JAERI-M
6351

A DIRECT METHOD FOR NUMERICAL SOLUTION OF A CLASS OF NONLINEAR VOLTERRA INTEGRO-DIFFERENTIAL EQUATIONS AND ITS APPLICATION TO THE NONLINEAR FISSION AND FUSION REACTOR KINETICS

December 1975

Y. NAKAHARA, T. ISE, K. KOBAYASHI and Y. ITOH*

日本原子力研究所
Japan Atomic Energy Research Institute

この報告書は，日本原子力研究所が JAERI-M レポートとして，不定期に刊行している研究報告書です．入手，複製などのお問合わせは，日本原子力研究所技術情報部（茨城県那珂郡東海村）あて，お申しこしください．

JAERI-M reports, issued irregularly, describe the results of research works carried out in JAERI. Inquiries about the availability of reports and their reproduction should be addressed to Division of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken, Japan.

JAERI-M 6351

A Direct Method for Numerical Solution of a Class of Nonlinear Volterra Integro-Differential Equations and its Application to the Nonlinear Fission and Fusion Reactor Kinetics

Yasuaki NAKAHARA, Takeharu ISE
Kensuke KOBAYASHI and Yasuyuki ITOH*

Division of Reactor Engineering, Tokai, JAERI
(Received December 9, 1975)

A new method has been developed for numerical solution of a class of nonlinear Volterra integro-differential equations with quadratic nonlinearity. After dividing the domain of the variable into subintervals, piecewise approximations are applied in the subintervals. The equation is first integrated over a subinterval to obtain the piecewise equation, to which six approximate treatments are applied, i.e. fully explicit, fully implicit, Crank-Nicolson, linear interpolation, quadratic and cubic spline. The numerical solution at each time step is obtained directly as a positive root of the resulting algebraic quadratic equation.

The point reactor kinetics with a ramp reactivity insertion, linear temperature feedback and delayed neutrons can be described by one of this type of nonlinear Volterra integro-differential equations. The algorithm is applied to the Argonne benchmark problem and a model problem for a fast reactor without delayed neutrons.

The fully implicit method has been found to be unconditionally stable in the sense that it always gives the positive real roots. The cubic spline method is divergent, and the other fourmethods are intermediate in between. From the estimation of the stability, convergency, accuracy and CPU time, it is concluded that the Crank-Nicolson method is best, then the linear interpolation method comes closely next to it.

Discussions are also made on the possibility of applying the algorithm to the fusion reactor kinetics in the form of a nonlinear partial differential equation.

---

* Department of Nuclear Engineering, Osaka University
Y. Itoh jointed in the present work during his summer stay at JAERI from July 23 to Aug. 30, 1975.

JAERI－M 6351

非線型ボルテラ型微積分方程式の直接数値解法とその非線型核分裂炉及び核融合炉動特性への応用

日本原子力研究所東海研究所原子炉工学部
中原康明，伊勢武治，小林健介，伊藤保之*

（1975年12月9日受理）

2次の非線型性を持つ非線型ボルテラ型微積分方程式の新しい数値解法が開発された。変数の領域を部分区間に分割し，この部分区間毎に区間別近似を適用する。まず，部分区間について方程式を積分して，区間別方程式を求め，これに陽近似，陰近似，クランク・ニコルソン法，1次内挿法及び2次と3次のスプライン法の6通りの近似的取扱いを試みた。各時間ステップ毎の数値解は得られた2次代数方程式の正の実根として直接求められる。

線型反応度投入，温度フィードバック及び遅発中性子のある場合の一点炉動特性は，ここで想定している型の非線型ボルテラ型微積分方程式で記述することができる。我々のアルゴリズムをアルゴンヌのベンチマーク問題と高速炉のモデル問題に適用した。いずれの場合も遅発中性子は無視してある。

陰近似は常に正の実根を与えるという意味で無条件に安定であることが分った。3次のスプライン法は発散する。他の4通りの近似法はこれらの中間に位置する。安定性，収束性，精度及びＣＰＵ時間の評価から最も良いのはクランク・ニコルソン法で，1次内挿法はわずかな差でこれに続くとの結論が得られた。

