional
C                theta-weighting method.
C      TIMER    Calculates and edits the execution time.
C
```

```
C COMMON BLOCKS REFERENCED:
C      NAME    USAGE
C
C      DTMD    Diagonal Theta-weighting Matrix and control options for the
C               DTM1D routine.
C      KINDAT  Kinetics parameters.
C      THETA   Theta-weighting coefficients.
C
C RESTRICTIONS/SPECIAL CONSIDERATIONS:
C
C      DTMTST is currently programmed to read and write the system input
C and system output files to a RAMDISK.  This was done to minimize I/O
C time since I/O to a RAMDISK is considerablely faster than I/O to a
C hard disk or floppy disk.  If insufficient RAM or a RAMDISK is not
C available the user can easily change to fixed drive I/O my modifing
C the drive specification in the OPEN statements.
C
C CHANGE HISTORY:
C      DATE    PROGRAMMER      DESCRIPTION
C
C      9/25/87 M. L. Zerkle    Initial Release
C
C*******************************************************************************
C
      COMMON/KINDAT/TIME,DELN,PN,PN1,OMGN,OMGN1,RHON,RHON1,RHODN,RHODN1,
     1              LEN,LEN1,BT,LP,B(25),L(25),SBL,CN(25),CN1(25),SLCN,
     2              SLCN1,SL2CN,SL2CN1,NDNPG
      REAL*8 TIME,DELN,PN,PN1,OMGN,OMGN1,RHON,RHON1,RHODN,RHODN1,LEN
      REAL*8 LEN1,BT,LP,B,L,SBL,CN,CN1,SLCN,SLCN1,SL2CN,SL2CN1
      INTEGER*4 NDNPG
      COMMON/DTMD/THETD(25),ITER,IOMG
      REAL*8    THETD
      INTEGER*4 ITER,IOMG
      COMMON/THETA/THETAP,THETAD
      REAL*8 THETAP,THETAD
      CHARACTER*79 TITLE
      INTEGER*4 I,J,K,N,NN,NSTEP,INOP(7)
      REAL*8 RAMP
C
      OPEN(5,FILE='D:DTMTST.DAT',STATUS='OLD')
      OPEN(6,FILE='D:DTMTST.OUT',STATUS='UNKNOWN')
C
      SBL = 0.0D0
      DO 10 I=1,NDNPG
        SBL = B(I)*L(I) + SBL
   10 CONTINUE
C
C Start New Case.
C
   20 READ(5,900,END=200) NSTEP,TITLE
      TIME  = 0.0D0
      PN    = 1.0D0
      PN1   = 1.0D0
      SLCN  = 0.0D0
      SL2CN = 0.0D0
      DO 30 I=1,NDNPG
        CN(I)    = PN*B(I)/(L(I)*LP)
        CN1(I)   = CN(I)
        SLCN     = CN(I)*L(I) + SLCN
        SL2CN    = CN(I)*L(I)**2 + SL2CN
   30 CONTINUE
      SLCN1  = SLCN
      SL2CN1 = SL2CN
      LEN  = SL2CN/SLCN
      LEN1 = LEN
      RHON  = 0.0D0
      RHODN = 0.0D0
      RHON1 = 0.0D0
      RHODN1= 0.0D0
C
C Edit Title and Header.
C
```

