NUMERICAL METHODS FOR SOLVING THE REACTOR KINETICS EQUATIONS

by

John Joseph Szeligowski

A Thesis Submitted to the Faculty of the

DEPARTMENT OF NUCLEAR ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 6 6

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: John J. Szeligowski

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

D. L. Hetrick

D. L. HETRICK
Professor of Nuclear Engineering

June 16 1966
Date

TABLE OF CONTENTS

TABLE OF CONTENTS--Continued

LIST OF TABLES

LIST OF TABLES--Continued

LIST OF FIGURES

ABSTRACT

A number of techniques for circumventing the maximum time-step limitation in numerical solution of the reactor kinetics equation are studied and critically compared. The limitations of finite-difference techniques are demonstrated and the improved methods of Hansen, Cohen, Adler, Keepin, and Collocation are discussed.

For comparison purposes, the example selected is that of a sinusoidal reactivity input with the amplitude and period of the sine curve related through the inhour equation, in an attempt to simulate a negative feedback system.

When generation times of 10^{-8} sec. are discussed, the prompt jump solution is used as a criterion for the accuracy of the various techniques. All methods are compared against each other, with emphasis placed on convergence as a function of the length of the time interval, running time on the digital computer (IBM 7072), and agreement with the prompt-jump solution at a generation time of 10^{-8} sec. It is concluded that the Collocation approximation, when contrasted with the other methods (as programmed here) is the most successful form of numerical solution for the example used.

INTRODUCTION

Kinetic Equations

Several techniques have been developed to solve numerically the point-reactor kinetics equations for slow power transients. These methods are attempts to overcome the inherent difficulty caused by short time constants in the kinetics equations.

The differential equations considered are

$$\frac{dn}{dt} = \frac{\rho - \beta}{\ell} n + \lambda C$$

$$\frac{dC}{dt} = \frac{\beta}{\ell} n - \lambda C$$

where

n = neutron density

C = precursor density

β = delayed-neutron fraction

λ = delayed-neutron decay constant

ℓ = neutron generation time

ρ = reactivity (time-dependent)

Only one average group of delayed neutrons is represented here to conserve IBM-7072 computer time. However, the one group approach will be valid in demonstrating the usefulness of various numerical approximations.

Choice of Reactivity Function

To simulate a realistic situation, the reactivity is taken to be one cycle of a sinusoidal function of time. This artificially simulates a ramp reactivity input and a feedback mechanism which tends to shut the reactor down. The time scale for the power transient created by the sinusoidal reactivity is taken as[1]

$$\tau = \frac{E}{\hat{n}} \approx \frac{4}{\omega}$$

where

τ = time between half-maximum power points

$\hat{n}$ = maximum neutron density (power)

E = energy evolved

ω = initial inverse period due to a reactivity step

This result is derived for large step inputs of reactivity, but is taken as the qualitative basis for simulating a realistic feedback model by a sinusoidal input if the second quarter-cycle of the sine curve is thought of as the shutdown portion of the response to a step. The amplitude of the sine curve corresponds to the magnitude of the step input in the following discussion.

The one-group inhour equation is

$$\rho_o(\text{step}) = \ell\omega + \frac{\beta\omega}{\omega + \lambda}$$

For slow transients, if the term in ℓ is small, the inhour relation

is approximately

$$\rho_o(\text{step}) = \frac{\beta\,\omega}{\omega + \lambda}$$

By choosing a reactivity step size, the initial inverse period ω of the step transient can be determined, thus determining the time scale of the power burst from

$$\tau = \frac{4}{\omega}$$

This leaves only the period of the sinusoidal reactivity undetermined, since its amplitude is taken to correspond to the size of the step. In an attempt to create a meaningful simulation, the time scale of the power transient is allowed to determine the period of the sinusoidal reactivity through the relation

$$\frac{T}{2} = \frac{4}{\omega}$$

where

T = half period of the sinusoidal input

ω = initial period of power transient as predicted by the inhour equation with $\ell \longrightarrow 0$.

In other words, a quarter cycle of the sinusoidal reactivity input is taken to be the time between half-maximum power points.[2] Thus, the amplitude (ρ_o) of the reactivity input determines the period of the reactivity input through the inhour equation, thereby simulating a realistic, self-limiting power transient. Figure 1 illustrates the situation and Appendix A indicates what reactivities were actually used.

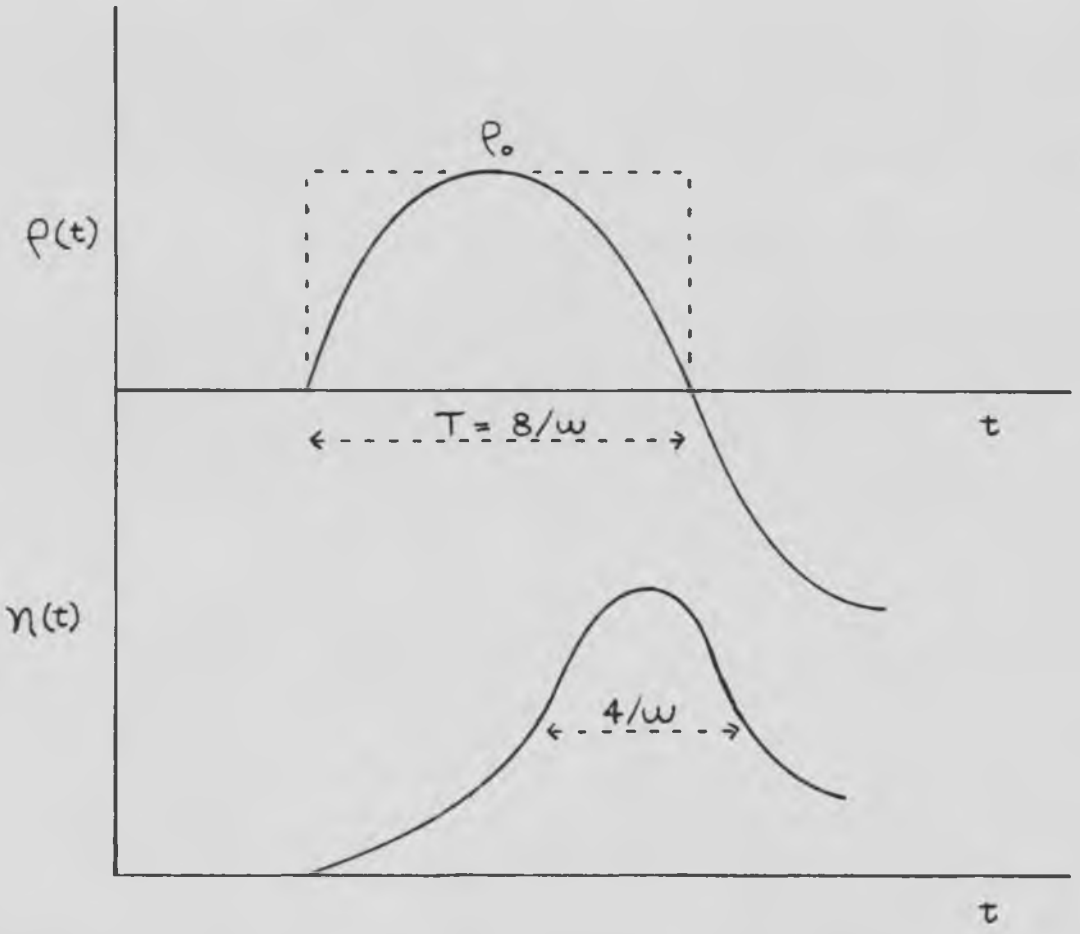

Fig. 1.--Relationship of Reactivity to Power

THE PROMPT-JUMP APPROXIMATION

Derivation

Since only slow transients are considered here, the prompt-jump approximation provides a suitable analytical check for digital computer results when ℓ is very small. In this approximation, the kinetic equations are integrable for a sinusoidal reactivity input.

The kinetics equations are

$$\dot{n} = \frac{\rho - \beta}{\ell} n + \lambda C$$

$$\dot{C} = \frac{\beta}{\ell} n - \lambda C$$

By solving for the precursor concentration in the first equation and substituting into the second equation it is found that

$$\ell \ddot{n} + (\lambda \ell + \beta - \rho)\dot{n} - (\dot{\rho} + \lambda \rho)n = 0.$$

Mathematically the prompt jump approximation is the limit as $\ell \rightarrow 0$ and $\ell \ddot{n} \rightarrow 0$ in the last equation. The result is

$$(\beta - \rho)\dot{n} = (\dot{\rho} + \lambda \rho)n.$$

Rearranging,

$$\frac{\dot{n}}{n} = \frac{\dot{\rho} + \lambda \rho}{\beta - \rho}$$

and

$$\frac{n}{n_o} = e^{\int_0^t \frac{\dot{\rho} + \lambda \rho}{\beta - \rho} \, dt}$$

Sinusoidal Reactivity

The reactivity input is of the form

$$\rho(t) = \rho_o \sin \frac{\pi}{T} t$$

where

ρ_o = amplitude

2τ = period.

Substitution yields

$$\frac{n}{n_o} = e^{\int_o^t \frac{\rho_o \frac{\pi}{T} \cos \frac{\pi}{T} t + \lambda \rho_o \sin \frac{\pi}{T} t}{\beta - \rho_o \sin \frac{\pi}{T}} dt}$$

which can be integrated to

$$\ln \frac{n}{n_o} = \ln \frac{\beta}{\left[\beta - \rho_o \sin \frac{\pi}{T} t\right]}$$

$$-\lambda t + \frac{T}{\pi} \frac{\lambda \beta}{\sqrt{\beta^2 - \rho_o^2}} \left[\sin^{-1}\left(\frac{\rho_o}{\beta}\right) - \sin^{-1}\left(\frac{\rho_o - \beta \sin \frac{\pi}{T} t}{\beta - \rho_o \sin \frac{\pi}{T} t}\right) \right]$$

This result is the prompt-jump solution to the one-delayed-group kinetics equations with a sinusoidal input reactivity. Except for the immediate neighborhood of a discontinuity in ρ, the prompt jump solution is valid when[3]

$$\beta - \rho \gg \sqrt{\ell(\dot{\rho} + \lambda \rho)}$$

A digital computer program for this solution gave the results shown in Figures 2 and 3. In Figure 2 and in subsequent figures the ordinate scale is the power relative to the initial power.

In Figure 3 the power is seen to increase with an increase in the length of the reactivity cycle. However, the amplitude of the sine wave

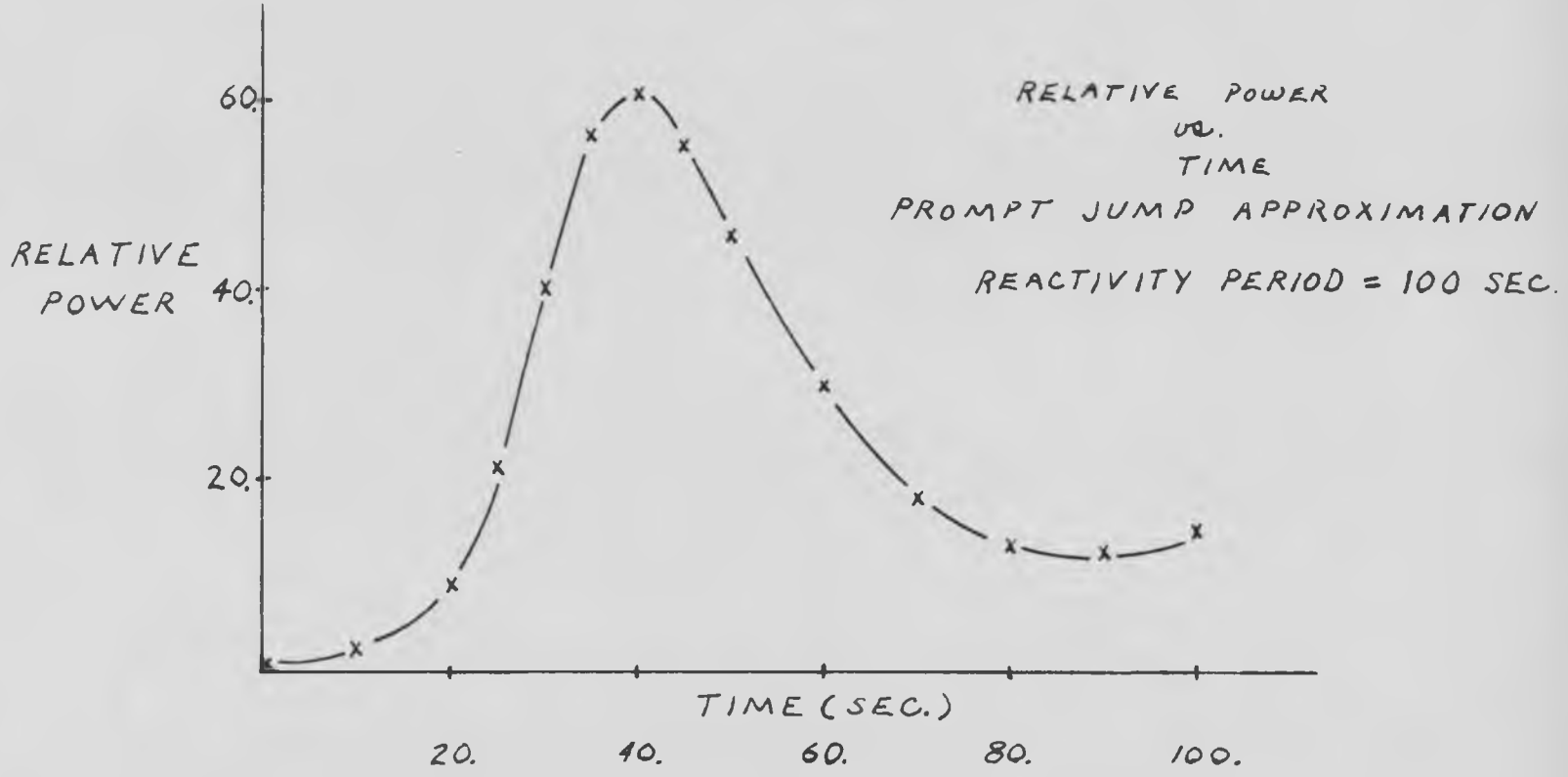

Fig. 2. --Prompt Jump Solution

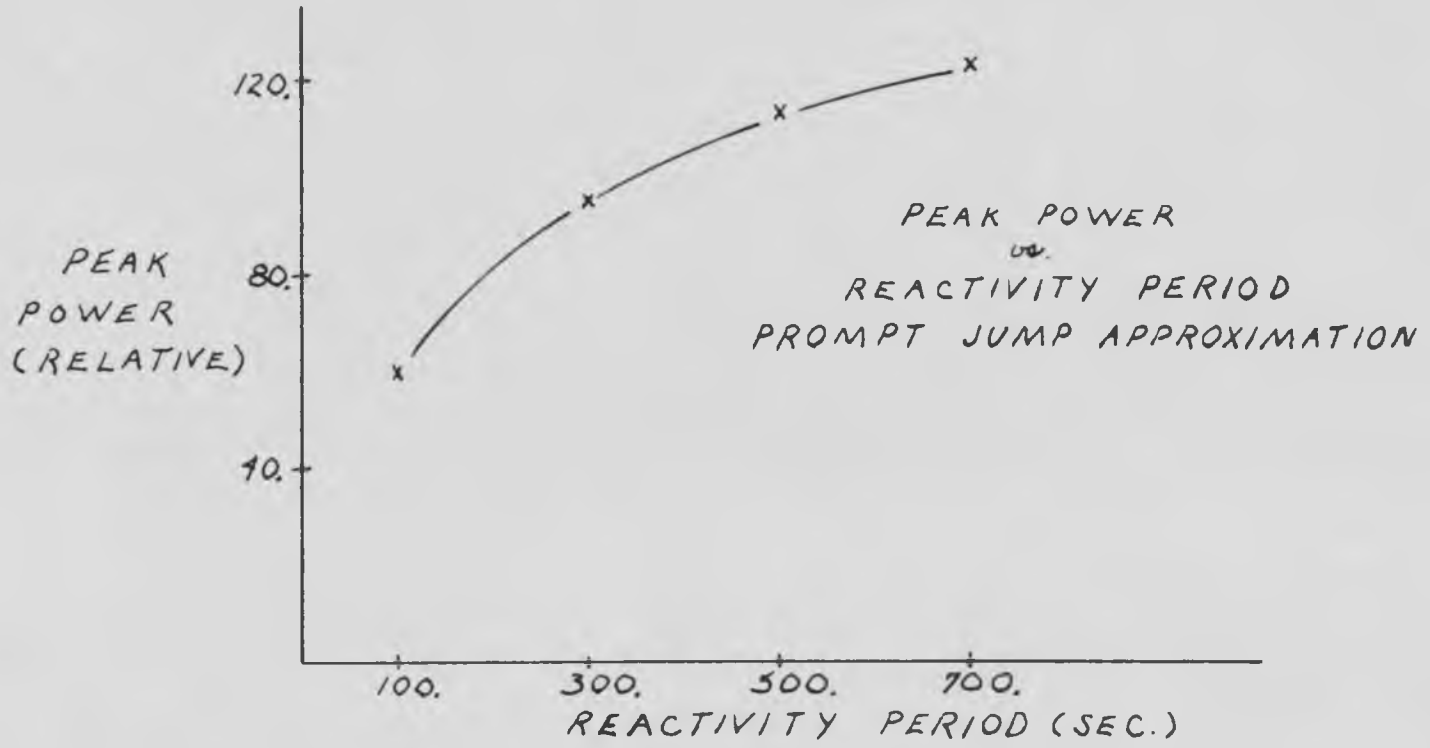

Fig. 3.--Peak Power from Prompt Jump Approximation

decreases with an increase in its period because of the relationships

$$\rho_0 = \frac{\beta \omega}{\omega + \lambda}$$

$$\frac{T}{2} = \frac{4}{\omega}$$

$$\rho_0 = \frac{8\beta}{8 + \lambda T}$$

In a self-limiting system with fixed reactivity feedback, smaller reactivity inputs yield smaller peak power. Here it must be considered that each reactivity input is simulating a different feedback system.