さらに，我々のアルゴリズムを，非線型偏微分方程式で記述できる核融合炉動特性問題に適用する可能性についても議論を行っている。

---

* 大阪大学工学部原子力工学科
伊藤は夏期実習生として1975年7月23日～8月30日の間東海研に滞在中にこの研究に参加した。

目　　次

Contents

1. Introduction

Recently a new algorithm has been developed for the numerical solution of a class of nonlinear Volterra integro-differential equations with quadratic nonlinearity:

$$\frac{d}{dt}n(t) = F\left(n(t), \int_0^t ds\, K(t,s,n(s)),\ n(t)\int_0^t ds\, n(s)\right), \tag{1}$$

with the initial condition $n(0) = n_1$[1]. In Eq. (1) F(x,y,z,---) means that F is expressed as a linear combination of x, y, z, ---, and K is a linear Volterra kernel, which is assumed to satisfy a uniform Lipschitz condition with respect to n and to be uniformly continuous in t and s.

The point reactor kinetics with a ramp reactivity insertion, linear temperature feedback and delayed neutrons can be described by an equation of this type. In a prompt approximation in which delayed neutron effects are neglected the point reactor kinetics equation can be written in the form:

$$\frac{d}{dt}n(t) = F\left(tn(t),\ n(t)\int_0^t ds\, n(s)\right). \tag{2}$$

Froehlich and Johnson presented an analytical solution to Eq. (2) in a form of a simple integration problem[2]. The accurate numerical values of the solution are given in tables [3], which can be used as a standard benchmark for the evaluation of approximate algorithms.

Iterative methods such as the Newton method and its variations have generally been used in solving numerically

nonlinear integral and differential equations[4]. The uniqueness and existence theorems have been proved for the solutions obtained from these methods. From the practical point of view, however, the iterative methods are rather time consuming, hence not economical, because they demand very fine mesh widths and many iterations to obtain reasonably accurate solutions. Indeed, in an example of the modified Newton method for a nonlinear integral equation of the Hammerstein type, Amann reported that 37 iterations were necessary to reduce the error bound to $O(10^{-3})$[5]. Extensive efforts have been directed by many authors to develop effective algorithms for the numerical solutions both of linear and nonlinear problems.

The piecewise interpolation method is one of the most promising candidates for achieving the success through further developments. Piecewise Lagrangian, Hermitian and spline interpolates have enjoyed successful applications in the linear boundary and initial value problems[6),7].

A twice continuously differentiable cubic spline has been applied by Guzek and Kemper to a linear Volterra integro-differential equation[8]:

$$\frac{d}{dt}n(t) = F\left(t,\ n(t),\ \int_0^t ds\, K(t,s,n(s))\right). \tag{3}$$

They formulated the algorithm with a discretization error of $O(h^3)$ under the requirement to the solution to Eq. (3) to be of class C^4, i.e., continuously differentiable up to the fourth order, where h is the discretized mesh width of the variable t. However, we were unhappy in applying the algorithm

of Guzek and Kemper to the nonlinear equation given by Eq. (2). The results were divergent.

On the other hand, Loscalzo and Talbot have shown that spline approximations to solutions of ordinary differential equations are divergent when spline functions of degree greater than three with full continuity are used.[9] Reportedly, it is noted by Hung that the same divergence occurs when the splines are applied to Volterra integro-differential equations.[10] Recently Kemper has extended the method of Loscalzo and Schoenberg to obtain spline functions of degree $2g$ with g continuous derivatives to approximate solutions of a class of functionlly differential equations, which include Volterra integro-differential equations.[10] The success of these methods depends strongly on the starting values, especially, of the higher derivatives. The algorithm of Kemper contains a sophisticated starting method and may not be amenable to easy programming.