```
      N = 4
      WRITE(6,901) TITLE
      WRITE(6,902)
      WRITE(6,903)
      CALL TIMER(0)
C
C Start Step Loop.
C
      DO 70 NN=1,NSTEP
        READ(5,*,END=200) (INOP(I),I=1,7),DELN,RAMP,THETAP,THETAD
        IOMG = INOP(4)
        ITER = INOP(5)
C
C   Initialize reactivity and omega.
C
        IF (INOP(6) .EQ. 0) THEN
          RHODN  = RAMP
          RHODN1 = RAMP
        ELSE IF (INOP(6) .EQ. 1) THEN
          RHON  = RAMP
          RHON1 = RAMP
          RAMP  = 0.0D0
          OMGN1 = (RHON1-BT)/LP + SLCN1/PN1
        ENDIF
C
C   If beginning of case edit initial condition.
C
        IF (TIME .EQ. 0.0D0) THEN
          N = N + 1
          OMGN = (RHON-BT)/LP + SLCN/PN
          OMGN1 = OMGN
          WRITE(6,904) TIME,PN,RHON,OMGN,SLCN,LEN
        ENDIF
C
C   Begin Step Loop.
C   Start Edit Loop.
C
        DO 60 K=1,INOP(2)
C
C     Neutronic Loop.
C
          CALL TIMER(1)
C
C       Bookkeeping.
C
          DO 50 J=1,INOP(1)
            PN     = PN1
            OMGN   = OMGN1
            RHON   = RHON1
            RHON1  = RHON + RAMP*DELN
            RHODN  = RHODN1
            RHODN1 = RAMP
            LEN    = LEN1
            SLCN   = SLCN1
            SL2CN  = SL2CN1
            DO 40 I=1,NDNPG
              CN(I) = CN1(I)
   40       CONTINUE
C
C       Select solution method.
C
            IF (INOP(3) .EQ. 0) THEN
              CALL THETAM
            ELSE IF (INOP(3) .EQ. 1) THEN
              CALL DTM1D
            ENDIF
C
   50     CONTINUE
          CALL TIMER(2)
C
C     Edit results.
C
```

```
          N = N + 1
          IF (N .GT. INOP(7)) THEN
            N = 5
            WRITE(6,901) TITLE
            WRITE(6,902)
            WRITE(6,903)
          ENDIF
          WRITE(6,904) TIME,PN1,RHON1,OMGN1,SLCN1,LEN1
   60   CONTINUE
   70 CONTINUE
C
C Print case execution time, then start another case.
C
      CALL TIMER(3)
      GOTO 20
C
C Exit.
C
  200 CONTINUE
      CLOSE(5)
      CLOSE(6)
C
  900 FORMAT(I1,A79)
  901 FORMAT('1',5X,A79)
  902 FORMAT('0',5X,' TIME',',   AMPLITUDE',',  REACTIVITY',',    OMEGA   '
     1      ,'   SUM(LC)  ',',     LE''    ')
  903 FORMAT(' ')
  904 FORMAT(' ',3X,F7.4,5(1PE12.4))
C
      STOP
      END
```

```
      BLOCK DATA
C*************************************************************************
C
C TITLE:
C
C      Block Data
C
C AUTHOR:
C
C      Michael L. Zerkle
C      MIT, Room 38-181
C      (617)253-0945
C
C PURPOSE:
C
C      Initialization of kinetics parameters.
C
C KEY LOCAL VARIABLES:
C      NAME    T*L     A       DESCRIPTION
C
C      B       R*8     25      Effective delayed neutron fraction of the
C                               ith delayed group.
C      BT      R*8             Total effective delayed neutron fraction.
C      L       R*8     25      Decay constant of the ith delayed group [1/s].
C      LP      R*8             Prompt neutron lifetime [s].
C      NDNPG   I*4             Number of delayed neutron precursor groups to
C
C COMMON BLOCKS REFERENCED:
C      NAME    USAGE
C
C      KINDAT  Kinetics parameters.
C
C CHANGE HISTORY:
C      DATE    PROGRAMMER      DESCRIPTION
C
C      9/25/87 M. L. Zerkle    Initial Release
C
C*************************************************************************
C
      COMMON/KINDAT/TIME,DELN,PN,PN1,OMGN,OMGN1,RHON,RHON1,RHODN,RHODN1,
     1              LEN,LEN1,BT,LP,B(25),L(25),SBL,CN(25),CN1(25),SLCN,
     2              SLCN1,SL2CN,SL2CN1,NDNPG
      REAL*8 TIME,DELN,PN,PN1,OMGN,OMGN1,RHON,RHON1,RHODN,RHODN1,LEN
      REAL*8 LEN1,BT,LP,B,L,SBL,CN,CN1,SLCN,SLCN1,SL2CN,SL2CN1
      INTEGER*4 NDNPG
      DATA BT,LP/7.0D-3,2.0D-5/
      DATA B/2.66D-4,1.491D-3,1.316D-3,2.849D-3,8.96D-4,1.82D-4,
     1       19*0.0D0/
      DATA L/1.27D-2,3.17D-2, 1.15D-1, 3.11D-1, 1.40D+0,3.87D+0,
     1       19*0.0D0/
      DATA NDNPG/6/
      END
```