Since the prompt-jump approximation is the solution in the limit as $\ell \longrightarrow 0$, it is expected that the results obtained using it would be more nearly accurate for small ℓ. This solution will, in fact, be considered exact for $\ell = 10^{-8}$ sec. and all other approaches will be compared to it at $\ell = 10^{-8}$ sec. Table 1 shows the data obtained for the prompt-jump approximation on the IBM-7072.

TABLE 1

DATA FOR PROMPT JUMP SOLUTION

Period of Reactivity Input (seconds)	Peak Power (relative)	Time of Peak Power (seconds)
100	61.53378	39.1
300	95.81981	137.3
500	113.46026	237.1
700	123.82470	337.1

FINITE-DIFFERENCE METHODS

Limitations

Finite-difference methods and related methods are unsuitable when the time scale for the dynamic process is many orders of magnitude greater than the smallest characteristic time of the system. In reactor kinetics calculations with small reactivities, the time scale might be hundreds of seconds, while the limiting characteristic time $\ell/(\beta-\rho)$ can be as small as 10^{-6} sec., requiring a prohibitively large number of time steps.

Finite-Difference Method

Writing the dynamics equations as

$$\frac{dx}{dt} = Ax + By$$

$$\frac{dy}{dt} = Cx + Dy$$

and using the forward difference to represent a derivative

$$\frac{X_{n+1} - X_n}{h} = A X_n + By_n$$

$$\frac{Y_{n+1} - Y_n}{h} = C X_n + Dy_n$$

where h is the interval size $t_{n+1} - t_n$.

The last form can be conveniently programmed to give reactor power as a function of time in response to a time-dependent reactivity.

This finite difference technique will, however, be the most sensitive to the fast time constants in the problem. The fast time constants appear in exponents such as

$$\alpha t' = \frac{\rho - \beta}{\ell} t'$$

which occur in formal solutions as terms like

$$\int_{t_o}^{t} y(t') e^{-\alpha(t-t')} \, dt'.$$

The derivation of such integrals will be seen in the next section.

It is the generation time ℓ, ranging from 10^{-3} to 10^{-8} sec., that causes instability in finite difference solutions for small reactivities. The finite difference approach can be made stable if the time interval in the computer program is kept smaller than the time constant.

$$\Delta t < 1/\alpha = \ell/(\beta - \rho).$$

To keep the time interval this small (microseconds in same cases) would require excessive computer time for transients lasting several minutes.

Results of computer solutions using the direct finite difference method are shown in Figures 4 and 5. A sinusoidal reactivity input with a 100 sec. period was used, and complete instability was noted when the generation time was smaller than 2×10^{-5} sec. at a time interval of 0.01 sec. Instabilities in the form of oscillations began showing, as indicated in Figure 4, at a generation time of 10^{-3} sec. with a time interval

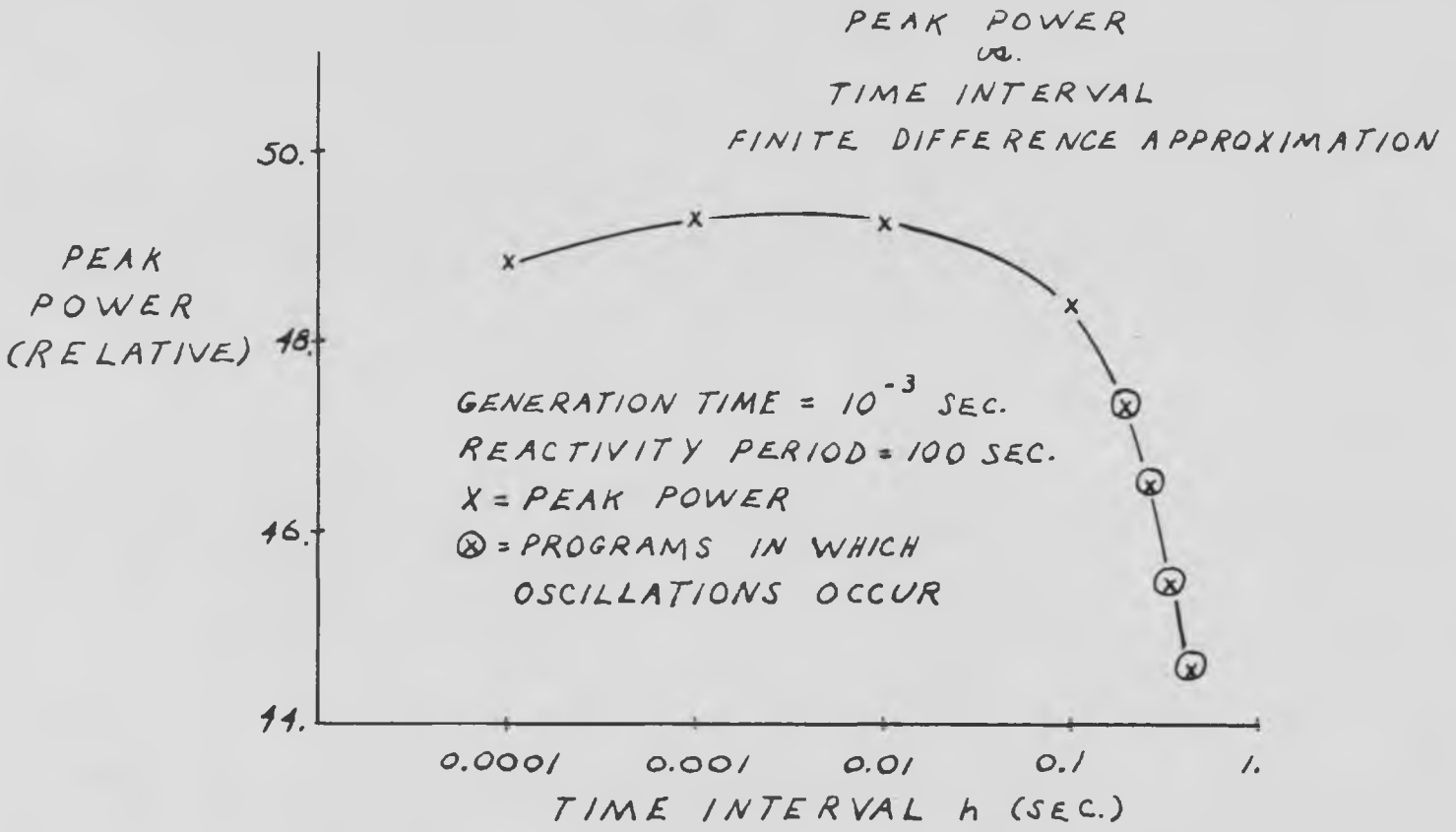

Fig. 4. --Breakdown of Finite Difference Approximation

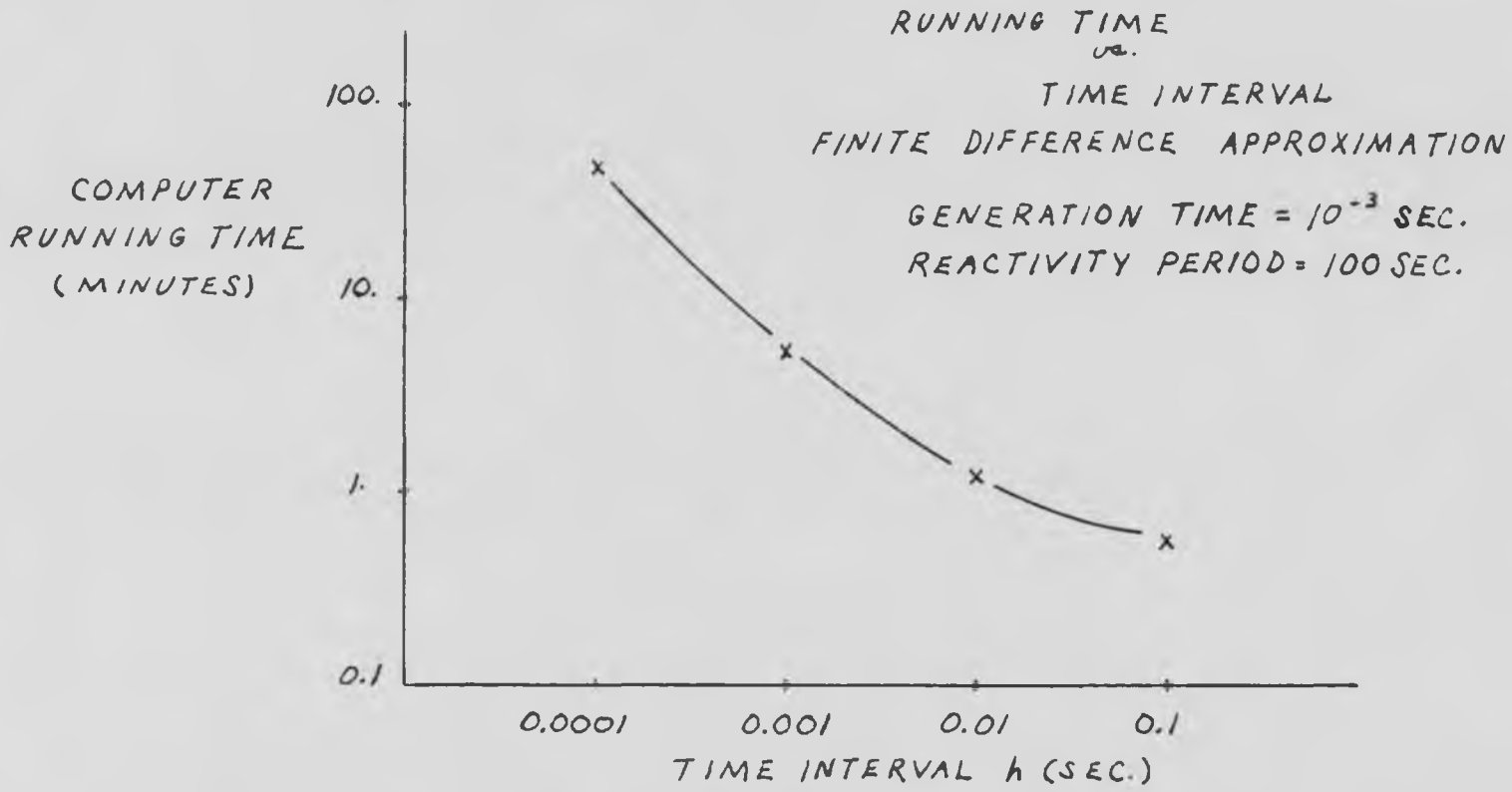

Fig. 5.--Running Times of Finite Difference Approximation

of 0.5 sec. Computer running time as a function of time interval is shown in Figure 5.

Direct finite difference leaves the entire range of generation time from 10^{-5} to 10^{-8} unmanageable unless the time interval is decreased and consequently the computer time excessively increased. The rest of this report concerns itself with attempts to improve on the above results.

Runge-Kutta Method

The next method to be discussed is an improved finite difference technique called the fourth-order Runge-Kutta approximation.[4] This is a somewhat more sophisticated approach than the previous one and is of interest here if it can significantly extend the range of usefulness of finite difference methods. The actual Runge-Kutta equations can be found in the reference cited or in any standard numerical analysis text.

Digital computer results are shown in Figures 6 and 7. They indicate that the Runge-Kutta approach is only slightly more stable than direct finite difference, and at the expense of computer time. Computer running times for both direct finite difference and Runge-Kutta methods are indicated in Tables 2 and 3. The data from both approaches tend to indicate that all such finite difference methods will be frustrated by the limiting time constant $\ell/(\beta - \rho)$.