Nonlinear problems require much more careful and ingenious considerations. It is worthwhile to note here the comments of Ortega and Rheinboldt.[4] They say in the preface of their text that they do not discuss iterative methods which require second or higher derivatives in their formulation because the analysis of such methods tends to be uninformative and cumbersome, and more importantly, because the evaluation of the k-th derivative requires in general N^{k+1} functional evaluations for solutions of N real unknowns. Indeed, especially in the initial value problems higher degree spline functions have not been so successful as in the boundary value problems. Inter-

polation methods requiring more than the first derivative may be said numerically unattractive except for special problems. Now it seems clever for us to direct our considerations to lower degree interpolates.

The purpose of the present paper is to develop an efficient algorithm for the numerical solution of Eq. (1). Let R(t) be the region defined by $R(t) = \{t;\ 0 \leq t \leq T_{max}\}$ and $[0, T_{max}]$ be divided into subintervals. We use a piecewise approximation for each subinterval, over which the equation is integrated to obtain a piecewise equation. Six approximate treatments have been applied, i.e., fully explicit, fully implicit, Crank-Nicolson, linear interpolation, quadratic and cubic spline methods. Using these approximations and performing elementary analytical integrations, we obtain algebraic quadratic equations for unknowns. The numerical solution at each time step is obtained directly as a positive root of this quadratic equation.

The Crank-Nicolson and linear interpolation methods have been applied by Kang and Hansen to a linear equation of the type of Eq. (1) without a nonlinear term[11]. In their numerical examples of the point reactor kinetics with delayed neutrons but without temperature feedback they observed the convergence of $O(h)$ and $O(h^2)$ for the Crank-Nicolson and linear interpolation methods, respectively. Feldstein and Sopka proposed the p-th order perturbed Taylor algorithm for the nonlinear equation in the form of Eq. (3) , but the Volterra kernel is nonlinear in their problem.[12] They showed that if $F \in C^p (p \geq 1)$, $n_i = n(t_i) + O(h^p)$ and the convergence is

uniform on closed intervals. Further, if $F \in C^{p+1}(p \geq 1)$, $n_i = n(t_i) + h^p e(t_i) + O(h^{p+1})$, where $e(t_i)$ is the error function which satisfies a certain functionally linear Volterra integro-differential equation. The algorithm due to Feldstein and Sopka permits general applications, but it is so complicated that it may present severe programming problems, as was mentioned by Feldstein and Sopka themselves.

On the other hand, the algorithm which we propose can be applied only to a special class of equations of the type of Eq. (1), but is very simple and can be programmed easily. It is shown that our algorithm can be applied also to a nonlinear partial differential equation with quadratic nonlinearity, to which belongs a nonlinear fusion reactor kinetics equation.

Nonlinear Volterra integral equations with singuralities in the kernel have been investigated by Garey[13]. His increment method uses the idea of product integration. We don't enter into its details here, because no singuralities are contained in the kernels of the equations of our interest.

2. Numerical Algorithm

Let the region be divided into N subintervals of the width h_i for the i-th interval with mesh points at $t_1 = 0$, $t_2 = h_i$, ---, $t_i = \sum_{j=1}^{i-1} h_j$, ---, $t_{N+1} = T_{max}$. Let n_i denote the approximation for $n(t_i)$ and let

$$n_i(t) = n(t), \text{ when } t \in [t_i, t_{i+1}].$$

Usually, in the discretization of Eq. (1) the differential term $d\, n_i(t)/dt$ is simply replaced by its difference form $(n_{i+1} - n_i)/h_i$. This convential scheme is not used in our algorithm. Let us integrate Eq. (1) over t in the interval $[t_i, t_{i+1}]$. Then we have

$$n_{i+1} - n_i = F\left(\int_{t_i}^{t_{i+1}} dt\, n_i(t), \int_{t_i}^{t_{i+1}} dt \int_0^t ds\, K(t, s, n(s)), \int_{t_i}^{t_{i+1}} dt\, n_i(t) \int_0^t ds\, n(s) \right). \tag{4}$$

In the case of a linear equation, this integrated scheme has been shown by Kang and Hansen to give the algorithm much more stable than the conventional difference scheme[11]. It has been affirmed by us that this is also true in the nonlinear case.

We apply to Eq. (14) the following piecewise approximations.