```
      SUBROUTINE DTM1D
C*************************************************************************
C
C TITLE:
C
C      Diagonal Theta-weighting Method:  Version 1.0, Double Precision (DTM1D)
C
C      Copyright (c) 1987 by Massachusetts Institute of Technology
C
C AUTHOR:
C
C      Michael L. Zerkle
C      MIT, Room 38-181
C      (617)253-0945
C
C PURPOSE:
C
C      This routine is an implementation of the Diagonal Theta-weighting
C Method algorithm, a numerical method for solving the point kinetics
C equations.  It is in a general, modular form where given the kinetics
C parameters at the beginning of the timestep it will calculate and return
C the kinetics parameteres at the end of the timestep.  All floating point
C calculations are performed in Double Precision.
C
C METHOD:
C
C      The Diagonal Theta-weighting Method is a stiffness decoupled
C numerical method for solving the point kinetics equations.  It is based
C on the observation that the stiffness characteristic is present only in
C the time response of the amplitude function and not in that of the delayed
C neutron precursors.  In order to decouple the stiffness from the delayed
C neutron precursor equations the inverse period is introduced, eliminating
C the amplitude function from the equation.  The inverse period is determined
C by assuming that within a small time interval it is a quadratic function
C of time.  The precursor equations are then solved using a difference
C analog with an inverse period determined using the quadratic omega
C approximation.  The weighting coefficients used in the difference analog
C are selected such that they force the method to hold steady state.
C
C      A general version of the algorithm, using a maximum of 25 delayed
C groups, is implemented here.  The terms dependent upon the timestep size
C are recalculated only if the timestep size changes.  Locations for adding
C feedback logic are identified.
C
C CALLING SEQUENCE:
C
C      CALL DTM1D
C
C ARGUMENT(S):
C      NAME    I/O     T*L     A       DESCRIPTION
C
C      NONE.
C
C KEY LOCAL VARIABLES:
C      NAME    T*L     A       DESCRIPTION
C
C      B       R*8     25      Effective delayed neutron fraction of the
C                               ith delayed group.
C      BT      R*8             Total effective delayed neutron fraction.
C      CN      R*8     25      Effective delayed neutron precursor population
C                               at time n (beginning of timestep).
C      CN1     R*8     25      Effective delayed neutron precursor population.
C      DELN    R*8             Timestep size, DELN=t(n+1)-t(n) [s].
C      DELT    R*8             Previous timestep size [s].
C      ELDN    R*8     25      ith group decay term.
C      ELDN2   R*8     25      ith group weighted decay term.
C      ETBE    R*8     25      Explicit ith group weighted decay term.
C      I       I*4             Delayed group index.
C      IOMG    I*4             Option for initial estimate of OMGN1.
C                               =0  OMGN1 = OMGN.
C                               =1  OMGN1 determined using the quadratic
C                                   omega approximation, assuming LEN1=LEN.
```