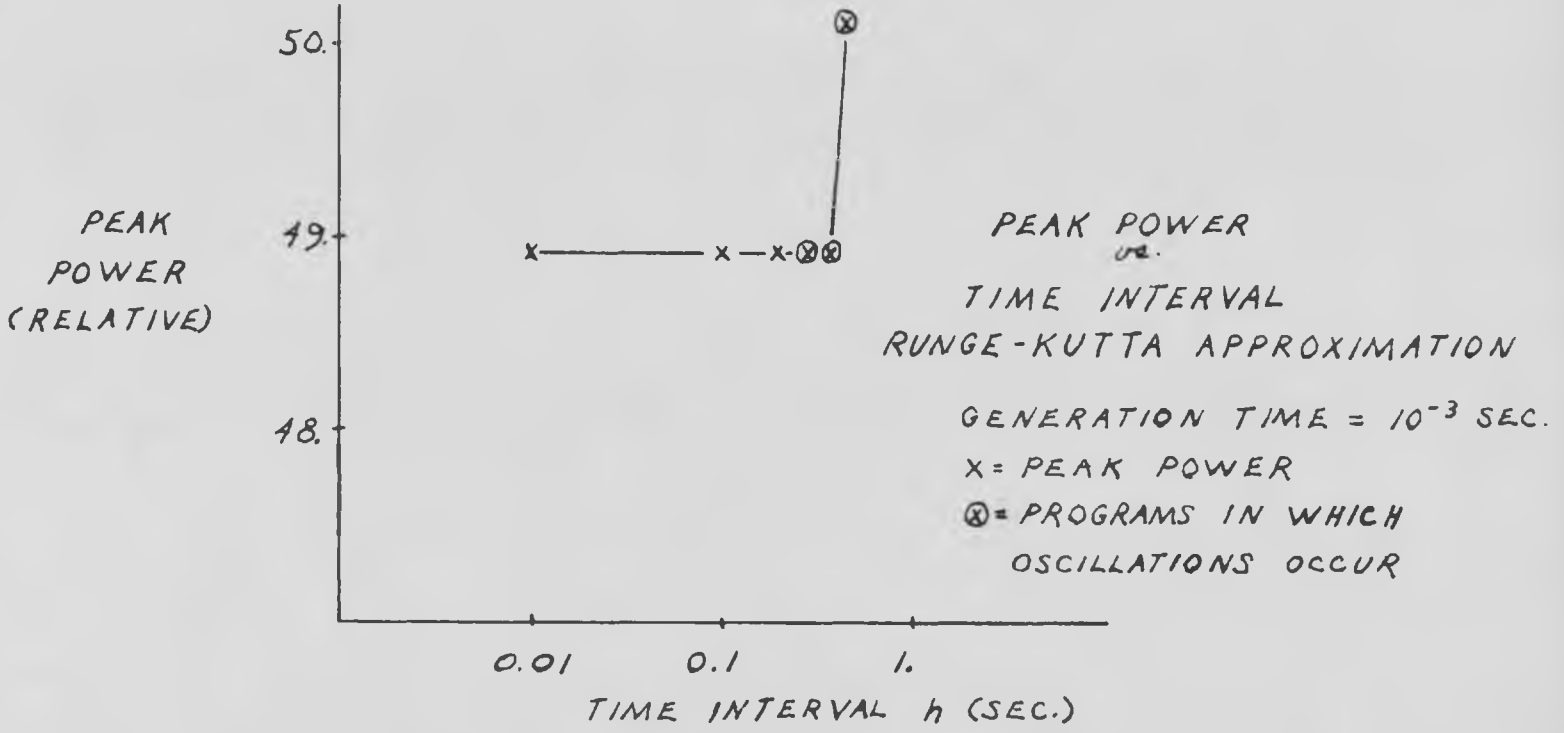

Fig. 6.--Effect of Time Interval on Runge-Kutta Approximation

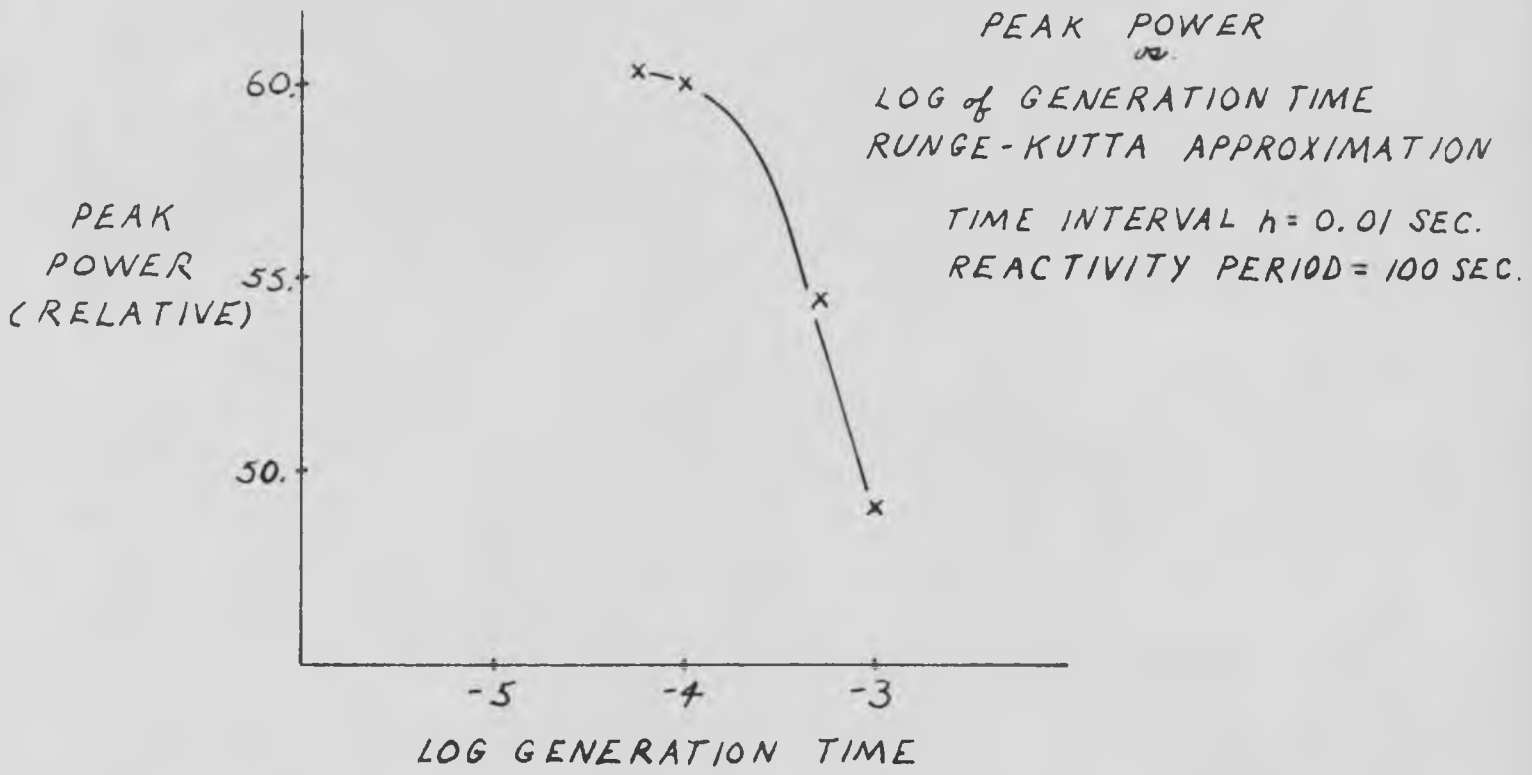

Fig. 7.--Runge-Kutta Approximation for Various Generation Times

TABLE 2

DATA FOR FINITE DIFFERENCE APPROXIMATION

Reactivity Period = 100 sec.

Generation Time (Sec.)	10^{-3}	5×10^{-4}	10^{-4}	5×10^{-5}	2×10^{-5}	10^{-5}
Time Interval (Sec.)						
.0001	***48.81445 39.3 47:13					
.001	49.30593 39.3 5:15					
.01	49.26162 39.3 1:05	54.68717 39.2 1:06	59.91751 39.1 1:06	60.64225 39.1 *60.4	**.4	**.3
.1	48.34376 39.4 :41					
.2	47.35269 39.4 *79.0					
.3	46.39278 39.3 *66.6					

TABLE 2 - Continued

Generation Time (Sec.)	10^{-3}	5×10^{-4}	10^{-4}	5×10^{-5}	2×10^{-5}	10^{-5}
Time Interval (Sec.)						
.4	45.46472					
	39.6					
	*62.4					
.5	44.53206					
	39.5					
	**60.0					

*format overflow at time indicated

**format overflow and oscillations prior to overflow

***In this table, and others where the format is similar, the data represents relative peak power, time of peak power, computer running time in order shown.

TABLE 3

DATA FOR RUNGE-KUTTA SOLUTION

Reactivity Period = 100 Sec.

Generation Time (Sec.)	10^{-3}	5×10^{-4}	10^{-4}	5×10^{-5}	2×10^{-5}	10^{-5}
Time Interval (Sec.)						
.01	***48.88681	54.54376	60.00157	60.75778		
	39.3	39.2	39.1	39.1		
	2:28	2:28	2:39	2:28	**.3	**.1
.1	48.88988					
	39.3					
	:44					
.2	48.88722					
	39.4					
	:52					
.3	48.88558					
	39.3					
	*67.5					
.4	48.87637					
	*60.0					

TABLE 3 - Continued

Generation Time (Sec.)	10^{-3}	5×10^{-4}	10^{-4}	5×10^{-5}	2×10^{-5}	10^{-5}
Time Interval (Sec.)						
.5	50.13141					
	39.5					
	**56.0					
1.	**10.0					

*format overflow at time indicated

**format overflow and oscillations prior to overflow

***In this table, and others where the format is similar, the data represents relative peak power, time of peak power, computer running time in order shown.

HANSEN'S METHOD

Quasilinear Equations

Consider the first-order inhomogeneous differential equation

$$\frac{dy}{dx} + P(x)y = Q(x)$$

which can be written as

$$\frac{dy}{dx} + P(x_o)y = [P(x_o) - P(x)]\, y + Q(x).$$

This last equation is called "quasilinear" and may be integrated to yield

$$y(x) = y(x_o)e^{-P(x_o)\,(x-x_o)}$$

$$+\int_{x_o}^{x} \left\{[P(x_o) - P(x')]\; y(x') + Q(x')\right\}\; e^{-P(x_o)\,(x-x')}\, dx'$$

No new difficulty is introduced if P is a function of both x and y. In terms of reactor kinetics parameters this may be written as

$$n(t) = n(t_o)e^{-\frac{\beta - \rho(t_o)}{\ell}\,(t-t_o)}$$

$$+\int_{t_o}^{t} \left\{\frac{1}{\ell}\,[\rho(t') - \rho(t_o)]\;\; n(t') + \lambda C(t')\right\} e^{-\frac{\beta - \rho(t_o)}{\ell}(t-t')}\, dt'.$$

where in general $\rho(t)$ would be a functional of the reactor power. Using the transformation

$$t' - t_o = \tau$$

$$t - t_o = h$$

$$dt' = d\tau$$

the solution becomes

$$n(t_o + h) = n(t_o) e^{-\frac{\beta - \rho(t_o)}{\ell} h}$$
$$+ \int_o^h \left\{ \frac{1}{\ell} \left[\rho(t_o + \tau) - \rho(t_o)\right] n(t_o + \tau) \right.$$
$$\left. + \lambda C(t_o + \tau) \right\} e^{-\frac{\beta - \rho(t_o)}{\ell} (h - \tau)} d\tau.$$

Here $n(t_o)$ is to be interpreted as the power at the beginning of a time step h, $n(t_o + h)$ as the power at the end of the time step, and $\rho(t_o)$ as the reactivity at the beginning of a time step. There is a companion equation for precursor density:

$$C(t_o + h) = C(t_o) e^{-\lambda h}$$
$$+ \frac{\beta}{\ell} \int_o^h n(t_o + \tau) e^{-\lambda(h - \tau)} d\tau.$$

Both can be written for the special case of constant reactivity:

$$n(t_o + h) = n(t_o) e^{\frac{\rho - \beta}{\ell} h} + \lambda \int_o^h C(t_o + \quad) e^{\frac{\rho - \beta}{\ell} (h - \tau)} d\tau$$

$$C(t_o + h) = C(t_o)^{-\lambda h} + \frac{\beta}{\ell} \int_o^h n(t_o + \tau) e^{-\lambda(h - \tau)} d\tau.$$

Hansen's Approximation

The Hansen approach[5] sets

$$n(t_o + \tau) = n(t_o) e^{\omega_o \tau}$$
$$C(t_o + \tau) = C(t_o) e^{\omega_o \tau}$$

where ω_o is the algebraically largest root of the inhour equation. Making the substitution gives

$$n(t_o + h) = n(t_o) e^{\frac{\rho - \beta}{\ell} h} + \lambda C(t_o) e^{\frac{\rho - \beta}{\ell} h} \int_o^h C(t_o) e^{(\omega_o - \frac{\rho - \beta}{\ell})\tau} d\tau$$

$$n(t_o + h) = n(t_o) e^{\frac{\rho - \beta}{\ell} h} + \lambda C(t_o) e^{\frac{\rho - \beta}{\ell} h} \frac{e^{(\omega_o - \frac{\rho - \beta}{\ell} h)} - 1}{\omega_o - \frac{\rho - \beta}{\ell}}$$

$$n(t_o + h) = n(t_o) e^{\frac{\rho - \beta}{\ell} h} + C(t_o) \frac{e^{\omega_o h} - e^{\frac{\rho - \beta}{\ell} h}}{\omega_o - \frac{\rho - \beta}{\ell}} \lambda$$

and

$$C(t_o + h) = C(t_o) e^{-\lambda h} + n(t_o) \frac{e^{\omega_o h} - e^{-\lambda h}}{\omega + \lambda} \frac{\beta}{\ell}$$

For the more general case of

$$\rho = \rho(t)$$

the above equations can be used with ω_o determined from the inhour equation for an appropriate average reactivity in each time step. In each interval

$$\rho = \frac{1}{2} \left[\rho(t_o) + \rho(t_o + h)\right]$$

was used to obtain the data presented here.

Hansen's method circumvents the problem of short time constants by predicting that the power curve can be reasonably well approximated by $e^{\omega_o t}$ in each interval thus creating an integrable expression.

Generalizing to m delayed groups, this method can be formulated as a matrix equation:

$$\psi\ (t_o + h) = G\ \psi\ (t_o)$$

$$\psi = \begin{bmatrix} n \\ C_1 \\ C_2 \\ \cdot \\ \cdot \\ \cdot \\ C_m \end{bmatrix}$$

$$G = \begin{bmatrix} e^{\frac{\rho-\beta}{\ell}h} & \frac{e^{\omega_o h} - e^{\frac{\rho-\beta}{\ell}h}}{\omega_o - \frac{\rho-\beta}{\ell}} \lambda_1 & \cdots & \frac{e^{\omega_o h} - e^{\frac{\rho-\beta}{\ell}h}}{- \quad -} \lambda_m \\ \frac{e^{\omega_o h} - e^{-\lambda_1 h}}{\omega_o + \lambda_1} \frac{\beta_1}{\ell} & e^{-\lambda_1 h} & \cdots & \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \frac{e^{\omega_o h} - e^{-\lambda_m h}}{\omega_o + \lambda_m} \frac{\beta_m}{\ell} & & & e^{-\lambda_m h} \end{bmatrix}$$

The results of programming this approximation for a sinusoidal reactivity input are shown graphically in Figures 8 and 9. In Figure 8, peak power is plotted against generation time and compared to the prompt-jump solution. The prompt-jump approximation is, as mentioned, a good criterion for accuracy when $\ell \longrightarrow 10^{-8}$. Figure 9 shows the convergence

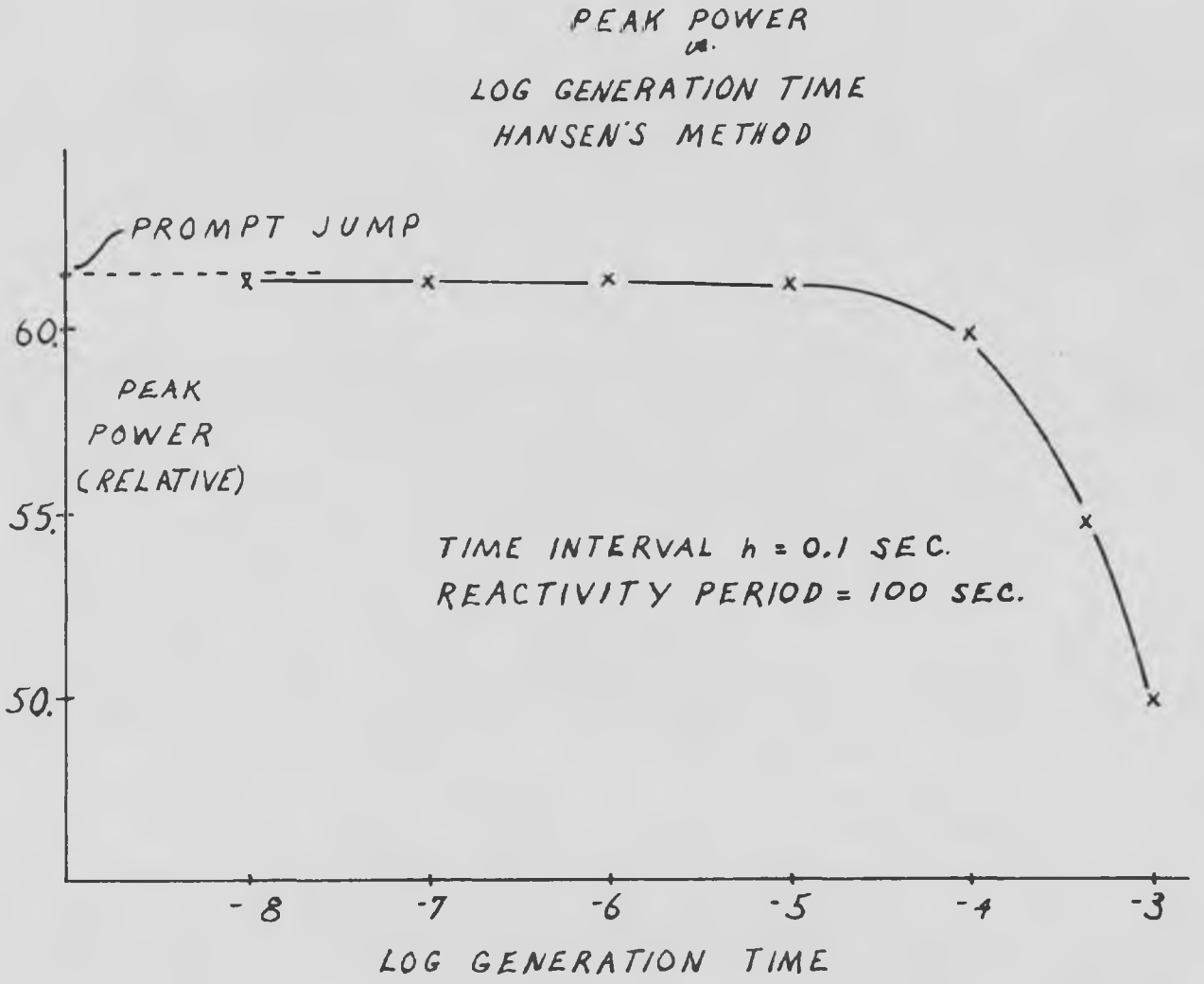

Fig. 8.--Hansen's Approximation for Various Generation Times

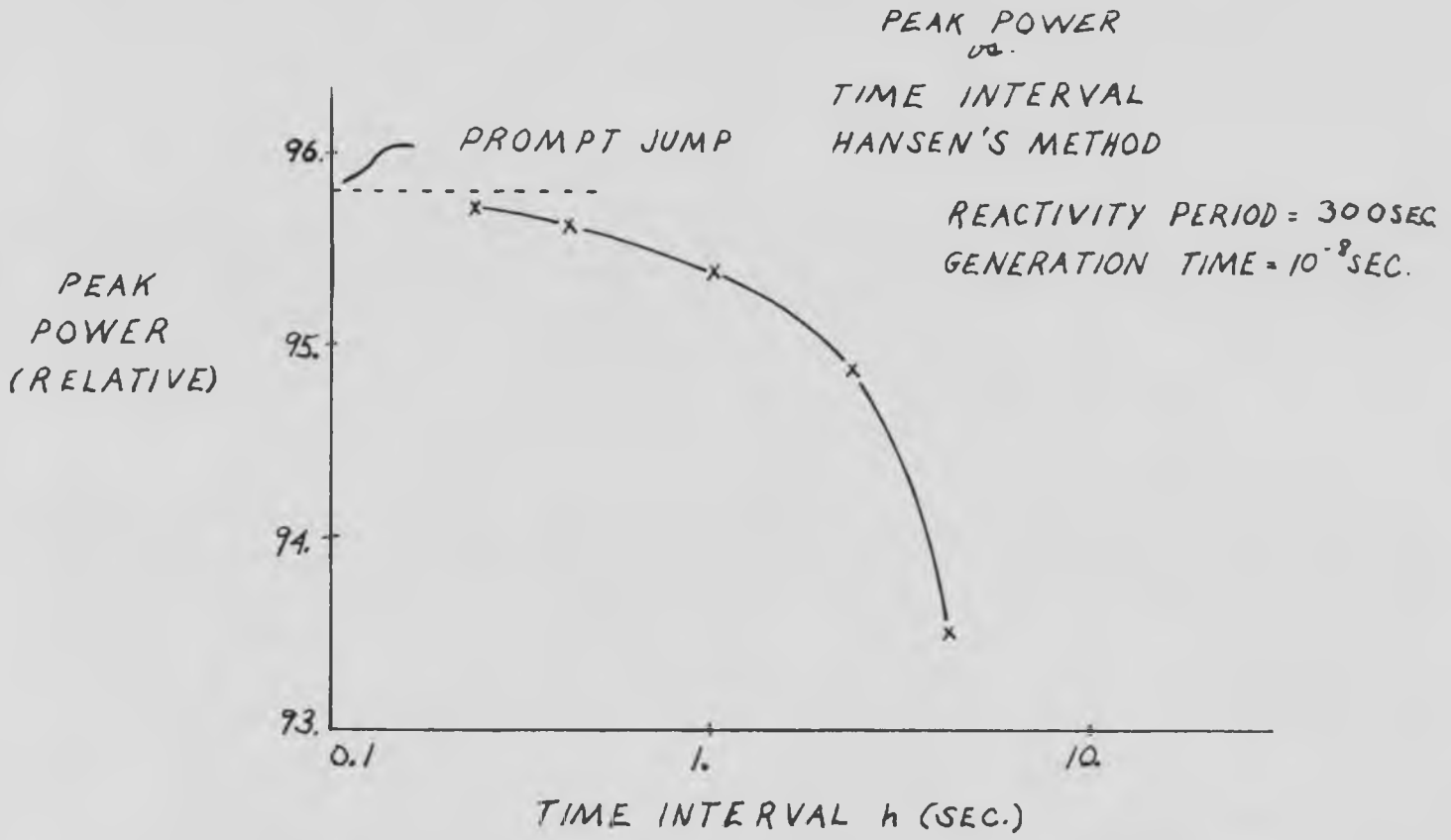

Fig. 9.--Convergence of Hansen's Method with Time Interval

of Hansen's technique at $\ell = 10^{-8}$ sec. and cycle time $2\tau = 300$ sec., and compares it to the prompt-jump solution. The complete set of data including computer running time and shorter generation times is presented in Tables 4, 5 and 6, with computer programming details in Appendix A.