(1) Fully implicit, fully explicit and Crank-Nicolson methods.

The three methods are given as the special cases of the weighted step function method, which writes

$$n_i(t) = \theta n_i + (1-\theta)n_{i+1}, \quad t \in [t_i, t_{i+1}]. \tag{5}$$

The values of the weight parameter θ, $\theta = 0$, $\frac{1}{2}$ and 1, give the fully implicit, Crank-Nicolson and fully explicit methods, respectively. Hence, they are formulated at once by the use of Eq. (5).

Substituting Eq. (5) in Eq. (4) and performing elementary analytical integration, we get for the first term in the Volterra functional F

$$\int_{t_i}^{t_{i+1}} dt\, n(t) = h_i[\theta n_i + (1-\theta)n_{i+1}]. \tag{6}$$

As for the second term, the formula will be given in the next chapter, because the explicit form of the Volterra kernel depends on problems.

A little care is needed for the first integration over s of the nonlinear term. When $s \notin [t_i, t]$ but $s \in [t_k, t_{k+1}]$,

$$\int_{t_k}^{t_{k+1}} ds\, n(s) = h_k[\theta n_k + (1-\theta)n_{k+1}].$$

When $s \in [t_i, t]$,

$$\int_{t_i}^{t} ds\, n(s) = (t-t_i)[\theta n_i + (1-\theta)n_{i+1}].$$

Hence we have

$$\int_{t_i}^{t_{i+1}} dt\, n_i(t) \int_0^t ds\, n(s) = \int_{t_i}^{t_{i+1}} dt\, n_i(t) \left\{ \sum_{k=1}^{i-1} h_k [\theta n_k + (1-\theta) n_{k+1}] + (t-t_i)[\theta n_i + (1-\theta) n_{i+1}] \right\}$$

$$= \left\{ \sum_{k=1}^{i-1} h_k [\theta n_k + (1-\theta) n_{k+1}] \right\} h_i [\theta n_i + (1-\theta) n_{i+1}] + \frac{1}{2} h_i^2 [\theta n_i + (1-\theta n_{i+1}]^2 . \tag{7}$$

(2) The linear interpolation method

In this case $n_i(t)$ is given by

$$n_i(t) = \frac{1}{h_i} [(t-t_i) n_{i+1} + (t_{i+1}-1) n_i], \quad t \in [t_i, T_{i+1}]. \tag{8}$$

Then, we have

$$\int_{t_i}^{t_{i+1}} dt\, n_i(t) = \frac{1}{2} h_i (n_{i+1} + n_i), \tag{9}$$

$$\int_{t_i}^{t_{i+1}} dt\, n_i(t) \int_0^t ds\, n(s)$$

$$= \frac{1}{24} h_i \left\{ 6 \left[\sum_{k=1}^{i-1} h_k (n_{k+1} + n_k) + h_i n_i \right] + h_i (3 n_{i+1} - n_i) \right\} n_{i+1}$$

$$+ \frac{1}{24} h_i \left\{ 6 \left[\sum_{k=1}^{i-1} h_k (n_{k+1} + n_k) + h_i n_i \right] + h_i (n_{i+1} - 3 n_i) \right\} n_i . \tag{10}$$

(3) The quadratic spline method

We define a quadratic spline function $s_i(t)$ approximation for $n_i(t)$ as follows:

$$n_i(t) \sim s_i(t) = n_i + (t-t_i)\dot{n}_i + \frac{(t-t_i)^2}{h_i^2}(n_{i+1} - n_i - h_i\dot{n}_i), \qquad (11)$$

where $\dot{n}$ is the conventional abbreviation of the time derivative dn/dt. The spline function $s_i(t)$ has, by its defnition, the following intrinsic properties:

$$s_i(t_i) = n_i, \qquad \dot{s}_i(t_i) = \dot{n}_i,$$

$$s_i(t_{i+1}) = n_{i+1}. \qquad (12)$$

As for n_{i+1}, $s_i(t_{i+1})$ gives an approximation to it in a sense of the trapezoidal rule. This can be seen as follows. Diffferentiating Eq. (11), we get