```
C       ITB     R*8     25      Implicit ith group production term.
C       ITER    I*4             Number of iterations to be used (>=1).
C       L       R*8     25      Decay constant of the ith delayed group [1/s].
C       LEN     R*8             Effective, multi-group decay parameter
C                                @ t(n) [1/s].
C       LEN1    R*8             Effective, multi-group decay parameter
C                                @ t(n+1) [1/s].
C       LP      R*8             Prompt neutron lifetime [s].
C       M       I*4             Iteration index.
C       NDNPG   I*4             Number of delayed neutron precursor groups to
C                                be used in this calculation.
C       OMGN    R*8             Inverse period @ t(n).
C       OMGN1   R*8             Inverse period @ t(n+1).
C       PN      R*8             Amplitude function @ t(n).
C       PN1     R*8             Amplitude function @ t(n+1).
C       PTN     R*8             Prompt term @ t(n).
C       PTN1    R*8             Prompt term @ t(n+1).
C       RHODN   R*8             Differential reactivity @ t(n) [1/s].
C       RHODN1  R*8             Differential reactivity @ t(n+1) [1/s].
C       RHON    R*8             Reactivity @ t(n).
C       RHON1   R*8             Reactivity @ t(n+1).
C       SBL     R*8             Sum B(i)*L(i), i=1,NDNPG [1/s].
C       SETBL   R*8             Sum of explicit terms.
C       SITBL   R*8             Sum of implicit terms.
C       SL2CN   R*8             Weighted delayed neutron source @ t(n)
C                                [s**-2].
C       SL2CN1  R*8             Weighted delayed neutron source @ t(n+1)
C                                [s**-2].
C       SLCN    R*8             Delayed neutron source @ t(n) [1/s].
C       SLCN1   R*8             Delayed neutron source @ t(n+1) [1/s].
C       SUM1    R*8             DTM source term.
C       TEMP1   R*8             Temporary variable.
C       TEMP2   R*8             Temporary variable.
C       THETD   R*8     25      Diagonal theta-weighting matrix used in DTM1D.
C                                THETD(i)=0 fully explicit.  THETD(i)=1 fully
C                                implicit.  (0 <= THETD(i) <= 1).
C       TIME    R*8             Simulated time [s].
C
C ROUTINES/FUNCTIONS CALLED:
C       NAME    DESCRIPTION
C
C       DEXP    Double precision exponential function.
C       QOMGD   Estimates OMGN1 using the quadratic omega approximation.
C
C COMMON BLOCKS REFERENCED:
C       NAME    USAGE
C
C       DTMD    Diagonal Theta-weighting Matrix and control options for the
C                DTM1D routine.
C       KINDAT  Kinetics parameters.
C
C CHANGE HISTORY:
C       DATE    PROGRAMMER      DESCRIPTION
C
C       9/25/87 M. L. Zerkle    Initial Release
C
C*****************************************************************************
C
      REAL*8 ELDN(25),ELDN2(25),ETBE(25),ITB(25),SETBL,SITBL
      REAL*8 DELT,PTN,PTN1,SUM1,TEMP1,TEMP2
      INTEGER*4 I,M
      COMMON/DTMD/THETD(25),ITER,IOMG
      REAL*8    THETD
      INTEGER*4 ITER,IOMG
      COMMON/KINDAT/TIME,DELN,PN,PN1,OMGN,OMGN1,RHON,RHON1,RHODN,RHODN1,
     1              LEN,LEN1,BT,LP,B(25),L(25),SBL,CN(25),CN1(25),SLCN,
     2              SLCN1,SL2CN,SL2CN1,NDNPG
      REAL*8 TIME,DELN,PN,PN1,OMGN,OMGN1,RHON,RHON1,RHODN,RHODN1,LEN
      REAL*8 LEN1,BT,LP,B,L,SBL,CN,CN1,SLCN,SLCN1,SL2CN,SL2CN1
      INTEGER*4 NDNPG
C
C Initialize if using different timestep.
```

```
C
      IF (DELT .NE. DELN) THEN
        DELT    = DELN
        SETBL = 0.0D0
        SITBL = 0.0D0
        DO 10 I=1,NDNPG
          ELDN(I) = DEXP(-L(I)*DELN)
          THETD(I) = 1.0D0/(DELN*L(I)) - ELDN(I)/(1.0D0-ELDN(I))
          ELDN2(I)      = L(I)*ELDN(I)
          ETBE(I)  = DELN*(1.0D0 - THETD(I))*B(I)*ELDN(I)
          ITB(I)   = DELN*THETD(I)*B(I)
          SETBL    = ETBE(I)*L(I) + SETBL
          SITBL    = ITB(I)*L(I) + SITBL
   10   CONTINUE
      ENDIF
C
C Calc. omega at t(n+1)
C   if IOMG = 0  set Omega(n+1) = Omega(n)
C   if      = 1  calc. Omega(n+1) using quadratic omega method assuming
C                LEN1 = LEN.
C
      IF (IOMG .EQ. 1) CALL QOMGD
C
C Calc. Diagonal Theta Method Source Term.
C
      SUM1 = 0.0D0
      DO 20 I=1,NDNPG
        SUM1 = ELDN2(I)*CN(I) + SUM1
   20 CONTINUE
C
C ***************
C This section is the guts of the Diagonal Theta Method.
C ***************
C
C   Calc. prompt terms for t(n) and t(n+1) before iterating.
C
      PTN  = OMGN*LP + BT - RHON
      PTN1 = OMGN1*LP + BT - RHON1
C
C   Iterate ITER times to converge precursors and omega.
C
      DO 40 M=1,ITER
        SLCN1  = 0.0D0
        SL2CN1 = 0.0D0
C
C     Calc. precursor populations and delayed source for t(n+1).
C
        DO 30 I=1,NDNPG
          TEMP1  = ELDN(I)*CN(I) + SLCN*ETBE(I)/PTN
          TEMP2  = (SUM1 + SETBL*SLCN/PTN)*ITB(I)/(PTN1-SITBL)
          CN1(I) = TEMP1 + TEMP2
          SLCN1  = L(I)*CN1(I) + SLCN1
          SL2CN1 = (L(I)**2)*CN1(I) + SL2CN1
   30   CONTINUE
C
C     Update effective, multigroup decay parameter for t(n+1).
C
        LEN1 = SL2CN1/SLCN1
C
C     Calc. omega for t(n+1).
C
        CALL QOMGD
C
C     Update prompt term and power for t(n+1).
C
        PTN1 = OMGN1*LP + BT - RHON1
        PN1  = LP*SLCN1/PTN1
C
C     Insert CALL to feedback routine or feedback algorithm here.
C
   40 CONTINUE
C
```