TABLE 4

HANSEN'S APPROXIMATION FOR VARIOUS GENERATION TIMES

Reactivity Period = 100 sec.; Time Increment = 0.1 sec.

Generation Time (Sec.)	Peak Power (Relative)	Time of Peak Power (Sec.)	Running Time
10^{-3}	49.23061	39.4	1:01
5×10^{-4}	54.65128	39.2	1:02
10^{-4}	59.87900	39.2	1:05
10^{-5}	61.20332	39.2	1:00
10^{-6}	61.33922	39.2	1:01
10^{-7}	61.35276	39.2	1:00
10^{-8}	61.35405	39.2	1:01

TABLE 5

CONVERGENCE OF HANSEN'S APPROXIMATION WITH TIME INCREMENTS

Generation Time = 10^{-3} sec.

Reactivity Period (sec.) / Time Increment (sec.)	100	300	500	700
0.05	49.30042 39.35 2:04			
0.1	49.23061 39.4 1:01			
0.2	49.08400 39.4 :31	86.61805 137.6 1:24	*104.84422 237.4 2:32	*115.4458 337.4 3:34
0.5	48.60400 39.5 :11	86.49839 138 :38	104.78853 237.5 :56	115.41496 337.5 1:18
1.	47.85715 40 :06	86.30115 138 :17	104.69371 238 :28	115.35957 338 :39
2.		85.88179 138 :08	104.49795 238 :14	115.24501 338 :19
5.		84.66593 140 :03	103.94169 240 :05	114.92367 340 :08

*Programs run for only one-half the reactivity period (see Appendix A for further discussion).

TABLE 6

CONVERGENCE OF HANSEN'S APPROXIMATION WITH TIME INCREMENTS

Generation Time = 10^{-8} sec.

Reactivity Period (sec.) / Time Increment (sec.)	100	300	500	700
0.05	61.44205 39.5 2:01			
0.1	61.35405 39.2 1:01			
0.2	61.17203 39.2 :30	95.72610 137.6 1:31	*113.40587 237.4 2:58	*123.78166 337.2 3:26
0.5	60.61603 39.5 :12	95.59420 138 :37	113.34630 237.5 1:01	123.75002 337.5 1:26
1.		95.37435 138 :18	113.24303 238 :31	123.69015 338 :42
2.		94.90262 138 :09	113.02837 238 :15	123.56583 338 :21
5.		93.52278 140 :04	112.41635 240 :06	123.21682 340 :09

*Programs run for only one-half the reactivity period (See Appendix A for further discussion).

METHODS OF COHEN AND ADLER

Introduction

Cohen's method,[6] the basis of the AIREK codes, and Adler's method,[7] may both be derived from the quasilinear form. The version of Cohen's method used here is highly simplified, and is not representative of the accuracy of the AIREK codes. However, the same degree of approximation is used here to study both of these methods, and Adler's method is significantly more accurate than Cohen's method for the same time step in the examples studied. Hence Adler's method was studied in greater detail.

Cohen's Method

Both methods begin with the quasilinear form (see previous section)

$$n(t_o + h) = n(t_o)e^{-\frac{\beta - \rho(t_o)}{\ell} h} + \int_o^h \left\{ \frac{1}{\ell} \left[\rho(t_o + \tau) - \rho(t_o)\right] n(t_o + \tau) + C(t_o + \tau) \right\} e^{-\frac{\beta - \rho(t_o)}{\ell}(h - \tau)} d\tau$$

$$C(t_o + h) = C(t_o)e^{-\lambda h} + \frac{\beta}{\ell} \int_o^h n(t_o + \tau) e^{-\lambda(h - \tau)} d\tau .$$

The version of Cohen's technique used here makes the approximations

$$\rho(t_o + \tau) = \rho(\text{average}) \Big|_{o}^{h}$$

$$n(t_o + \tau) = n(\text{average}) \Big|_{o}^{h}$$

$$C(t_o + \tau) = C(\text{average}) \Big|_{o}^{h}$$

This is similar to Hansen's approach in that a particular form for the power in a time interval is assumed. However, once an initial value for the power at a particular time is chosen, an iterative process takes place to minimize the error. The convergence criterion is of the form

$$\frac{n(x)}{n(x+1)} > .99999 \text{ or } \frac{n(x+1)}{n(x)} > .99999,$$

depending on the relative magnitude of n at successive iterations, where

n = power at $t_o + h$

t_o = time at the beginning of an interval

h = length of interval

x = number of the iteration

For the first calculation

$$n(\text{average}) = n(t_o).$$

For the second it becomes

$$n(\text{average}) = \frac{1}{2}[n(1) + n(t_o)]$$

and for the third

$$n(\text{average}) = \frac{1}{2}[n(2) + n(t_o)]$$

Each time a new $n(x)$ is calculated, and then averaged with $n(t_o)$, until the desired convergence is achieved.

Thus Cohen's method, like Hansen's, allows direct integration of the quasilinear forms by assuming a form for $n(t_o + \tau)$, and thereby circumvents the time interval limitations.

The results of the computer programs of Cohen's technique are shown in Figure 10. Tables 7, 8, and 9 present all the data obtained.

TABLE 7

CONVERGENCE OF COHEN'S APPROXIMATION WITH TIME INCREMENTS

*Reactivity Period = 100 sec.; Generation Time = 10^{-3} sec.
**Convergence = .99999

Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
0.1	49.28295	39.3	:58
0.05	49.34738	39.35	1:58
0.02	49.36298	39.32	4:48

* Programs run for only one-half the reactivity period (See Appendix A).

** See page 32.

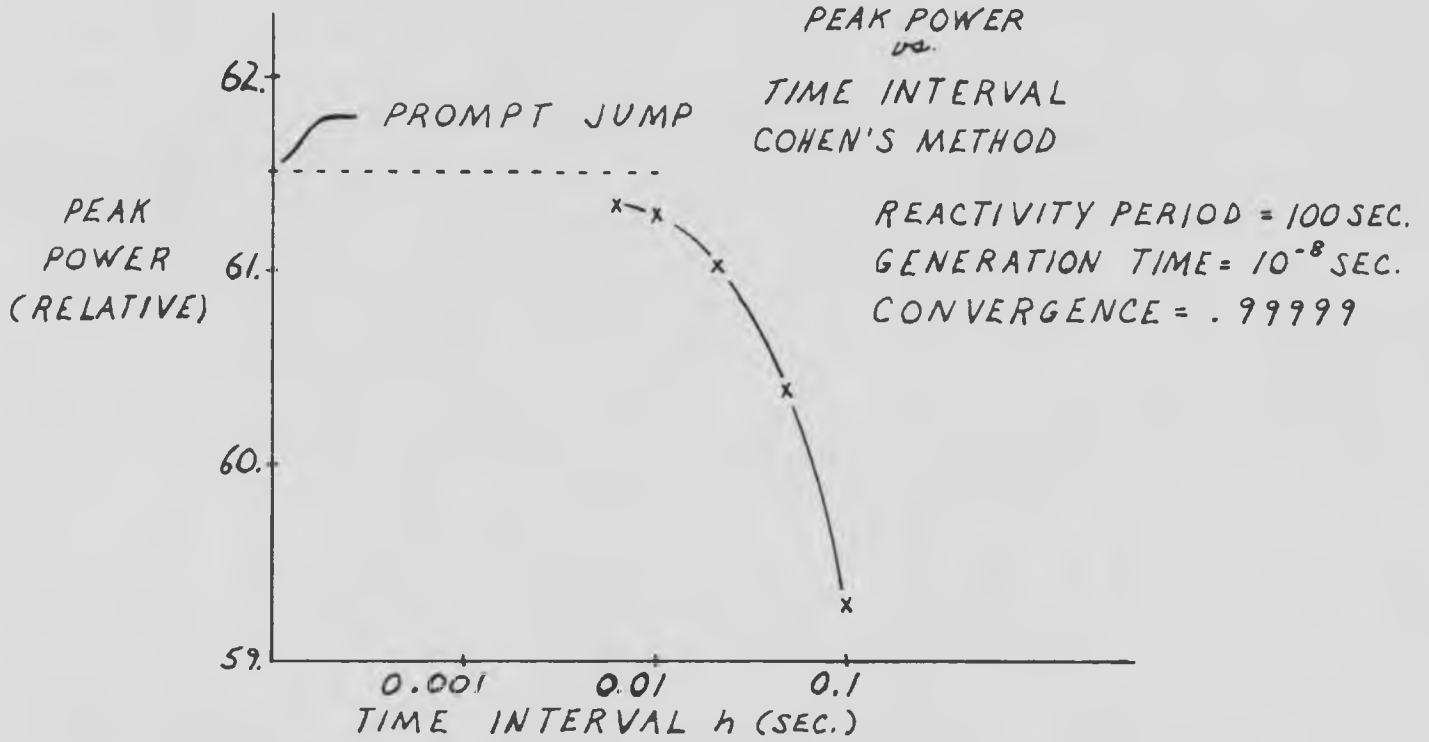

Fig. 10.--Convergence of Cohen's Technique with Time Interval

TABLE 8

CONVERGENCE OF COHEN'S APPROXIMATION WITH TIME INCREMENTS

*Reactivity Period = 100 sec.; Generation Time = 10^{-8} sec.
**Convergence = .99999

Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
0.1	59.27916	39.2	:58
0.05	60.38602	39.15	1:56
0.02	61.06495	39.12	4:50
0.01	61.28962	39.11	9:40
0.0075	61.34388	39.105	12:54

*Programs run for only one-half the reactivity period (See Appendix A).

**See page 32.

TABLE 9

COHEN'S APPROXIMATION FOR VARIOUS GENERATION TIMES

*Reactivity Period = 100 sec.; Time Increment = 0.1 sec.
**Convergence = .99999

Generation Time (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
10^{-3}	49.28295	39.3	:58
10^{-4}	59.05344	39.2	1:00
10^{-5}	59.27913	39.2	:58
10^{-6}	59.27916	39.2	:58
10^{-7}	59.27916	39.2	:58
10^{-8}	59.27916	39.2	:58

* Programs run for only one-half the reactivity period (See Appendix A).

** See page 32.

Adler's Method

Adler begins with the quasilinear form and integrates analytically as far as possible. The procedure can start with

$$\frac{dn}{dt} + \alpha n = \left[\frac{\rho(t) - \beta}{\ell} + \alpha\right] n + \lambda C$$

where α was chosen as

$$\alpha = \frac{\beta - \rho_o}{\ell} ,$$

and where ρ_o is some constant reactivity (average or initial), in the Hansen and Cohen approaches. Other possible forms for α include

$$\alpha = \beta/\ell$$

$$\alpha = 1/\ell$$

The latter was used by Adler in his paper.[7] Then

$$n = n_o e^{-\alpha t} + \int_o^t \left\{\left[\frac{\rho(t') - \beta}{\ell} + \alpha\right] n(t') + \lambda C(t')\right\} e^{-\alpha(t-t')} dt$$

Using

$$C(t) = C_o e^{-\lambda t} + \frac{\beta}{\ell} \int_o^t n(t') e^{-\lambda(t-t')} \, dt'$$

the solution is

$$n = n_o e^{-\alpha t} + \int_o^t \left\{\left[\frac{\rho(t') - \beta}{\ell} + \alpha\right] n(t') + \lambda C_o e^{-\lambda t} + \frac{\lambda \beta}{\ell} \int_o^{t'} n(t'') e^{-\lambda(t'-t'')} \, dt''\right\} e^{-\alpha(t-t')} \, dt' .$$

The last equation can be rearranged to

$$n = n_o e^{-\alpha t} + \int_o^t \left[\frac{\rho(t') - \beta}{\ell} + \alpha\right] n(t') e^{-\alpha(t-t')} \, dt' + \lambda C_o \int_o^t e^{-\lambda t' - \alpha(t-t')} \, dt'$$

$$+ \frac{\lambda \beta}{\ell} \int_o^t \int_o^{t'} n(t'') e^{-\lambda(t'-t'')} dt'' e^{-\alpha(t-t')} dt'$$

At this point integration of the third term is directly possible and a partial integration of the fourth term can be undertaken giving

$$n = n_o e^{-\alpha t}$$
$$+ \int_o^t \frac{\rho(t') - \rho_o}{\ell} n(t') e^{-\alpha(t-t')} dt'$$
$$+ \frac{\lambda C_o}{\alpha - \lambda} \left[e^{-\lambda t} - e^{-\alpha t} \right]$$
$$+ \frac{\lambda \beta}{\ell(\alpha - \lambda)} \int_o^t n(t') \left[e^{-\lambda(t-t')} - e^{-\alpha(t-t')} \right] dt'.$$

The transformations

$$t - t_o = h$$
$$t' - t_o = \tau$$
$$dt' = d\tau$$

will put the equation in a form that can be programmed in time steps "h" on the digital computer as were the finite difference techniques and Cohen's and Hansen's method.

The terms $\rho(t')$ and $n(t')$ become average values throughout each time increment and an iteration process takes place with n as in the Cohen approximation, until a specified convergence criterion is satisfied. The graph in Figure 11 shows the effect of various time increments in Adler's formulation for the same reactivity input. Adler's technique was fully tested for the effect of varying convergence requirements, and the results appear in Tables 10 - 18. Appendix A discusses some of the details of the digital programming of this and the other techniques.

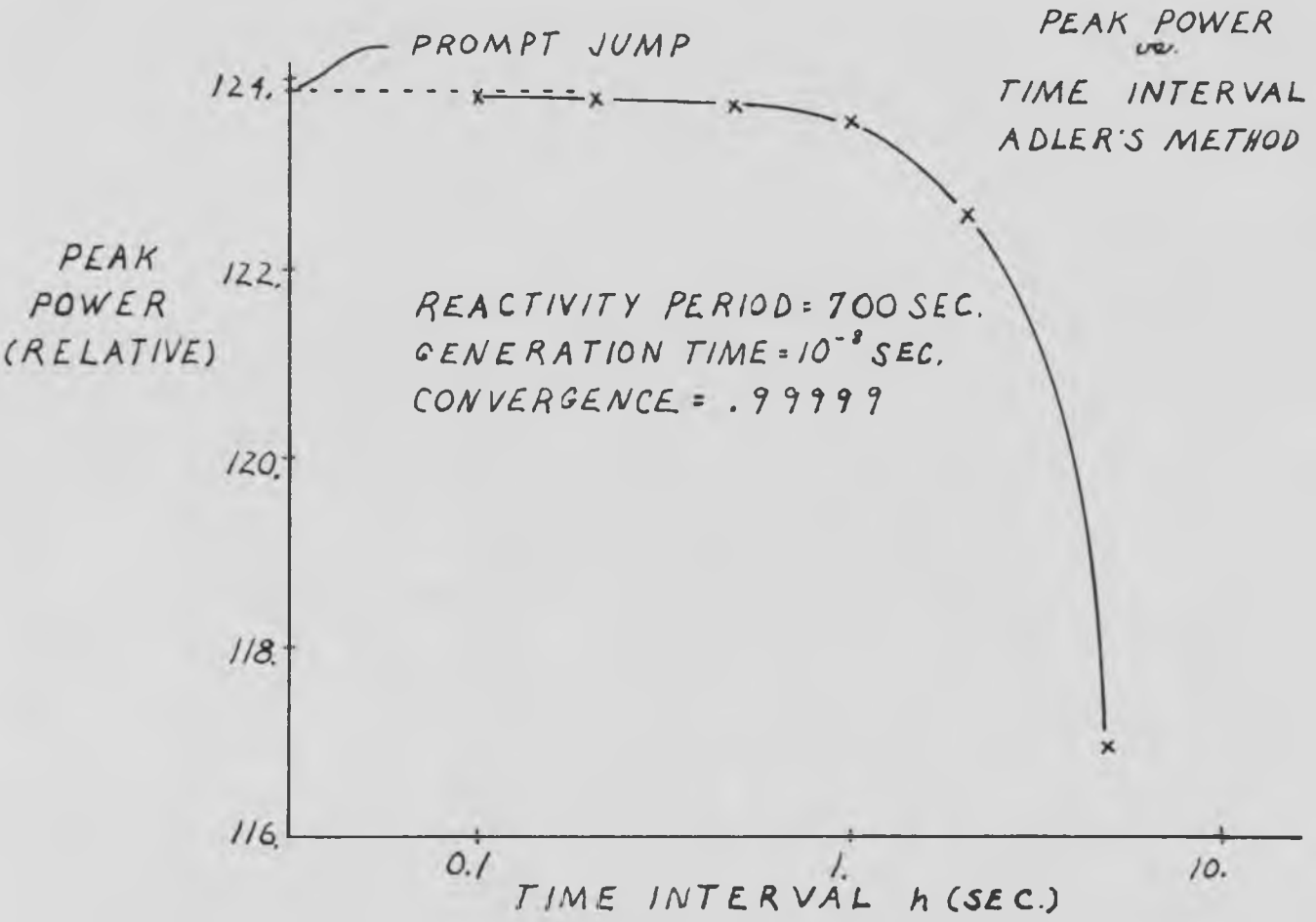

Fig. 11.--Convergence of Adler's Approximation with Time Interval

TABLE 10

CONVERGENCE OF ADLER'S APPROXIMATION WITH TIME INCREMENTS

*Reactivity Period = 100 sec.; Generation Time = 10^{-3} sec.

Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
**Convergence = .999			
.05	49.36382	39.35	2:04
.1	49.37081	39.3	1:02
.2	49.37100	39.4	:30
.5	49.38556	39.5	:14
Convergence = .9999			
.05	49.36382	39.35	2:04
.1	49.37081	39.3	1:02
.2	49.37133	39.4	:32
.5	49.38690	39.5	:14
Convergence = .99999			
.05	49.36382	39.35	2:24
.1	49.37085	39.3	1:06
.2	49.37140	39.4	:36
.5	49.38691	39.5	:14

* Programs run for only one-half the reactivity period (See Appendix A).

** See page 32.

TABLE 11

CONVERGENCE OF ADLER'S APPROXIMATION WITH TIME INCREMENTS

*Reactivity Period = 300 sec.; Generation Time = 10^{-3} sec.

Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
**Convergence = .999			
.1	86.69631	137.5	3:06
.2	86.70041	137.4	1:34
.5	86.73855	137.5	:36
1.	86.81210	138	:20
Convergence = .9999			
.1	86.69631	137.5	3:04
.2	86.70041	137.4	1:34
.5	86.74652	137.5	:42
1.	86.82014	138	:22
Convergence = .99999			
.1	86.69631	137.5	3:06
.2	86.70055	137.4	1:42
.5	86.74663	137.5	:44
1.	86.82019	138	:22

*Programs run for only one-half the reactivity period (See Appendix A).

**See page 32.

TABLE 12

CONVERGENCE OF ADLER'S APPROXIMATION WITH TIME INCREMENTS

*Reactivity Period = 500 sec.; Generation Time = 10^{-3} sec.

Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
**Convergence = .999			
.2	104.87904	237.2	2:34
.5	104.90932	237.5	1:02
1.	104.89938	238	:32
2.	104.62297	238	:18
Convergence = .9999			
.2	104.87904	237.2	2:34
.5	104.91621	237.5	1:08
1.	104.92786	238	:36
2.	104.63497	238	:18
Convergence = .99999			
.05	104.82415	237.25	10:20
.1	104.87759	237.3	5:10
.2	104.87920	237.2	2:50
.5	104.91673	237.5	1:12
1.	104.92806	238	:36
2.	104.64010	238	:18

*Programs run for only one-half the reactivity period (See Appendix A).

**See page 32.

TABLE 13

CONVERGENCE OF ADLER'S APPROXIMATION WITH TIME INCREMENTS

*Reactivity Period = 700 sec.; Generation Time = 10^{-3} sec.
**Convergence = .99999

Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
0.1	115.45711	337.2	*7:16
0.2	115.45723	337.2	*3:56
0.5	115.49206	337.5	1:49
1.	115.46908	338	:50
2.	115.05252	338	:26
5.	110.74269	340	:11

* Programs run for only one-half the reactivity period (See Appendix A).

** See page 32.

TABLE 14

CONVERGENCE OF ADLER'S APPROXIMATION WITH TIME INCREMENT

*Reactivity Period = 100 sec.; Generation Time = 10^{-8} sec.

Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
**Convergence = .999			
.5	61.44530	39.5	:13
.2	61.51352	39.2	:32
.1	61.52749	39.2	1:04
.05	61.52242	39.15	2:08
Convergence = .9999			
.5	61.44649	39.5	:14
.2	61.51739	39.2	:34
.1	61.52793	39.2	1:05
.05	61.52242	39.15	2:08
Convergence = .99999			
.5	61.44660	39.5	:15
.2	61.51742	39.2	:36
.1	61.52798	39.2	1:10
.05	61.52250	39.15	2:30
.02	61.51349	39.12	3:16

* Programs run for only one-half the reactivity period (See Appendix A)

** See page 32.

TABLE 15

CONVERGENCE OF ADLER'S APPROXIMATION WITH TIME INCREMENT

*Reactivity Period = 300 sec ; Generation Time = 10^{-8} sec

Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
**Convergence = .999			
1.	95.60244	138	:20
.5	95.74317	137.5	:38
.2	95.79855	137.4	1:36
.1	95.80855	137.4	3:12
Convergence = .9999			
1.	95.61131	138	:22
.5	95.76404	137.5	:43
.2	95.79950	137.4	1:39
.1	95.80855	137.4	---
Convergence = .99999			
1.	95.61163	138	:22
.5	95.76421	137.5	:54
.2	95.79992	137.4	1:48
.1	95.80874	137.4	3:21

* Programs run for only one-half the reactivity period (See Appendix A)

** See page 32.

TABLE 16

CONVERGENCE OF ADLER'S APPROXIMATION WITH TIME INCREMENT

*Reactivity Period = 500 sec.: Generation Time = 10^{-8} sec.

Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
**Convergence = .999			
2.	112.42492	238	:18
1.	113.16851	238	:33
.5	113.37093	237.5	1:04
.2	113.42622	237.5	2:40
Convergence = .9999			
2.	112.43835	238	:18
1.	113.19711	238	:37
.5	113.38704	237.5	1:10
.2	113.42622	237.2	2:40
Convergence = .99999			
2.	112.44626	238	:19
1.	113.19725	238	:37
.5	113.38747	237.5	1:13
.2	113.42727	237.2	2:58

* Programs run for only one-half the reactivity period (See Appendix A)

** See page 32.

TABLE 17

CONVERGENCE OF ADLER'S APPROXIMATION WITH TIME INCREMENT

*Reactivity Period = 700 sec.; Generation Time = 10^{-8} sec.
**Convergence = .99999

Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
.1	123.79086	337.1	*7:42
.2	123.77824	337.2	*4:08
.5	123.73867	337.5	1:42
1.	123.52596	338	:51
2.	122.67681	338	:27
5.	116.94429	340	:12

* Programs run for only one-half the reactivity period (See Appendix A).

** See page 32.

TABLE 18

ADLER'S APPROXIMATION FOR VARIOUS GENERATION TIMES

*Reactivity Period = 100 sec.; Time Increment = 0.1 sec.
**Convergence = .99999

Generation Time (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
10^{-3}	49.37085	39.3	*1:06
5×10^{-4}	54.81102	39.2	*1:08
10^{-4}	60.05458	39.2	*1:10
10^{-5}	61.37820	39.1	*1:10
10^{-6}	61.51314	39.2	*1:10
10^{-7}	61.52661	39.2	*1:10
10^{-8}	61.52798	39.2	1:10

* Programs run for only one-half the reactivity period (See Appendix A).

** See page 32.

COLLOCATION METHOD

The Collocation Technique [8, 9] is another approximation along the line of those already discussed. The point of the method is best understood if it is begun with the Volterra type integral equation previously derived from the quasilinear form. The general form of the Volterra integral equation is

$$x(t_o + h) = x(t_o) + \int_o^h K(t_o, \tau)\, x(t_o + \tau)\, d\tau$$

and is the type of equation used in the Adler approximation.

The Collocation Technique consists of letting

$$x(t_o + \tau) = g(\tau) = x(t_o)\left[1 + A_1 \tau + A_2 \tau^2\right]$$

where

$$x(t_o + \tau) \Rightarrow n(t_o + \tau).$$

The integral equation becomes

$$g(h) = x(t_o) + \int_o^h K(t_o, \tau)\, g(\tau)\, d\tau + E,$$

and E is the inherent error.

Then

$$x(t_o)(1 + A_1 h + A_2 h^2)$$
$$= x(t_o) + x(t_o) \int_o^h K(t_o, \tau)\left[1 + A_1 \tau + A_2 \tau^2\right] d\tau + E.$$

The error can be written as

$$E = \left\{ A_1 h + A_2 h^2 - \int_o^h K(t_o, \tilde{\tau}) \left[1 + A_1 \tilde{\tau} + A_2 \tilde{\tau}^2 \right] d\tilde{\tau} \right\} x(t_o)$$

Since the assumed form for the power trace in the time interval is parabolic, it contains two undetermined parameters A_1 and A_2. These are determined by forcing the error to be zero in the middle and at the end of the time step. This gives

$$E(A_1, A_2, t_o, \frac{h}{2}) = 0$$

$$E(A_1, A_2, t_o, h) = 0$$

and provides two equations to solve for A_1 and A_2. Thus the form of the power trace is a known function which can be substituted into an integral equation such as derived by Adler, thereby avoiding the time constant difficulty. Figure 12 is a graph showing the convergence of the method with decreasing time steps for a specific reactivity. The complete data obtained using Collocation is presented in Tables 19-21.

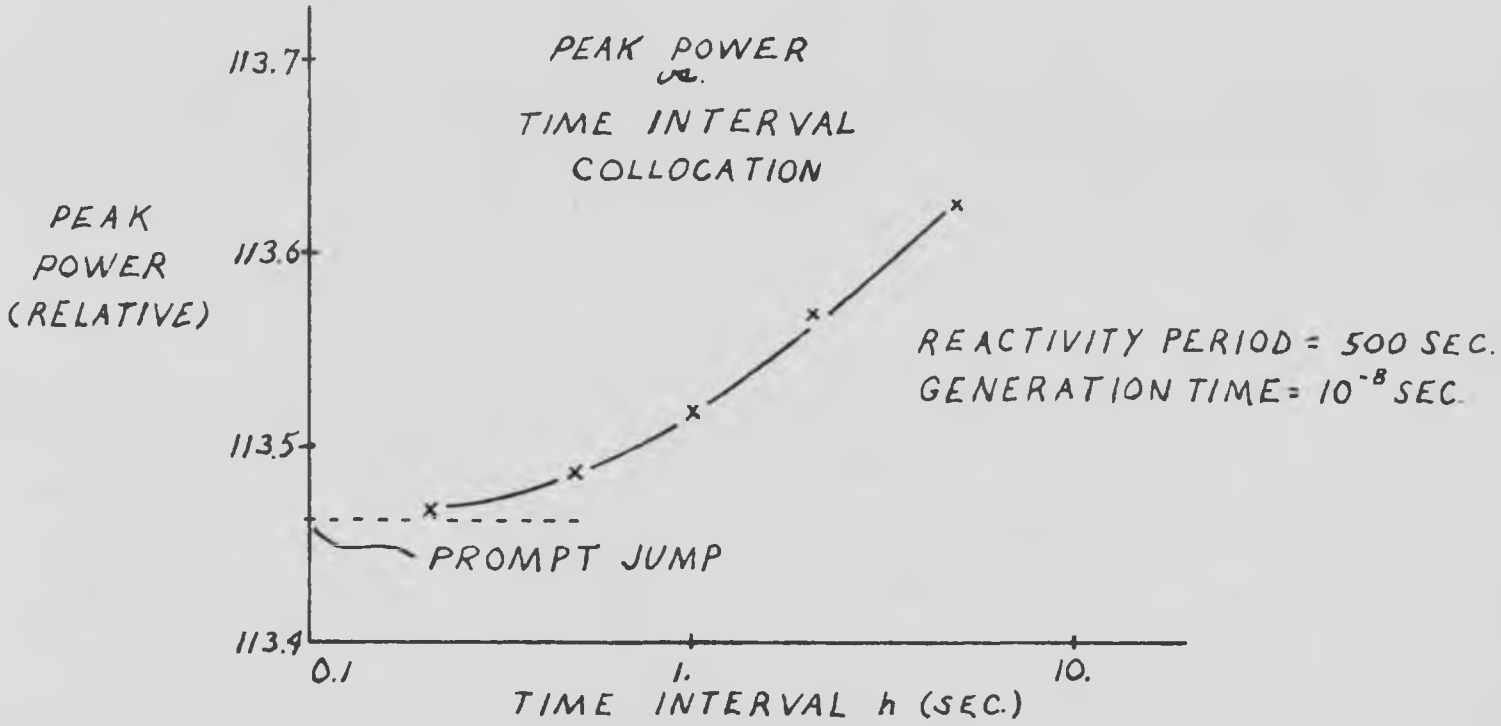

Fig. 12. --Convergence of Collocation with Time Interval

TABLE 19

COLLOCATION FOR VARIOUS GENERATION TIMES

Reactivity Period = 100 sec.; Time Increment = 0.1 sec.

Generation Time (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
10^{-3}	49.37198	39.3	1:40
5×10^{-4}	54.81343	39.2	1:40
10^{-4}	60.09191	39.2	1:41
10^{-5}	61.40204	39.2	1:39
10^{-6}	61.52494	39.2	1:36
10^{-7}	61.53460	39.2	1:36
10^{-8}	61.56489	39.1	1:36

TABLE 20

CONVERGENCE OF COLLOCATION WITH TIME INCREMENTS

Generation Time = 10^{-3} sec.

Reactivity Period (sec.)	100	300	500	700
Time Increment (sec.)				
0.05	49.37961 39.35 3:27			
0.1	49.37198 39.3 1:40			
0.2	49.36493 39.4 :50	86.70376 137.4 2:30	*104.89117 237.2 4:10	*115.47633 337.2 5:50
0.5	49.38228 39.5 :20	86.73220 137.5 1:01	104.90671 237.5 1:40	115.48772 337.5 2:21
1.		86.79686 138 :30	104.94007 237 :51	115.50958 337 1:10
2.		86.89424 138 :15	104.99487 238 :25	115.54333 338 :36
5.			105.04247 240 :10	115.58129 340 :14

* Programs run for only one-half the reactivity period (see Appendix A for further discussion).

TABLE 21

CONVERGENCE OF COLLOCATION WITH TIME INCREMENTS

Generation Time = 10^{-8} sec.

Reactivity Period (sec.)	50	150	250	350
Time Increment (sec.)				
0.05	61.47340 39.15 3:20			
0.1	61.56489 39.1 1:36			
0.2	61.62902 39.2 :49	95.84471 137.4 2:25	*113.46765 237.2 4:00	*123.82056 337.2 5:48
0.5	61.77586 39.2 :19	95.88692 137.5 :58	113.48812 237.5 1:38	123,83659 337 2:16
1.		95.94360 138 :30	113.51796 237 :48	123.85537 337 1:08
2.		96.04278 138 :14	113.57297 238 :25	123.88930 338 :34
5.			113.62422 240 :09	123.93014 340 :13

* Programs run for only one-half the reactivity period (see Appendix A for further discussion).