$$\dot{s}_i(t_{i+1}) = \frac{2}{h_i}(n_{i+1}-n_i) - \dot{n}_i.$$

If we put $\dot{s}_i(t_{i+1}) = \dot{n}_{i+1}$ and rearrange the equation given above to obtain

$$n_{i+1} - n_i = \frac{1}{2} h_i(\dot{n}_{i+1}+\dot{n}_i),$$

we see that it is nothing but the formula based on the trapezoidal rule for

$$\int_{t_i}^{t_{i+1}} dt\, \dot{n}_i(t).$$

The value of $\dot{n}_{i+1}$ necessary to proceed to the next step $[t_{i+1}, t_{i+2}]$ can be given by $\dot{s}_i(t_{i+1})$.

Thus we postulate

(i) The spline algorithm I: n_{i+1} is determined from Eq. (4)

making use of Eq. (11) and $\dot{n}_{i+1} = \dot{s}_i(t_{i+1})$.

However, as will be shown in the next chapter, this algorithm is not good for relatively wide subintervals. It will be noticed immediately that $\dot{n}_{i+1}$ can be determined by Eq. (1) itself. That is,

$$\dot{n}_{i+1} = \dot{n}(t_{i+1}) = F\left(n_{i+1}, \int_0^{t_{i+1}} ds\, K(t,s,n(s)), n_{i+1} \int_0^{t_{i+1}} ds\, n(s)\right). \tag{13}$$

Once n_{i+1} is known, $\dot{n}_{i+1}$ is given by Eq. (13).

(ii) The spline algorithm II: n_{i+1} is obtained by the same treatment as in the algorithm I, but $\dot{n}_{i+1}$ is approximated by Eq. (13).

The algorithm II is much more preferable to I. The last term in the Volterra functional of Eq. (13) is now given by

$$n_{i+1} \int_0^{t_{i+1}} ds\, n(s) = \sum_{k=1}^{i} \left[h_k n_k + \frac{h_k^2}{2} \dot{n}_k + \frac{h_k}{3}(n_{k+1} - n_k - h_k \dot{n}_k)\right] n_{i+1}. \tag{14}$$

Substituting Eq. (11) in Eq. (4), we integrate the non-linear term to get

$$\int_{t_i}^{t_{i+1}} dt\, n_i(t) \int_0^t ds\, n(s) = \int_{t_i}^{\tau_{i+1}} dt\, n_i(t) \sum_{k=1}^{i-1} \int_{t_k}^{t_{k+1}} ds\, n_k(s) + \int_{t_i}^{t_{i+1}} dt\, n_i(t) \int_{t_i}^{t} ds\, n(s).$$

The first term on the right hand side of Eq. (14) can be written as

$$\int_{t_i}^{t_{i+1}} dt\, n_i(t) \sum_{k=1}^{i-1} \int_{t_k}^{t_{k+1}} ds\, n_k(s) = U_i H_i , \tag{15}$$

where

$$U_i = \int_{t_i}^{t_{i+1}} dt\, n_i(t) = h_i n_i + \frac{h_i^2}{2} \dot{n}_i + \frac{h_i}{3} (n_{i+1} - n_i - h_i \dot{n}_i),$$

$$H_i = H_{i-1} + U_{i-1} \quad , \quad H_1 = 0.$$

The second term is rather complicated.

$$\int_{t_i}^{t_{i+1}} dt\, n_i(t) \int_{t_i}^{t} ds\, n_i(s)$$

$$= \int_{t_i}^{t_{i+1}} dt \left[n_i + \dot{n}_i (t-t_i) + \frac{(t-t_i)^2}{h_i^2} (n_{i+1} - n_i - h_i \dot{n}_i) \right]$$

$$\times \left[n_i (t-t_i) + \frac{\dot{n}_i}{2} (t-t_i)^2 + \frac{(t-t_i)^3}{3 h_i^2} (n_{i+1} - n_i - h_i \dot{n}_i) \right]$$

$$= \frac{2}{9} h_i^2 (n_i)^2 + \frac{1}{9} h_i^3 n_i \dot{n}_i + \frac{1}{72} h_i^4 (\dot{n}_i)^2$$

$$+ \frac{2}{9} h_i^2 \left(n_i + \frac{1}{4} h_i \dot{n}_i\right) n_{i+1} + \frac{1}{18} h_i^2 (n_{i+1})^2 . \tag{16}$$

Finally we try a cubic spline method of Guzek and Kemper[8].