```
C Update time.
C
      TIME = TIME + DELN
C
      RETURN
      END
```

```
      SUBROUTINE QOMGD
C*************************************************************************
C
C TITLE:
C
C      Quadratic Omega Approximation, Double Precision (QOMGD)
C
C      Copyright (c) 1987 by Massachusetts Institute of Technology
C
C AUTHOR:
C
C      Michael L. Zerkle
C      MIT, Room 38-181
C      (617)253-0945
C
C PURPOSE:
C
C      This routine determines the inverse period, omega, at the end of a
C timestep assuming that the inverse period is a quadratic function of time.
C It requires the kinetics parameters at the beginning and end of the
C timestep.
C
C CALLING SEQUENCE:
C
C      CALL QOMGD
C
C ARGUMENT(S):
C      NAME    I/O     T*L     A       DESCRIPTION
C
C      NONE.
C
C KEY LOCAL VARIABLES:
C      NAME    T*L     A       DESCRIPTION
C
C      BT      R*8             Total effective delayed neutron fraction.
C      DELN    R*8             Timestep size, DELN=t(n+1)-t(n) [s].
C      LEN     R*8             Effective, multi-group decay parameter
C                               @ t(n) [1/s].
C      LEN1    R*8             Effective, multi-group decay parameter
C                               @ t(n+1) [1/s].
C      LP      R*8             Prompt neutron lifetime [s].
C      OMGN    R*8             Inverse period @ t(n).
C      OMGN1   R*8             Inverse period @ t(n+1).
C      RHODN   R*8             Differential reactivity @ t(n) [1/s].
C      RHODN1  R*8             Differential reactivity @ t(n+1) [1/s].
C      RHON    R*8             Reactivity @ t(n).
C      RHON1   R*8             Reactivity @ t(n+1).
C      SBL     R*8             Sum B(i)*L(i), i=1,NDNPG [1/s].
C      TEMP1   R*8             Temporary variable.
C      TEMP2   R*8             Temporary variable.
C      TEMP3   R*8             Temporary variable.
C      TEMP4   R*8             Temporary variable.
C
C ROUTINES/FUNCTIONS CALLED:
C      NAME    DESCRIPTION
C
C      DSQRT   Double precision square root function, intrinsic.
C
C COMMON BLOCKS REFERENCED:
C      NAME    USAGE
C
C      DTMD    Diagonal Theta-weighting Matrix and control options for the
C               DTM1D routine.
C      KINDAT  Kinetics parameters.
C      THETA   Theta-weighting coefficients.
C
C RESTRICTIONS/SPECIAL CONSIDERATIONS:
C
C      During the first timestep following a step change in reactivity a
C small timestep size should be used to prevent the square root term from
C becoming imaginary.
C
```