KEEPIN'S METHOD

The final approximation considered here is that of Keepin,[10] which is based on a Laplace transform of the kinetic equation. Integrating the precursor equation

$$\dot{C} = \frac{\beta}{\ell} n - \lambda C$$

yields

$$C = C_o e^{-\lambda t} + \frac{\beta}{\ell} \int_o^t n(t') e^{-\lambda(t-t')} dt'.$$

Integrating by parts gives

$$C = (\lambda C_o - \frac{\beta}{\ell} n_o) e^{-\lambda t} + \frac{\beta}{\ell} n - \frac{\beta}{\ell} \int_o^t \dot{n}(t') e^{-\lambda(t-t')} dt'$$

which can be substituted into

$$\dot{n} = \frac{\rho - \beta}{\ell} n + \lambda C$$

resulting in

$$\dot{n} = \frac{\rho}{\ell} n - \frac{\beta}{\ell} \int_o^t \dot{n}(t') e^{-\lambda(t-t')} dt' + (\lambda C_o - \frac{\beta}{\ell} n_o) e^{-\lambda t}.$$

Rewriting this and taking the Laplace transform gives the result

$$\ell L(\dot{n}) = L(\rho n) - \beta L(\dot{n}) L(e^{-\lambda t}) + (\lambda \ell C_o - \beta n_o) L(e^{-\lambda t})$$

which can be reduced to

$$\ell(sN-n_o) = L(\rho n) - \beta \frac{sN-n_o}{s+\lambda}$$

$$+ \frac{\lambda \ell C_o - \beta n_o}{s+\lambda}$$

Solving for N gives

$$N = \frac{n_o}{s} + \frac{1}{s\left(\ell + \frac{\beta}{s+\lambda}\right)} \left\{ L(\rho n) + \frac{\lambda \ell C_o - \beta n_o}{s+\lambda} \right\}$$

Define

$$G(s) = \frac{1}{s\left(\ell + \frac{\beta}{s+\lambda}\right)}$$

which in the time domain is

$$G(t) = A_1 + A_2 e^{-\alpha t}$$

where

$$\alpha = \lambda + \beta/\ell$$

$$A_1 = \frac{1}{\ell} \quad \frac{\lambda}{\lambda + \beta/\ell}$$

$$A_2 = \frac{1}{\ell} \quad \frac{\beta/\ell}{\lambda + \beta/\ell} .$$

Thus

$$N = \frac{n_o}{s} + G(s) \left\{ L(\rho n) + \frac{\lambda \ell C_o - \beta n}{s+\lambda} \right\}$$

and in the time domain

$$n(t) = n_o + \int_o^t G(t-t') \, S(t') \, dt'$$

where $S(t')$ replaces the bracketed term in the previous equation. Note that $G(s)$ is the zero power transfer function and $G(t)$ is the impulse response.

This last equation, which is easily generalized to several groups of delayed neutrons, forms the basis of Keepin's RTS code.[10] However, the treatment of the integral in the RTS code does not circumvent the time-step limitation. Therefore, the following alternative procedure was employed. Let

$$y = n(t)$$

$$nh = t$$

$$y(o) = n(o)$$

then

$$\begin{aligned} y(nh) = {} & y(o) \\ & + \int_o^h G(nh - t')\, S(t')\, dt' \\ & + \int_h^{2h} G(nh - t')\, S(t')\, dt' \\ & \cdot \\ & \cdot \\ & \cdot \\ & + \int_{(n-1)h}^{nh} G(nh - t')\, S(t')\, dt' \end{aligned}$$

Proceeding as in the previous cases, a form is assumed for $S(t')$, making the equation integrable and eliminating the time constant difficulty. Letting $S = \bar{S} = S(\text{average})$, where

$S(\text{average})$ = average value in the time interval

results in an approximation similar to Adler's and Cohen's which can in a like manner be iterated until a convergence criterion is met. The form with this approximation becomes

$$y(nh) \cong y(o) + \sum_{m=1}^{n} \bar{S}_m \int_{(m-1)h}^{mh} G(nh-t')\, dt'$$

and when G(t) is replaced by the time function it represents and integrated the result is

$$y(nh) \cong y(o) + \sum_{m=1}^{n} \bar{S}_m \left[A_1 h + \underline{A_2}\, e^{-\alpha(n-m)h}(1-e^{-\alpha h})\right]$$

For convenience, this will be called the exact Keepin equation.

A highly simplified approximation results if ℓ is small.

Then $\alpha h \gg 1$

when $\alpha = \lambda + \beta/\ell$

Then $e^{-\alpha h} \ll 1$

and $e^{-\alpha(n-m)h} \ll 1$

unless

$m = n.$

Using these approximations,

$$y(nh) = y(o) + A_1 h \sum_{m=1}^{n} \bar{S}m + \frac{A_2}{\alpha} \bar{S}n$$

for small ℓ. This will be called the approximate Keepin equation. The results of the approximate Keepin equation appear in Figure 13. The complete set of data is shown in Table 22. For further details, see Appendix A.

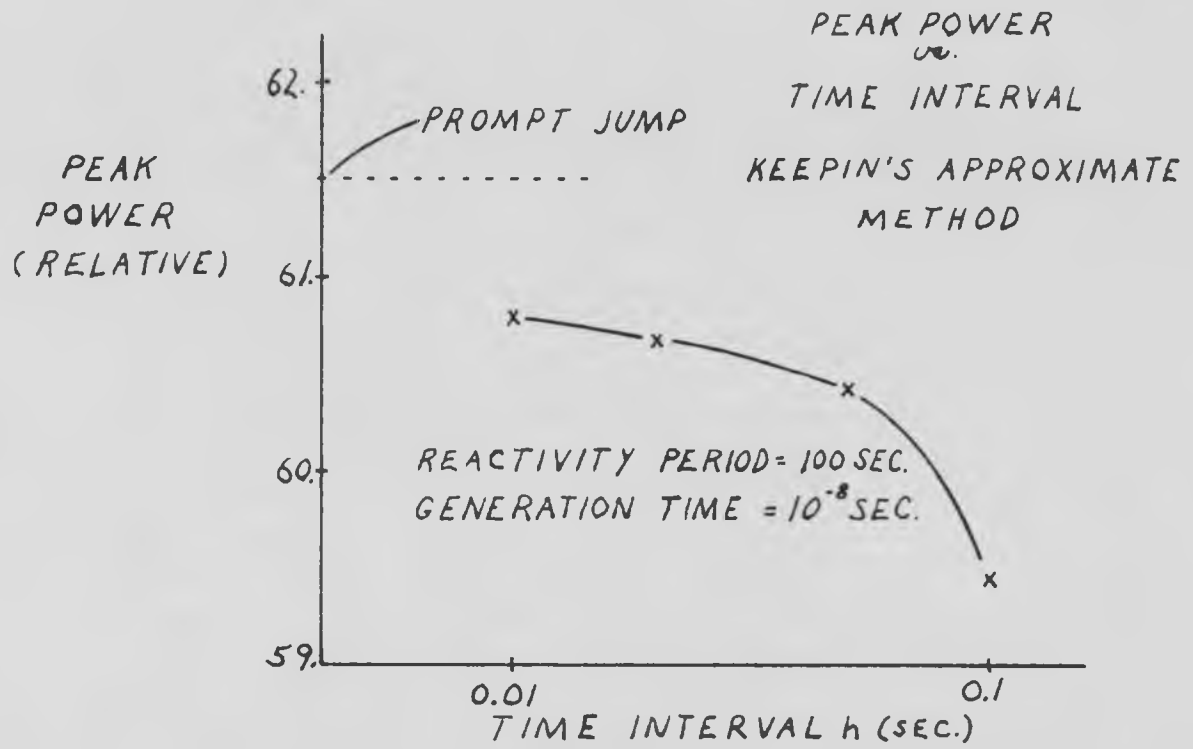

Fig. 13. --Convergence of Keepin's Approximate Method with Time Interval

TABLE 22

DATA FOR KEEPIN'S APPROXIMATION

Reactivity Period = 100 sec.

	Time Increment (sec.)	Peak Power (relative)	Time of Peak Power (sec.)	Running Time
Generation Time $= 10^{-5}$ sec.				
*(Exact Keepin)				
**One Iteration	0.1	46.77348	39.5	7:36
**Convergence = .99999	0.1	46.77350	39.5	7:37
Generation Time $= 10^{-8}$ sec.				
(Approximate Keepin)				
**One Iteration	0.1	34.56708	39.2	:49
**Convergence = .99999	0.01	60.80332	39.11	8:50
	0.02	60.70177	39.12	
	0.05	60.41129	39.15	1:51
	0.1	59.45542	39.2	:55

* See page 58.

** See page 32.

CONCLUSIONS

The spirit of the techniques discussed here lies in representing the slowly-varying factor inside an integral by an assumed known function, instead of using a standard numerical integration formula for the complete integral. For example, consider:

$$I(h) = \int_0^h F(t_0 + \tau)\, e^{\alpha \tau}\, d\tau$$

$F(t_0 + \tau)$ = slowly varying portion of integrand

$\alpha = \frac{\beta - \rho_0}{\ell}$ or $\frac{\beta}{\ell}$ or $\frac{1}{\ell}$

I(h) = integral derived from quasi-linear forms or from Laplace transform of kinetics equations.

Now if

$$F(t_0 + \tau) = F(t_0)\,(1 + A\,\tau + B\,\tau^2 + C\,\tau^3 \ldots .)$$

the integral can be evaluated, thereby eliminating the fast transient difficulty caused by $e^{\alpha \tau}$.

In the methods of Cohen and Adler, only the first term was used here, together with iterations to improve the prediction. This was also used in Keepin's method as modified here; the original RTS code itself is severely limited by the time-constant difficulty. The Collocation method uses three terms of the series, with the coefficients evaluated by requiring an integral equation to be satisfied at two points

in each time interval. Hansen's method is essentially the same, using an assumed exponential for F instead of a polynomial.

The graph presented in Figure 10 shows that the Cohen method, while converging strongly on the prompt jump solution, requires a particularly small time increment to achieve convergence. Since such a small interval requires excessive computer running time, this approach will not be given any further consideration. The probable reason for the necessity of such small time increments is that Cohen's approximation works directly on the basic integral forms derived from the quasilinear differential equations. These forms contain both $n(t_o + \tau)$ and $C(t_o + \tau)$, and both terms have to be approximated in the time interval.

Adler's further integration of the equations eliminates the $C(t_o + \tau)$ term necessitating the use of an approximate form for only $n(t_o + \tau)$.

Due to its synthesis through Laplace transform, the Keepin formulation also proved to be unwieldy with respect to computer time (see Table 22 for specific running times). The Keepin equation, as derived earlier, requires a completely new calculation for reactor power at each point in time. All the other methods could be handled by recurrence relations:

$$n(1) = n(o) + \Delta n \Big|_0^1$$

$$n(2) = n(1) + \Delta n \Big|_1^2$$

$$n(3) = n(2) + \Delta n \Big|_2^3$$

where n(x) is the power at time "x". Keepin's technique requires complete recalculation at each point in time. From the Keepin equation

$$n(1) = n(0) + \bar{\rho n} \Big|_0^1 \left[A_1 h + \frac{A_2}{\alpha} e^{-\alpha(0) h} (1 - e^{-\alpha h})\right]$$

$$n(2) = n(0) + \bar{\rho n} \Big|_0^1 \left[A_1 h + \frac{A_2}{\alpha} e^{-\alpha(1)h} (1 - e^{-\alpha h})\right]$$

$$+ \bar{\rho n} \Big|_1^2 \left[A_1 h + \frac{A_2}{\alpha} e^{-\alpha(0)h} (1 - e^{-\alpha h})\right]$$

$$n(3) = n(0) + \bar{\rho n} \Big|_0^1 \left[A_1 h + \frac{A_2}{\alpha} e^{-\alpha(2)h} (1 - e^{-\alpha h})\right]$$

$$+ \bar{\rho n} \Big|_1^2 \left[A_1 h + \frac{A_2}{\alpha} e^{-\alpha(1)h} (1 - e^{-\alpha h})\right]$$

$$+ \bar{\rho n} \Big|_2^3 \left[A_1 h + \frac{A_2}{\alpha} e^{-\alpha(0)h} (1 - e^{-\alpha h})\right].$$

As can be seen, n(3) is not related to n(2) in any convenient way by the Keepin approach. In the Adler method, for example, only the Δ n is computed at each step and then added to the power computed for the previous time step. Therefore, Keepin's equations, along with Cohen's will not be given further consideration here.

The three approaches left are Adler's, Collocation and Hansen's. The results obtained for the three methods are compared in Figures 14-17.

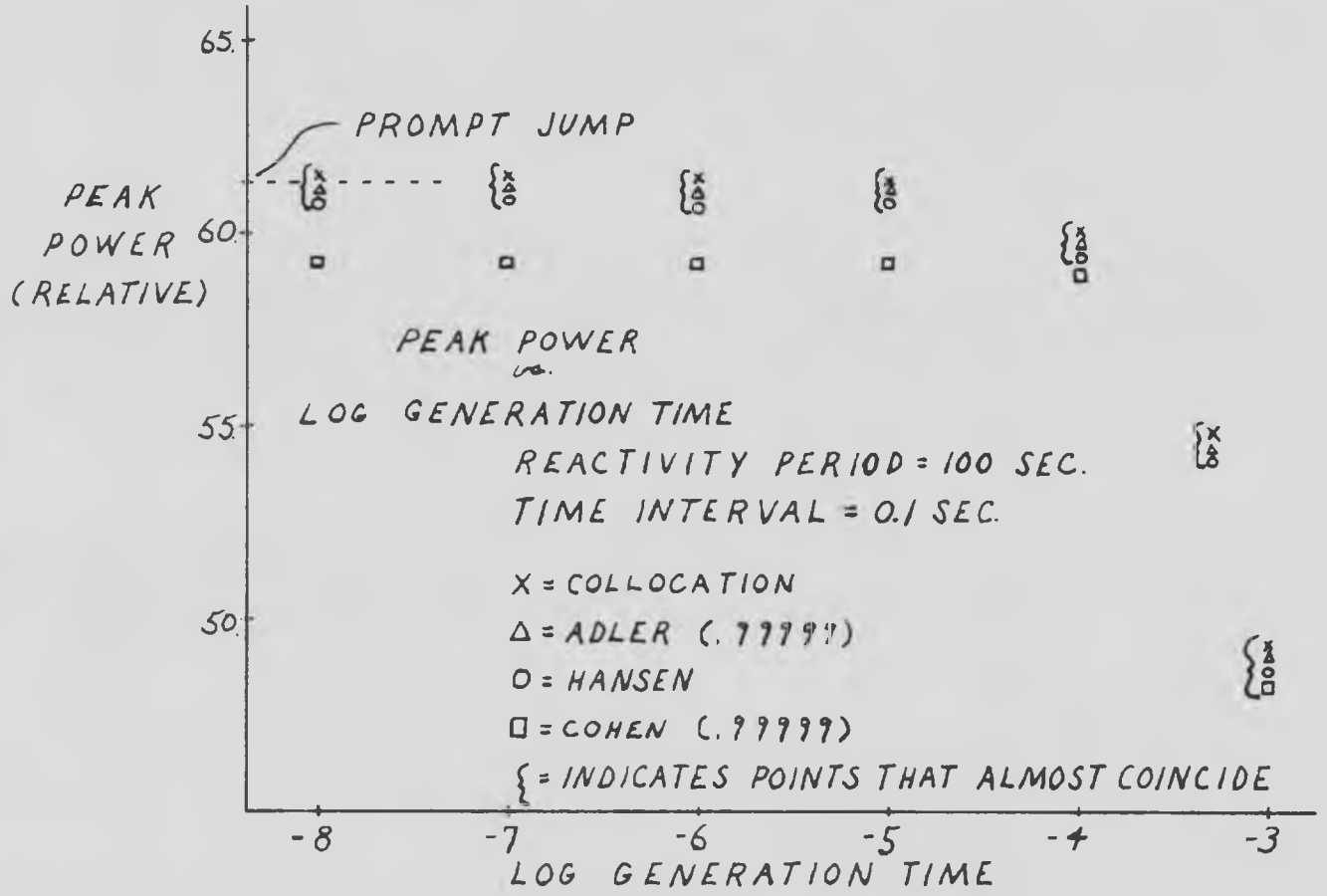

Fig. 14. --Comparison of Collocation, Adler, Hansen and Cohen

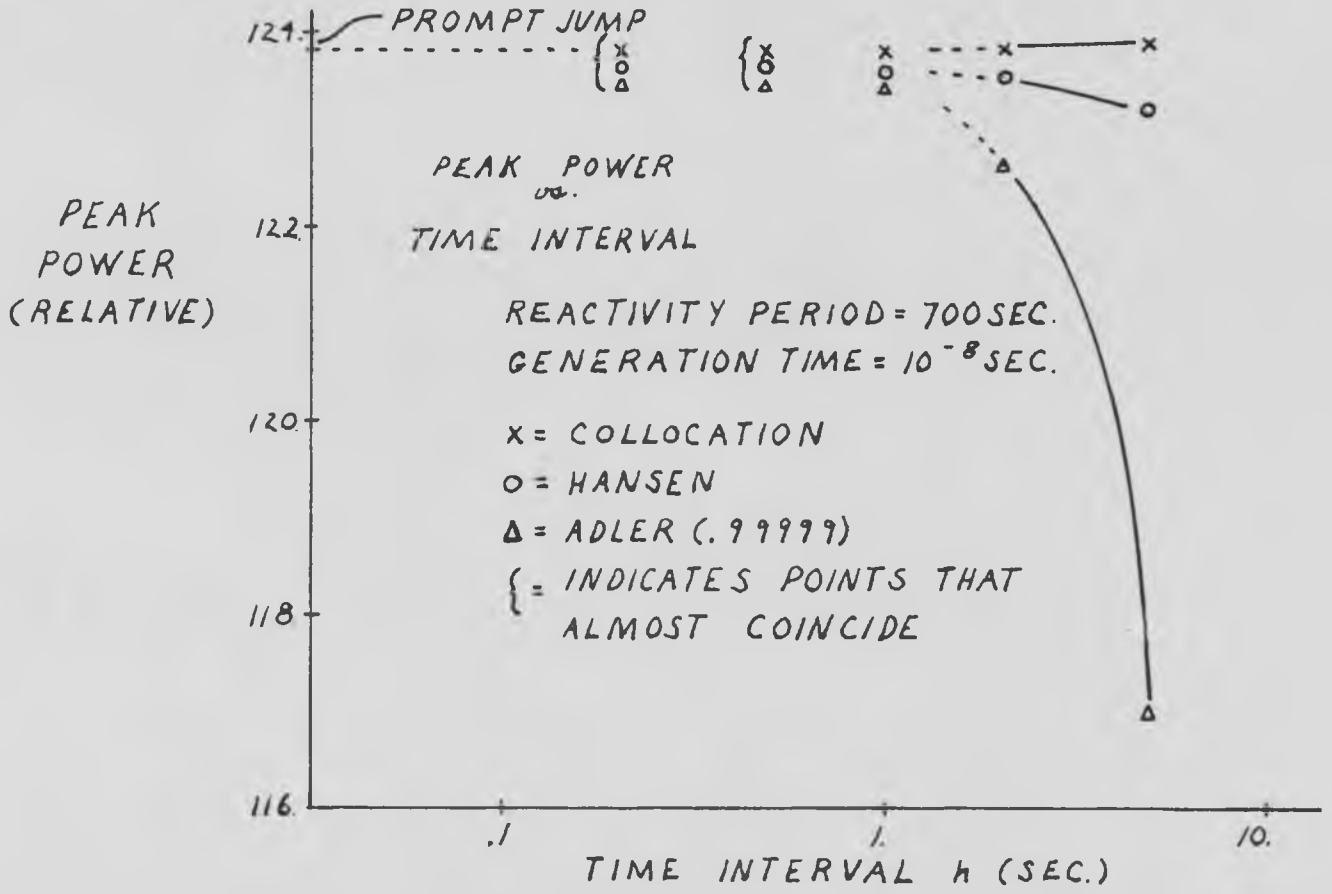

Fig. 15. --Comparative Convergence of the Collocation, Hansen, and Adler Techniques