(4) The cubic spline method

The cubic spline function defined by Guzek and Kemper is given by[8]

$$S_i(t) = n_i + (t-t_i) \dot{n}_i + \frac{(t-t_i)^2}{2} \ddot{n}_i + \frac{(t-t_i)^3}{2 h_i^2} (\dot{n}_{i+1} - \dot{n}_i - h_i \ddot{n}_i). \tag{17}$$

Intrinsic relations for $s_i(t)$ are

$s_i(t_i) = n_i$, $\dot{s}_i(t_i) = \dot{n}_i$, $\ddot{s}_i(t_i) = \ddot{n}_i$ and $\dot{s}_i(t_{i+1}) = \dot{n}_{i+1}$.

Note that n_{i+1} does not appear in the defining expression of $s_i(t)$. What is determined directly is not n_{i+1} but $\dot{n}_{i+1}$. This fact may suggest that the original differential form Eq. (1) is more suitable than the integrated Eq. (4) for obtaining the first derivative itself. Guzek and Kemper were successful with this differential scheme. We also tried it in our problem but were unhappy in getting the divergent results. Thus, we turned back to the integrated equation.

Once $\dot{n}_{i+1}$ is obtained, it determines $s_i(t)$ completely, from which follow the values $n_{i+1} = s_i(t_{i+1})$ and $\ddot{n}_{i+1} = \ddot{s}_i(t_{i+1})$ at the next time step.

Now consider Eq. (4) again. Elementary integrations give

$$\int_{t_i}^{t_{i+1}} dt\, n_i(t) = h_i n_i + \frac{h_i^2}{2}\dot{n}_i + \frac{h_i^3}{6}\ddot{n}_i + \frac{h_i^2}{12}(\dot{n}_{i+1} - \dot{n}_i - h_i\ddot{n}_i)$$
$$\equiv \Pi_i(\dot{n}_{i+1}) \tag{18}$$

and

$$\int_{t_i}^{t_{i+1}} dt\, n_i(t) \sum_{k=1}^{i-1} \int_{t_k}^{t_{k+1}} ds\, n_k(s) = \sum_{k=1}^{i-1} \Pi_k(\dot{n}_{k+1})\Pi_i(\dot{n}_{i+1}), \tag{19}$$

$$\int_{t_i}^{t_{i+1}} dt\, n_i(t) \int_{t_i}^{t} ds\, n(s)$$

$$= n_i\left[\frac{h_i^2}{2} n_i + \frac{h_i^3}{6}\dot{n}_i + \frac{h_i^4}{24}\ddot{n}_i + \frac{h_i^3}{60}(\dot{n}_{i+1} - \dot{n}_i - h_i\ddot{n}_i)\right]$$

$$+ \dot{n}_i\left[\frac{h_i^3}{3} n_i + \frac{h_i^4}{8}\dot{n}_i + \frac{h_i^5}{30}\ddot{n}_i + \frac{h_i^4}{72}(\dot{n}_{i+1} - \dot{n}_i - h_i\ddot{n}_i)\right]$$

$$+ \ddot{n}_i\left[\frac{h_i^4}{8} n_i + \frac{h_i^5}{20}\dot{n}_i + \frac{h_i^6}{72}\ddot{n}_i + \frac{h_i^5}{168}(\dot{n}_{i+1} - \dot{n}_i - h_i\ddot{n}_i)\right]$$

$$+ (\dot{n}_{i+1} - \dot{n}_i - h_i\ddot{n}_i)\left[\frac{h_i^3}{15} n_i + \frac{h_i^4}{36}\dot{n}_i + \frac{h_i^5}{126}\ddot{n}_i + \frac{h_i^4}{288}(\dot{n}_{i+1} - \dot{n}_i - h_i\ddot{n}_i)\right]. \tag{20}$$

All the results derived above are rearranged finally to give the algebraic quadratic equations:

$$A_i (n_{i+1})^2 + B_i\, n_{i+1} + C_i = 0 \tag{21}$$

for the algorithms (1), (2) and (3). For the algorithms (4), we have

$$A_i (\dot{n}_{i+1})^2 + B_i\, \dot{n}_{i+1} + C_i = 0. \tag{22}$$

Explicit expressions of A_i, B_i and C_i will be given for the point reactor kinetics equations in the next chapter.