```
C CHANGE HISTORY:
C      DATE    PROGRAMMER      DESCRIPTION
C
C      9/25/87 M. L. Zerkle    Initial Release
C
C*********************************************************************************
C
      COMMON/KINDAT/TIME,DELN,PN,PN1,OMGN,OMGN1,RHON,RHON1,RHODN,RHODN1,
     1              LEN,LEN1,BT,LP,B(25),L(25),SBL,CN(25),CN1(25),SLCN,
     2              SLCN1,SL2CN,SL2CN1,NDNPG
      REAL*8 TIME,DELN,PN,PN1,OMGN,OMGN1,RHON,RHON1,RHODN,RHODN1,LEN
      REAL*8 LEN1,BT,LP,B,L,SBL,CN,CN1,SLCN,SLCN1,SL2CN,SL2CN1
      INTEGER*4 NDNPG
      REAL*8 TEMP1,TEMP2,TEMP3,TEMP4
C
C  Calc. omega @ t(n+1).
C
      TEMP1 = (LEN1*LP + (BT-RHON1))*DELN + 2.0D0*LP
      TEMP2 = RHODN1+RHODN+2.0D0*SBL-LEN1*(BT-RHON1)-LEN*(BT-RHON)
      TEMP3 = (((LEN+OMGN)*LP + BT - RHON)*DELN - 2.0D0*LP)*OMGN
      TEMP4 = -TEMP1 + DSQRT(TEMP1**2-4.0D0*DELN*LP*(TEMP3-TEMP2*DELN))
      OMGN1 = TEMP4/(2.0D0*DELN*LP)
C
      RETURN
      END
```

```
      SUBROUTINE THETAM
C*************************************************************************
C
C TITLE:
C
C       Theta-Weighting Method (THETAM)
C
C       Copyright (c) 1987 by Massachusetts Institute of Technology
C
C AUTHOR:
C
C       Michael L. Zerkle
C       MIT, Room 38-181
C       (617)253-0945
C
C PURPOSE:
C
C       This routine solves the point kinetics equations using a general
C version of the traditional theta-weighting method.
C
C METHOD:
C
C       The theta-weighting method is a difference analog in which the
C theta's are weighting coefficients used to specify the contribution of
C the explicit and implicit terms used in the solution.  Two user defined
C weighting coefficients are used:  one for the prompt response associated
C with the amplitude function (THETAP) and the other for the delayed
C response associated with the delayed neutron precursors (THETAD).
C
C       The theta-weighting method is derived by substituting the precursor
C difference analog into the amplitude function analog.  One can then
C solve for the amplitude function at the end of the timestep using the
C amplitude and precursor populations at the beginning of the timestep.
C The new amplitude function is then substituted into the precursor
C difference analog to determine the precursor populations at the end
C of the timestep.
C
C CALLING SEQUENCE:
C
C       CALL THETAM
C
C ARGUMENT(S):
C       NAME    I/O     T*L     A       DESCRIPTION
C
C       NONE.
C
C KEY LOCAL VARIABLES:
C       NAME    T*L     A       DESCRIPTION
C
C       B       R*8     25      Effective delayed neutron fraction of the
C                                ith delayed group.
C       BT      R*8             Total effective delayed neutron fraction.
C       CN      R*8     25      Effective delayed neutron precursor population
C                                at time n (beginning of timestep).
C       CN1     R*8     25      Effective delayed neutron precursor population.
C       DELN    R*8             Timestep size, DELN=t(n+1)-t(n) [s].
C       DELT    R*8             Previous timestep size [s].
C       I       I*4             Delayed group index.
C       L       R*8     25      Decay constant of the ith delayed group [1/s].
C       LEN1    R*8             Effective, multi-group decay parameter
C                                @ t(n+1) [1/s].
C       LP      R*8             Prompt neutron lifetime [s].
C       NDNPG   I*4             Number of delayed neutron precursor groups to
C                                be used in this calculation.
C       OMGN1   R*8             Inverse period @ t(n+1).
C       PN      R*8             Amplitude function @ t(n).
C       PN1     R*8             Amplitude function @ t(n+1).
C       RHON    R*8             Reactivity @ t(n).
C       RHON1   R*8             Reactivity @ t(n+1).
C       SBLE    R*8             Sum of explicit B(i)L(i) terms.
C       SBLI    R*8             Sum of implicit B(i)L(i) terms.
C       SL2CN1  R*8             Weighted delayed neutron source @ t(n+1)
```