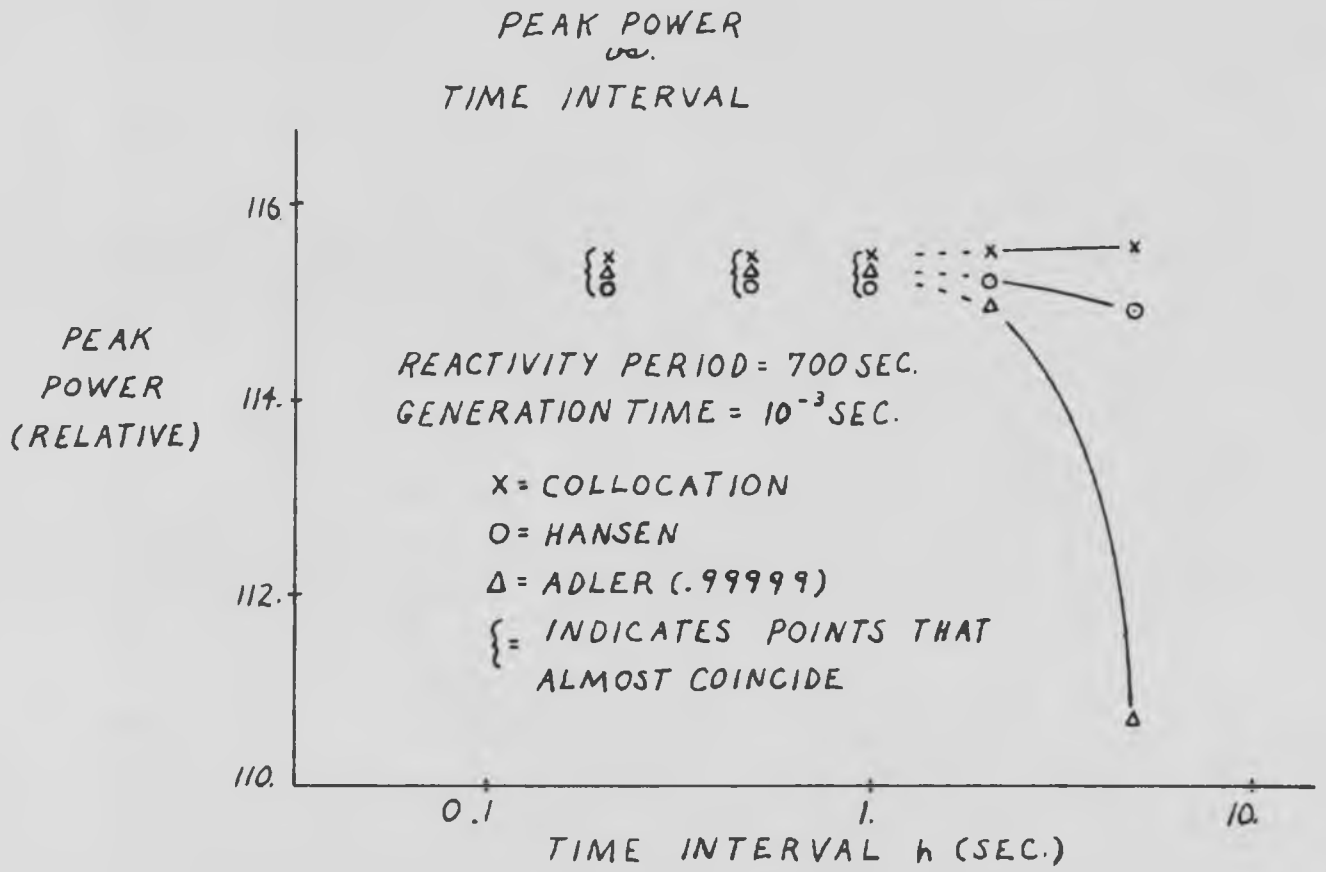

Fig. 16.--Comparative Convergence of a Generation Time of 10^{-3} Sec.

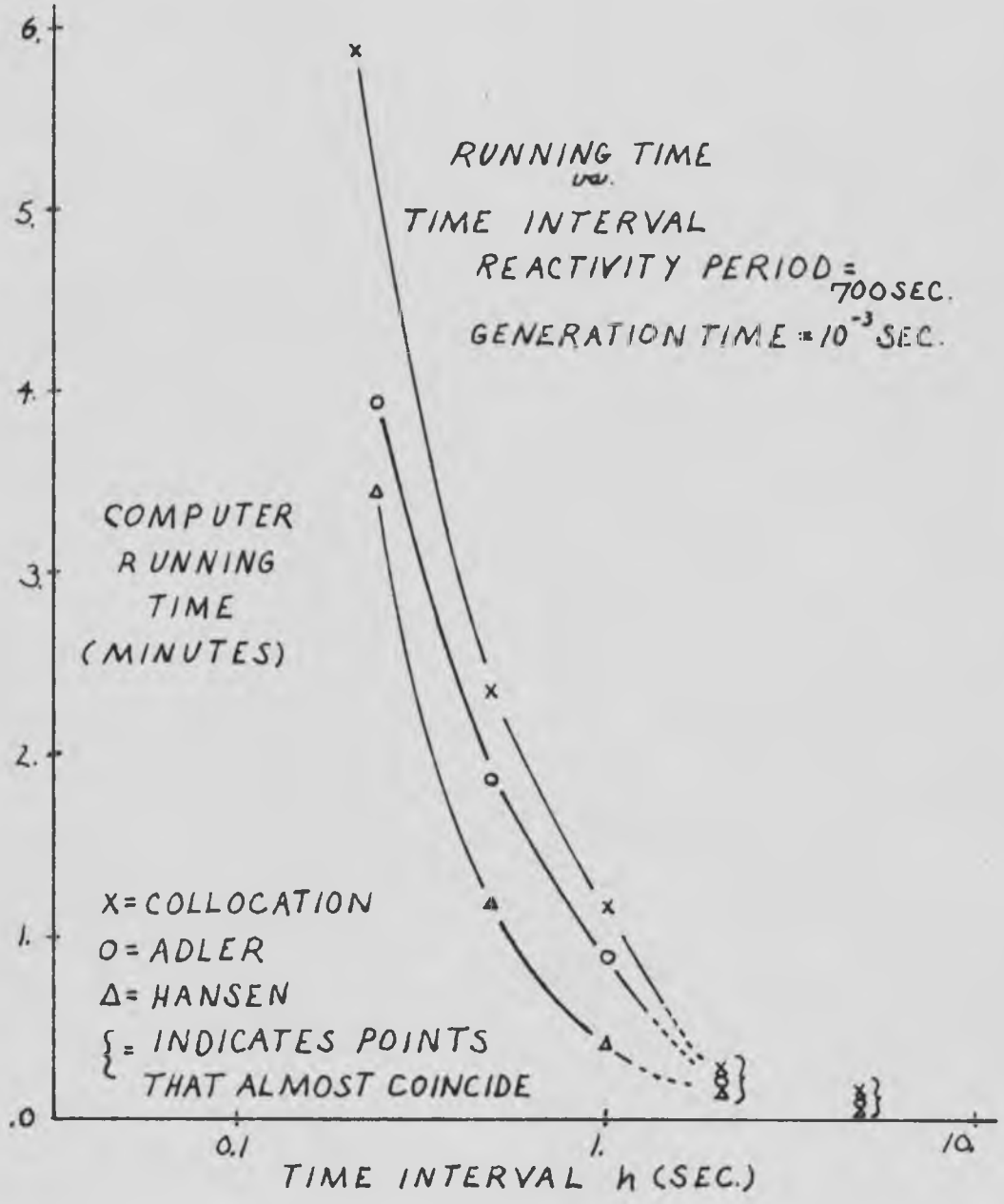

Fig. 17.--Comparison of Computer Times for Collocation, Adler, and Hansen.

The graphs, and all the data obtained, seem to indicate that the Collocation method converges most rapidly and gives the best results when compared to the prompt jump solution for $\ell = 10^{-8}$ sec. Figure 17 shows that Collocation takes slightly longer to run than the others for the same Δt. However, Collocation is closer to the prompt jump with a time increment of 0.5 sec. than are Hansen's and Adler's methods with an interval length of 0.2 sec. for reactivity periods of 500 and 700 sec. Only Adler's method is more accurate for any of the reactivity inputs considered here (reactivity period of 100 sec.).

Hansen's method is not a curve fitting process in the same sense that the other approximations are. It is an exponential fit based on the inhour equation and cannot be expected to be as accurate as the Collocation curve fitting process.

The Adler approach uses only the first term of the power series expansion for $n(t_o + \tau)$. It proves to be the most sensitive to the length of the time interval as seen in Figures 15 and 16. Collocation, on the other hand, is least sensitive to the interval size as seen in the same graphs. Appendices B through H show examples of typical computer programs used to gather the data presented in Tables 1 - 22.

APPENDICES

COMPUTING DETAILS AND COMPUTER PROGRAMS

APPENDIX A

COMPUTING DETAILS

In all cases the one group, source-free reactor kinetic equations were programmed using accepted values for the parameters:

$$\beta = .0079$$

$$\lambda = .077 \text{ sec}^{-1}$$

$$\ell = 10^{-3} \text{ — } 10^{-8} \text{ sec.}$$

Several reactivity inputs were considered. All were sinusoidal with their magnitudes and periods being related in the manner described in the Introduction:

$$\rho_0 = \frac{\beta \omega}{\omega + \lambda}$$

$$\frac{T}{2} = \frac{4}{\omega}$$

Choosing the length of a transient (by selecting T) its period can be found with the above relationships. The following reactivities were selected:

T	ρ_0
50	.0053
150	.0032
250	.0023
350	.0018

The variable T is the half period of the sinusoidal input. In some cases, in an effort to conserve computer time, programs were run for only half a reactivity cycle. The running times for these programs were doubled on the assumption that the computing time would increase by a factor of two if an entire cycle were run. This was demonstrated for a few cases. Those cases run for only half the reactivity cycle are indicated by an asterisk (*) in the data tables.

The times presented in the tables represent computing time plus time to put computed values in tape storage. They do not include the time to print out the data or the time to compile the program. Computing times are, of course, relevant only to the University of Arizona IBM-7072.

Two other very recent approaches were noted after this study was completed. The first,[11] as yet unpublished, appears to be an improvement on the Runge-Kutta method, but details were not available at the time of writing. The second[12] is related to Hansen's method, but uses rational matrix functions to approximate the exponential matrix which expresses the state variables at any time in terms of their values at an earlier time.

APPENDIX B

COMPUTER PROGRAM USED FOR DIRECT FINITE DIFFERENCE

```
      SUBROUTINE TIME (LOCX)
      DIMENSION T(3)
      IMASK = 000000001000
      TEMP1 = 0
      RDT=0.
A                 XU    80,TEMP1
A                 ZAI   +0
A                 STI   80
A                 XZA   80,T
A                 ZAP   80
A                 ZAI   9992(2,5)
A                 AI    +1
A                 AI    9992
A                 MSM   9991
A                 STI   RDT
A                 PC    IMASK
A                 TR    25,RDT
A                 BCB   2,*
A                 TLF   25
A                 XZA   80,LOCX
A                 CAM   80,RDT
A                 XI    80,TEMP1
      RETURN
      END

      CALL TIME (II)
      PRINT 20, II
   20 FORMAT (1X110)
      X = .01
      P = 50.
      AMDA = .077
      B = .077
      C = 8.*P/(8.+AMDA*P)
      EE = .00002
      T = 0.
      RUX = 1.
      RUXO = 1.
      CRUX = 0.
      DO 12 J = 1,1000
      DO 13 K = 1,10
      T = T+X
      RHO = C*SIN(3.1416/8*(T-X))
      POW = X*((RHO-P)/EE+RUX+AMDA*CRUX)+RUX
```

```
      COW = X**(5/EE*RUX-AMDA*CRUX)+CRUX
      CRUX = COW
      RUX = POW
   12 CONTINUE
      PRINT 3, T, POW, COW, RHO, EE
    3 FORMAT (5F20.5)
   12 CONTINUE
      CALL TIME (JJ)
      PRINT 30, JJ
   30 FORMAT (1X110)
      STOP
      END
```

APPENDIX C

COMPUTER PROGRAM USED FOR RUNGE-KUTTA APPROXIMATION

```
      SUBROUTINE TIME (LOCX)
      DIMENSION T(3)
      IMASK = 000000001000
      TEMP1 = 0
      RDT=9.
A                 XU    80,TEMP1
A                 ZA1   +0
A                 STI   80
A                 XZA   80,T
A                 ZA2   80
A                 ZA1   9992(2,5)
A                 A1    +1
A                 A1    9992
A                 MSM   9991
A                 STI   RDT
A                 PC    IMASK
A                 TP    25,RDT
A                 BCB   2,*
A                 TLF   25
A                 XZA   *0,LOCX
A                 EAN   80,RDT
A                 XL    80,TEMP1
      RETURN
      END

      CALL TIME (II)
      PRINT 20, II
   20 FORMAT (1X110)
      AMDA = .077
      B = .0079
      P = 50.
      PP = 3.1416/P
      C = B*B./(P.+AMDA*P)
      EC = .00005
      X = .01
      T = 0.
      POW = 0.
      RUX = 1.
      CRUX = B/(EC*AMDA)
      DO 1 I = 1,1000
      DO 2 II = 1,10
      T = T+X
      RHO = C*SINF(PP*(T-X))
      PHA = C*SINF(PP*(T-X/2.))
```

```
      RHB = C*SINF(PP*T)
      Z = ((RHO-B)/EE*RUX+AMDA*CRUX)*X
      Y = (B/EE*RUX-AMDA*CRUX)*X
      ZZ = ((RHA-B)/EE*(RUX+Z/2.)+AMDA*(CRUX+Y/2.))*X
      YY = (B/EE*(RUX+Z/2.)-AMDA*(CRUX+Y/2.))*X
      ZZZ = ((RHA-B)/EE*(RUX+ZZ/2.)+AMDA*(CRUX+YY/2.))*X
      YYY = (B/EE*(RUX+ZZ/2.)-AMDA*(CRUX+YY/2.))*X
      ZZZZ = ((RHB-B)/EE*(RUX+ZZZ)+AMDA*(CRUX+YYY))*X
      YYYY = (B/EE*(RUX+ZZZ)-AMDA*(CRUX+YYY))*X
      A = (Z+2.*ZZ+2.*ZZZ+ZZZZ)/6.
      AA = (Y+2.*YY+2.*YYY+YYYY)/6.
      POW = POW+A
      RUX = POW
      CRUX = CRUX+AA
    2 CONTINUE
      PRINT 3, T, POW, A, RHO
    3 FORMAT (4F20.5)
    1 CONTINUE
      CALL TIME (JJ)
      PRINT 30, JJ
   30 FORMAT (1X,I10)
      STOP
      END
```