The numerical solution at each time step is given by a positive real root of Eq. (21);

$$n_{i+1} = \frac{-B_i + \sqrt{B_i^2 - 4A_iC_i}}{2A_i} \tag{23}$$

because physics requires positive solutions. Thus, the existence and uniqueness of the solution are assured only when

$$\frac{-B_i + \sqrt{B_i^2 - 4A_iC_i}}{2A_i} \geq 0, \tag{24}$$

$$\frac{-B_i - \sqrt{B_i^2 - 4A_iC_i}}{2A_i} < 0. \tag{25}$$

For Eq. (22) negative solutions for $\dot{n}_{i+1}$ are permitted but n_{i+1} must be positive. The stable algorithms are those which always satisfy these conditions and give convergent results. The stability and error bounds of the algorithms (1) and (2) have been discussed already for the nonlinear point reactor kinetics equation in the prompt approximation.[1] General discussions on them are not transparent when the Volterra kernel is taken into consideration, because it brings great complexities in A_i, B_i and C_i for some problems.

3. Application to a Nonlinear Point Reactor Kinetics Equation

Temporal behavior of the nuclear fission reactor can be approximated by the point reactor model, which is formulated by the following nonlinear system of differential equations[3]:

$$\left.\begin{aligned} \frac{d}{dt}n(t) &= \frac{\rho(t,T)-\beta}{\ell}n(t) + \sum_{j=1}^{J}\lambda_j R_j(t), \\ \frac{d}{dt}R_j(t) &= \frac{\beta_j}{\ell}n(t) - \lambda_j R_j(t), \\ \rho(t,T) &= r(t) - f(T), \\ \frac{d}{dt}T(t) &= \frac{1}{C}n(t). \end{aligned}\right\} \tag{26}$$

Initial conditions are defined consistently with

$$\frac{d}{dt}n = \frac{d}{dt}R_j = 0 \quad \text{for} \quad t = 0-\varepsilon, \tag{27}$$

where ε is a infinitesimal parameter. All the functions involved are piecewise continuous functions of time t.

Symbols are defined as follows:

$n(t)$ = reactor power density (w/cm^3),

$\rho(t,T)$ = reactivity,

l = prompt neutron lifetime (sec),

λ_j = decay constant of the j-th delayed neutron precursors (sec^{-1}),

$R_j(t)$ = concentration of delayed neutron precursors of the j-th group (cm^{-3}),

β_j = fraction of delayed neutrons for the j-th group,

$\beta = \sum_{j=1}^{J} \beta_j$, J being the number of delayed neutron groups,

$r(t)$ = reactivity insertion,

$f(T)$ = temperature feedback,

$T(t)$ = temperature (°C),

C = heat capacity ($W.sec/cm^3.°C$),

We assume the ramp reactivity insertion

$$r(t) = \alpha t \tag{28}$$

and the linear temperature feedback

$$f(T) = \gamma(T-T_o), \tag{29}$$

where

α = input reactivity ramp rate (sec^{-1}),

γ = temperature coefficient (>0 for the negative feedback by the definition of f(T)) ($°C^{-1}$).