```
C                                [s**-2].
C      SLCN     R*8              Delayed neutron source @ t(n) [1/s].
C      SLCN1    R*8              Delayed neutron source @ t(n+1) [1/s].
C      SUM      R*8              Sum of weighted delayed source terms.
C      TDLE     R*8      25      Explicit Delayed Theta*DELN*L(i) term.
C      TDLI     R*8      25      Implicit Delayed Theta*DELN*L(i) term.
C      TEMP1    R*8              Temporary storage variable.
C      TEMP2    R*8              Temporary storage variable.
C      TEMP3    R*8              Temporary storage variable.
C      THETAD   R*8              Delayed weighting coefficient used in THETAM.
C                                THETAD=0 fully explicit.  THETAD=1 fully
C                                implicit. (0 <= THETAD <= 1).
C      THETAP   R*8              Prompt weighting coefficient used in THETAM.
C                                THETAP=0 fully explicit.  THETAP=1 fully
C                                implicit.  (0 <= THETAP <= 1).
C      TIME     R*8              Simulated time [s].
C
C ROUTINES/FUNCTIONS CALLED:
C      NAME     DESCRIPTION
C
C      NONE.
C
C COMMON BLOCKS REFERENCED:
C      NAME     USAGE
C
C      KINDAT   Kinetics parameters.
C      THETA    Theta-weighting coefficients.
C
C RESTRICTIONS/SPECIAL CONSIDERATIONS:
C
C      NONE.
C
C CHANGE HISTORY:
C      DATE     PROGRAMMER       DESCRIPTION
C
C      9/25/87 M. L. Zerkle      Initial Release
C
C*****************************************************************************
C
      COMMON/KINDAT/TIME,DELN,PN,PN1,OMGN,OMGN1,RHON,RHON1,RHODN,RHODN1,
     1              LEN,LEN1,BT,LP,B(25),L(25),SBL,CN(25),CN1(25),SLCN,
     2              SLCN1,SL2CN,SL2CN1,NDNPG
      REAL*8 TIME,DELN,PN,PN1,OMGN,OMGN1,RHON,RHON1,RHODN,RHODN1,LEN
      REAL*8 LEN1,BT,LP,B,L,SBL,CN,CN1,SLCN,SLCN1,SL2CN,SL2CN1
      INTEGER*4 NDNPG
      COMMON/THETA/THETAP,THETAD
      REAL*8 THETAP,THETAD
      REAL*8 DELT,SUM,TEMP1,TEMP2,TEMP3,TDLE(25),TDLI(25),SBLE,SBLI
      INTEGER*4 I
C
C Initialize if using different timestep.
C
      IF (DELT .NE. DELN) THEN
        DELT = DELN
        SBLE = 0.0D0
        SBLI = 0.0D0
        DO 10 I=1,NDNPG
          TDLE(I) = 1.0D0 - (1.0D0-THETAD)*DELN*L(I)
          TDLI(I) = 1.0D0 + THETAD*DELN*L(I)
          SBLE = B(I)*L(I)*(1.0D0-THETAD)*DELN/TDLI(I) + SBLE
          SBLI = B(I)*L(I)*THETAD*DELN/TDLI(I) + SBLI
   10   CONTINUE
      ENDIF
C
C Update Power
C
      SUM = 0.0D0
      DO 20 I=1,NDNPG
        SUM = TDLE(I)*L(I)*CN(I)/TDLI(I) + SUM
   20 CONTINUE
C
      TEMP1 = LP + (1.0D0-THETAP)*DELN*(RHON-BT) + THETAP*DELN*SBLE
```

```
      TEMP2 = TEMP1*PN + DELN*LP*(THETAP*SUM+(1.0D0-THETAP)*SLCN)
      TEMP3 = LP - THETAP*DELN*((RHON1-BT) + SBLI)
      PN1   = TEMP2/TEMP3
CC    OMGN1 = (PN1-PN)/(DELN*PN1)
C
C Update Precursors
C
      SLCN1 = 0.0D0
      SL2CN1 = 0.0D0
      DO 30 I=1,NDNPG
        TEMP1  = (B(I)*DELN*(THETAD*PN1 + (1.0D0-THETAD)*PN)/LP
     1          + TDLE(I)*CN(I))/TDLI(I)
        CN1(I) = TEMP1
        SLCN1  = L(I)*TEMP1 + SLCN1
        SL2CN1 = (L(I)**2)*TEMP1 + SL2CN1
   30 CONTINUE
      LEN1  = SL2CN1/SLCN1
C
C Update Omega and Time
C
      OMGN1 = (RHON1-BT)/LP + SLCN1/PN1
      TIME  = TIME + DELN
C
      RETURN
      END
```