APPENDIX D

COMPUTER PROGRAM USED FOR HANSEN'S APPROACH

```
      DIMENSION T(3)
      IMASK = 000000001000
      TEMP1 = 0
      RDT=0.
A                 XU    80,TEMP1
A                 ZA1   +0
A                 ST1   80
A                 XZA   80,1
A                 ZA2   80
A                 ZA1   9992(2,5)
A                 AI    +1
A                 A1    9992
A                 MSM   9991
A                 ST1   RDT
A                 PC    IMASK
A                 TR    25,RDT
A                 BCD   2,*
A                 TLF   25
A                 XZA   80,LOCX
A                 EAM   80,RDT
A                 XL    80,TEMP1
      RETURN
      END

      CALL TIME (1)
      PRINT 20, I
   20 FORMAT (1X,I10)
      AMDA = .077
      B = .0079
      EE = .0005
      A = .1
      P = 50.
      C = 8./(3.+.077*P)*B
      T = 0.
      RUXO = 1.
      CRUXO = B/(EE*AMDA)
      DEL = 2.*P/A
      JERK = DEL
      DO 1 I = 1,JERK
      T = T+A
      RHO = C/2.*(SINF(3.1416/P*T)+SINF(3.1416/P*(T-A )))
      X = (RHO-B)/EE
    2CPAR=(-RHO/EE+AMDA+L/EE+SQRTF((RHO/EE-AMDA-B/EE)**2+4.*
     1AMDA*RHO/EE))/(-2.)
      PER = -AMDA*RHO/(EE*PAR)
```

```
    4 PO =RUXO*EXPEF(X*A)+CRUXO*(EXPEF(PER*A)-EXPEF(X*A))*AM
     1DA/(PER-X)
    8 CRUXO=CRUXO*EXPEF(-AMDA*A)+RUXO*(EXPEF(PER*A)-EXPEF(-A
     1MDA*A))*D/EF*1./(PER+AMDA)
      RUXO = PO
      PRINT 5, T, PO, A, P, PER, EF
    5 FORMAT (6F19.5)
    1 CONTINUE
      CALL TIME (J)
      PRINT 30, J
   30 FORMAT (1X110)
      STOP
      END
```

APPENDIX E

COMPUTER PROGRAM USED FOR COHEN'S METHOD

```
      SUBROUTINE TIME (LOCX)
      DIMENSION T(3)
      IMASK = 000000001000
      TEMP1 = 0
      RDT=0.
A               XC    80,TEMP1
A               ZAI   +0
A               STI   80
A               XZA   80,T
A               ZA2   80
A               ZAI   9992(2,5)
A               AI    +1
A               AI    9992
A               MSM   9991
A               STI   RDT
A               PC    IMASK
A               TR    25,RDT
A               BCB   2,*
A               TLF   25
A               XZA   80,LOCX
A               EAN   80,RDT
A               XL    80,TEMP1
      RETURN
      END

      CALL TIME (I)
      PRINT 20, I
   20 FORMAT (1X110)
      DIMENSION RON(2)
      EE = .000000001
      P = 50.
      AMDA = .077
      R = .0079
      PP = 3.1416/P
      RHO = 0.5*./(P.+P*AMDA)
      DIMENSION RR(2)
      RR(1) = .01
      RR(2) = .0075
      DO 14 KK = 1,2
      T = 0.
      X = RR(KK)
      RUXO = 1.
      RUX = 1.
```

```
      CRUXO = EF*RUXO/(EF*AMDA)
      CRUX = CRUXO
      BBB = EXP(EF(-AMDA*X)
      DEL = P/X
      JERK = DEL
      DO 100 I = 1,JERK
      T = T+X
      A = RHO*SINF(PP*(T-X))
      ZZ = RHO*(SINF(PP*T)+SINF(PP*(T-X)))/2.
   17 BB = 0.
   18 C = EF/(E-A)
    5 POW(1) = RUXO*BB+(RUX*(ZZ-A)/EF+AMDA*CRUX)*C*(1.-BB)
      RUX = (RUXO+POW(1))/2.
      CRUX=CRUXO*BBB+E/(EF*AMDA)*RUX*(1.-PPP)
      CRUX = (CRUX+CRUXO)/2.
      POW(2) = RUXO*BB+(RUX*(ZZ-A)/EF+AMDA*CRUX)*C*(1.-BB)
      RUX = (RUXO+POW(2))/2.
      CRUX=CRUXO*BBB+E/(EF*AMDA)*RUX*(1.-PPP)
      CRUA = CRUX
      CRUX = (CRUX+CRUXO)/2.
      IF(POW(1)-POW(2)) 6,8,4
    6 IF(POW(1)/POW(2)-.99999) 5,8,8
    4 IF(POW(2)/POW(1)-.99999) 5,8,8
    8 PRINT 3, T, POW(2), POW(1), CRUX, ZZ, C
    3 FORMAT (6F19.5)
      RUXO = POW(2)
      RUX = POW(2)
      CRUXO = CRUA
      CRUX = CRUA
  100 CONTINUE
      CALL TIME (J)
      PRINT 30, J
   30 FORMAT (1X110)
   14 CONTINUE
      STOP
      END.
```

APPENDIX F

COMPUTER PROGRAM USED FOR ADLER'S METHOD

```
      SUBROUTINE TIME (LOCX)
      DIMENSION T(3)
      IMASK = 000000100
      TEMP1 = 0
      RDT=0.
A                 XU    80,TEMP1
A                 ZA1   +0
A                 STI   80
A                 XZA   80,T
A                 ZA2   80
A                 ZA1   9992(2,5)
A                 A2    +1
A                 A1    9992
A                 MSM   9991
A                 STI   RDT
A                 PC    IMASK
A                 TR    25,RDT
A                 BCB   2,*
A                 TLF   25
A                 XZA   80,LOCX
A                 EAN   80,RDT
A                 XL    80,TEMP1
      RETURN
      END

      CALL TIME (I)
      PRINT 20, I
   20 FORMAT (1X,I10)
      E = .1
      EE = .00001
      AMDA = .077
      X = .1
      B = .0079
      P = 50.
      PP = 3.1416/P
      DIMENSION POW(2)
      RHO = B*8./(8.+P*AMDA)
      K = 2
      DO 31 JJ = 1,K
      EE = EE*E
      T = 0.
      PUXO = 1.
      CPUXO = B/(EE*AMDA)
```

```
      RUX = 1.
      DEL = P/X
      JERK = DEL
      DO 10 I = 1,JERK
      T = T+X
      A = (P-RHO*SINF(PP*(T-X)))/EE
      ZZ = 0.
   14 BB = EXPEF(-AMDA*X)
      BBB = (SINF(PP*T)+SINF(PP*(T-X)))/2.
   50POW(1)=RUXO*ZZ+RHO*(BBB-SINF(PP*(T-X)))*RUX/(EE*A)*(1.-
     1ZZ)+CRUXO*AMDA/(A-AMDA)*(BB-ZZ)+AMDA*B/(EE*(A-AMDA))*R
     2UX*(1./AMDA-BB/AMDA+ZZ/A-1./A)
      RUX = (RUXO+POW(1))/2.
  550POW(2)=RUXO*ZZ+RHO*(BBB-SINF(PP*(T-X)))*RUX/(EE*A)*(1.-
     1ZZ)+CRUXO*AMDA/(A-AMDA)*(BB-ZZ)+AMDA*B/(EE*(A-AMDA))*R
     2UX*(1./AMDA-BB/AMDA+ZZ/A-1./A)
      RUX = (RUXO+POW(2))/2.
      IF(POW(1)-POW(2)) 6,8,4
    6 IF(POW(1)/POW(2)-.99999) 5,8,8
    4 IF(POW(2)/POW(1)-.99999) 5,8,8
    8 PRINT 3, T, POW(2), POW(1), A, BBB, CRUXO
    3 FORMAT (6E19.5)
      RUXO = POW(2)
      CRUXO = CRUXO*BB+B/EE*RUX*(1./AMDA-BB/AMDA)
      RUX = POW(2)
   10 CONTINUE
      CALL TIME (J)
      PRINT 30, J
   30 FORMAT (1X110)
   31 CONTINUE
      STOP
      END
```

APPENDIX G

COMPUTER PROGRAM USED FOR THE COLLOCATION APPROXIMATION

```
      SUBROUTINE TIME (LOCX)
      DIMENSION T(3)
      IMASK = 0000001000
      TEMP1 = 0
      RDT=0.
A                 XU    80,TEMP1
A                 ZA1   +0
A                 ST1   80
A                 XZA   80,T
A                 ZA2   80
A                 ZA1   9992(2,5)
A                 AI    +1
A                 A1    9992
A                 MSM   9991
A                 STI   RDT
A                 PC    IMASK
A                 TR    25,RDT
A                 BCB   2,*
A                 TLF   25
A                 XZA   80,LOCX
A                 EAN   80,RDT
A                 XL    80,TEMP1
      RETURN
      END

      CALL TIME (I)
      PRINT 20, I
   20 FORMAT (1X110)
      EE = .001
      B = .0079
      AMDA = .077
      X = .2
      P = 250.
      PP = 3.1416/P
      C = 8.*B/(8.+AMDA*P)
   13 T = 0.
      RUX = 1.
      CRUX = B/(EE*AMDA)
      DEL = P/X
      JERK = DEL
      DO 1 I = 1,JERK
      T = T+X
      PHO = C*SINF((T-X)*PP)
      RHA = C*(SINF(T*PP)+SINF((T-X)*PP))/2.
```

```
      RHB = C*(SINF((T-X/2.)*PP)+SINF((T-X)*PP))/2.
      BB = (B-RHO)/EE
      Y = EXPEF(-AMDA*X)
    4 YY = EXPEF(-BB*X)
   28 D=-RUX*X+RUX*((RHA-RHO)/EE-AMDA*B/(EE*(BB-AMDA)))*((BB
     1*Y-1.)/BB**2+YY/(BB**2))+RUX*AMDA
     2*B/(EE*(BB-AMDA))*((AMDA*X-1.)/(AMDA**2)+Y/(AMDA**2))
      DD=-RUX*(X**2)+RUX*((RHA-RHO)/EE-AMDA*B/(EE*(BB-AMDA))
     1)*(((BB**2)*(X**2)-2.)/(BB**3
     2)-2.*YY/(BB**3))+RUX*AMDA*B/(EE*(BB-AMDA))*((AMDA**2*X
     3**2-2.*AMDA*X+2.)/(AMDA**3)-2.*Y/(AMDA**3))
      DDD=(-RUX+RUX*YY+EE*AMDA*CRUX/(B-RHO-AMDA*EE)*(Y-YY)+R
     1UX*((RHA-RHO)/EE-AMDA*B/(EE*(BB-AMDA)))
     2*(1./BB-YY/BB)+RUX*AMDA*B/(EE*(BB-AMDA))*(1./AMDA-Y/AM
     3DA))
      RHA=RHB
      XX = X/2.
      X = XX
      Y = EXPEF(-AMDA*X)
    6 YY = EXPEF(-BB*X)
   29 G=-RUX*X+RUX*((RHA-RHO)/EE-AMDA*B/(EE*(BB-AMDA)))*((BB
     1*X-1.)/BB**2+YY/(BB**2))+RUX*AMDA
     2*B/(EE*(BB-AMDA))*((AMDA*X-1.)/(AMDA**2)+Y/(AMDA**2))
      GG=-RUX*(X**2)+RUX*((RHA-RHO)/EE-AMDA*B/(EE*(BB-AMDA))
     1)*(((BB**2)*(X**2)-2.)/(BB**3
     2)-2.*YY/(BB**3))+RUX*AMDA*B/(EE*(BB-AMDA))*((AMDA**2*X
     3**2-2.*AMDA*X+2.)/(AMDA**3)-2.*Y/(AMDA**3))
      GGG=(-RUX+RUX*YY+EE*AMDA*CRUX/(B-RHO-AMDA*EE)*(Y-YY)+R
     1UX*((RHA-RHO)/EE-AMDA*B/(EE*(BB-AMDA)))
     2*(1./BB-YY/BB)+RUX*AMDA*B/(EE*(BB-AMDA))*(1./AMDA-Y/AM
     3DA))
      A = ((-DDD)*GG-DD*(-GGG))/(D*GG-DD*G)
      AA = (D*(-GGG)-(-DDD)*G)/(D*GG-DD*G)
      X = XX*2.
      POW = RUX*(1.+A*X+AA*X**2)
      PRINT 3, T, POW, RHA, D, DD
    3 FORMAT (5F12.5)
      Z = X*AMDA
      CRUX = CRUX*EXPEF(-Z)+B/EE*RUX*(1./AMDA+A*(X/AMDA-1./A
     1MDA**2)+AA
     2*(X**2/AMDA-2.*X/AMDA**2+2./AMDA**3)+EXPEF(-Z)*(-1./AM
     3DA+A/AMDA**2-2.*AA/AMDA**3)
      RUX = POW
    1 CONTINUE
      CALL TIMF (J)
      PRINT 30, J
   30 FORMAT (1X,I10)
      STOP
      END
```

APPENDIX H

COMPUTER PROGRAM USED FOR KEEPIN'S METHOD

```
      SUBROUTINE TIME (LOCX)
      DIMENSION T(3)
      IMASK = 000000001000
      TEMP1 = 0
      RDT=0.
A                 XU    80,TEMP1
A                 ZAI   +0
A                 STI   80
A                 XZA   80,1
A                 ZA2   80
A                 ZAI   9992(2,5)
A                 AI    +1
A                 AI    9992
A                 MSM   9991
A                 STI   RDT
A                 PC    IMASK
A                 TR    25,RDT
A                 BCR   2,*
A                 TLF   25
A                 XZA   80,LOCX
A                 EAN   80,RDT
A                 XL    80,TEMP1
      RETURN
      END

      CALL TIME (II)
      PRINT 20, II
   20 FORMAT (1X110)
      DIMENSION RHA(1000)
      DIMENSION Z(1000)
      DIMENSION RUX(1000)
      DIMENSION POWW(2)
      RUX0 = 1.
      B = .0079
      AMDA = .077
      EE = .001
      P = 50.
      C = B.*B/(8.+AMDA*P)
      PP = 3.1416/P
      X = .1
      A = AMDA/(EE*(AMDA+B/EE))
      AA = B/(EE*(AMDA*EE+B))
      AAA = AMDA+B/EE
```

```
      RUX(1) = 1.
      RUXA = 1.
      T = 0.
      DO 1 KK = 1,1000
      T = T+X
      RHA(KK)=C/2.*(SINF((T-X)*PP)+SINF(T*PP))
      IF(T-10.) 4,7,7
    4 Z(KK) = EXPEF(-(T-X)*AAA)
      GO TO 1
    7 Z(KK) = 0.
    1 CONTINUE
      ZZ = EXPEF(-AAA*X)
      D = 0.
      DO 5 LLL = 1,1000
      D = D+X
      JERK = LLL
      WO = 0.
      DO 2 K = 1,JERK
      L = JERK-K
      WOP = RUX(K)*RHA(K)*(A*X+AA/AAA*Z(L)*(1.-ZZ))
      WO = WO+WOP
    2 CONTINUE
      POW = RUXO+WO
      RUX(K) = (POW+RUXA)/2.
   15 POWW(1) = POW-WOP+RUX(K)*RHA(K)*(A*X+AA/AA/*Z(L)*(1.-Z
     1ZI)
      RUX(K) = (POWW(1)+RUXA)/2.
      POWW(2) = POW-WOP+RUX(K)*RHA(K)*(A*X+AA/AAA*Z(L)*(1.-Z
     1Z))
      RUX(K) = (POWW(2)+RUXA)/2.
      IF(POWW(1)-POWW(2)) 12,13,14
   12 IF(POWW(1)/POWW(2)-.99999) 15,13,13
   14 IF(POWW(2)/POWW(1)-.99999) 15,12,13
   13 I = K+1
      RUX(I) = POWW(2)
      RUXA = POWW(2)
      PRINT 3, D, POWW(2)
    3 FORMAT (2F20.5)
    5 CONTINUE
      CALL TIME (III)
      PRINT 30, III
   30 FORMAT (1X110)
      STOP
      END
```

LIST OF REFERENCES

1. Hetrick, D. L. Nonlinear Reactor Dynamics. Unpublished Lecture Notes, 1964, p. 124.

2. Ibid., p. 126.

3. Ibid., p. 39.

4. Margenau, Henry, and George Murphy. The Mathematics of Physics and Chemistry. New York: D. Van Nostrand Company, 1943, p. 472.

5. Hansen, K. F., B. V. Koen and W. W. Little, Jr. "Stable Numerical Solutions of the Reactor Kinetics Equation," Nuclear Science and Engineering, Vol. 22, 1965, pp. 51-59.

6. Cohen, E. Richard. "Some Topics in Reactor Kinetics," Proc. Geneva Conf., Vol. 11, 1958, p. 302.

7. Adler, F. T. "Reactor Kinetics: Integral Equation Formulation," Reactor Science and Technology (Journal of Nuclear Energy, Parts A/B), Vol. 15, 1961, p. 81.

8. Flatt, H. P. "The Numerical Solution of the Reactor Kinetics Equations by the Method of Collocation," Nuclear Computing Technical Bulletin, Bulletin No. 5 (August 10, 1962). Los Angeles, California: IBM Data Processing Division.

9. Kaganove, J. J. "Numerical Solution of the One-Group, Space-Independent Reactor Kinetics Equations for Neutron Density Given the Excess Reactivity," ANL-6132 (February, 1960).

10. Keepin, G. R., and C. W. Cox. "General Solution of the Reactor Kinetic Equations," Nuclear Science and Engineering, Vol. 8 (1960), pp. 670-690.

11. Gear, C. W. Personal Communication, May 16, 1966. (To be published as ANL-7126)

12. Porsching, T. A. "Numerical Solution of the Reactor Kinetics Equations by Approximate Exponentials," Nuclear Science and Engineering, Vol. 25 (1966), pp. 183-188.