```
      SUBROUTINE TIMER(MODE)
C*****************************************************************************
C
C TITLE:
C
C      Execution timer (TIMER)
C
C      Copyright (c) 1987 by Massachusetts Institute of Technology
C
C AUTHOR:
C
C      Michael L. Zerkle
C      MIT, Room 38-181
C      (617)253-0945
C
C PURPOSE:
C
C      This routine calculates and edits the total execution duration.
C
C CALLING SEQUENCE:
C
C      CALL TIMER(MODE)
C
C ARGUMENT(S):
C      NAME    I/O     T*L     A       DESCRIPTION
C
C      MODE    I       I*4             =0  Initialize.
C                                      =1  Get starting time.
C                                      =2  Get ending time and update
C                                          total execution time.
C                                      =3  Edit total execution time.
C
C FILE(S):
C      UNIT    NAME            [R/W]   USAGE
C
C      6       D:DTMTST.OUT    W       System Output.
C
C KEY LOCAL VARIABLES:
C      NAME    T*L     A       DESCRIPTION
C
C      I       I*2             Do index and temporary variable.
C      IB      I*2     4       Beginning time vector.
C      IE      I*2     4       Ending time vector.
C      ISEC    I*2             Total execution time [s].
C      IT      I*2     4       Total execution time vector.
C
C ROUTINES/FUNCTIONS CALLED:
C      NAME    DESCRIPTION
C
C      GETTIM  Gets the time vector from the system clock.
C      MOD     Modulus function, intrinsic.
C
C COMMON BLOCKS REFERENCED:
C      NAME    USAGE
C
C      NONE.
C
C RESTRICTIONS/SPECIAL CONSIDERATIONS:
C
C      NONE.
C
C CHANGE HISTORY:
C      DATE    PROGRAMMER      DESCRIPTION
C
C      9/25/87 M. L. Zerkle    Initial Release
C
C*****************************************************************************
C
      INTEGER*4 MODE
      INTEGER*2 I,IB(4),IE(4),IT(4),ISEC
C
C  Initialize time arrays.
```

```
C
      IF (MODE .EQ. 0) THEN
        DO 10 I=1,4
          IB(I) = 0
          IE(I) = 0
          IT(I) = 0
   10   CONTINUE
C
C  Get starting time.
C
      ELSE IF (MODE .EQ. 1) THEN
        CALL GETTIM(IB(1),IB(2),IB(3),IB(4))
C
C  Get Ending time and update total time.
C
      ELSE IF (MODE .EQ. 2) THEN
        CALL GETTIM(IE(1),IE(2),IE(3),IE(4))
        DO 20 I=1,4
          IT(I) = IE(I)-IB(I) + IT(I)
   20   CONTINUE
C
C  Calc. and print total time.
C
      ELSE IF (MODE .EQ. 3) THEN
        IF (IT(4) .GE. 0) THEN
          IT(3) = IT(3) + IT(4)/100
          IT(4) = MOD(IT(4),100)
        ELSE
          I = -IT(4)/100 + 1
          IT(3) = IT(3) - I
          IT(4) = IT(4) + 100*I
        ENDIF
C
        IF (IT(3) .GE. 0) THEN
          IT(2) = IT(2) + IT(3)/60
          IT(3) = MOD(IT(3),60)
        ELSE
          I = -IT(3)/60 + 1
          IT(2) = IT(2) - I
          IT(3) = IT(3) + 60*I
        ENDIF
C
        IF (IT(2) .GE. 0) THEN
          IT(1) = IT(1) + IT(2)/60
          IT(2) = MOD(IT(2),60)
        ELSE
          I = -IT(2)/60 + 1
          IT(1) = IT(1) - I
          IT(2) = IT(2) + 60*I
        ENDIF
C
        IF (IT(1) .LT. 0) IT(1) = IT(1) + 24
        ISEC = 3600*IT(1) + 60*IT(2) + IT(3)
C
        WRITE(6,900) IT,ISEC,IT(4)
      ELSE
      ENDIF
C
  900 FORMAT('0',5X,'EXECUTION TIME = ',I2,':',I2,':',I2,'.',I2,/,
     1       ' ',5X,'                 ',I8,'.',I2,' SEC')
C
      RETURN
      END
```