Conventionally, Eq. (26) has been solved iteratively by using, for example, the predictor-corrector method. Fortunately, Eq. (26) can be unified to a single equation, whose form is what we have considered in the preceeding chapters. Using Eqs. (28) and (29), we get

$$\frac{d}{dt}n(t) = \frac{\alpha t-\beta}{\ell}n(t) + \sum_{j=1}^{J}\lambda_j\left[R_{j0}e^{-\lambda_j t} + \frac{\beta_j}{\ell}\int_0^t ds\, n(s)e^{-\lambda_j(t-s)}\right] - \frac{\gamma}{C\ell}n(t)\int_0^t ds\, n(s). \tag{30}$$

Thus the integral of the Volterra kernel K can be written now in an explicit form

$$\int_0^t ds\, K(t,s,n(s)) = \frac{\alpha}{\ell} t\, n(t) + \sum_{j=1}^{J} \lambda_j \left[R_{j0} e^{-\lambda_j t} + \frac{\beta_j}{\ell} \int_0^t ds\, n(s) e^{-\lambda_j (t-s)} \right] \tag{31}$$

with the initial relation $R_{j0} = \beta_j/(\lambda_i \ell)$.

When the ramp reactivity insertion is terminated at a time $T_s (< T_{max})$, the reactivity term has to be replaced by

$$\frac{\alpha}{\ell} [t]_{T_s} n(t),$$

where

$$[t]_{Ts} = \begin{cases} t & \text{for } t \leq Ts, \\ 0 & \text{for } t > Ts. \end{cases}$$

This abrupt termination of the reactivity insertion may violate the continuity assumed, but it does not put any obstacles in applying our algorithm to the present problem.

Results of the elementary integration of the integral

$$\int_{t_i}^{t_{i+1}} dt \int_0^t ds\, K(t,s,n(s))$$

are listed in Appendix I for the algorithms (1) ~ (4) given in Chapter 2. Here we show only the final equations to be solved.

(1) Fully implicit, fully explicit and Crack-Nicolson methods

$$A_i (n_{i+1})^2 + B_i\, n_{i+1} + C_i = 0 \tag{32}$$

with

$$A_i = \frac{\gamma}{C\ell}\frac{h_i^2}{2}(1-\theta)^2,$$

$$B_i = \frac{\gamma}{C\ell}h_i^2\theta(1-\theta)n_i + b_{i+1}, \tag{33}$$

$$C_i = -b_i n_i + \frac{\gamma}{C\ell}\frac{h_i^2}{2}\theta^2 n_i^2 - \sum_{j=1}^{J}(e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}})S_{ij},$$

where

$$\left.\begin{aligned} b_{i+1} &= 1 - (1-\theta)a_i, \\ b_i &= 1 + \theta a_i, \\ a_i &= \frac{(\alpha t_i - \beta)}{\ell}h_i + \frac{\alpha}{2\ell}h_i^2 - \frac{\gamma}{C\ell}h_i H_i \\ &\quad + \frac{1}{\ell}\sum_{j=1}^{J}\beta_j\left[h_i - \frac{1}{\lambda_j}(1-e^{-\lambda_j h_i})\right], \end{aligned}\right\} \tag{34}$$

$$H_i = \sum_{k=1}^{i-1}[\theta n_k + (1-\theta)n_{k+1}]n_k, \tag{35}$$

$$S_{ij} = R_{jo} + \frac{\beta_j}{\ell\lambda_j}\sum_{k=1}^{i-1}[\theta n_k + (1-\theta)n_{k+1}](e^{\lambda_j t_{k+1}} - e^{\lambda_j t_k}). \tag{36}$$

For the fully explicit method it is seen that $A_i = 0$. Hence, in this case the equation becomes linear in n_{i+1};

$$n_{i+1} = \left(1 + a_i - \frac{\gamma}{C\ell}\frac{h_i^2}{2}n_i\right)n_i + \sum_{j=1}^{J}(e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}})S_{ij}. \tag{37}$$

It is easily seen that $S_{ij} > 0$ if n_k and n_{k+1} are positive, since all physical parameters R_{jo}, β_j, 1 and λ_j are defined to be positive and $t_{k+1} > t_k$. Hence the sufficient and

necessary condition for the positiveness of the solution of Eq. (37) are given by

$$1 + a_i - \frac{\gamma}{C\ell}\frac{h_i^2}{2} n_i > 0. \tag{38}$$

When $\theta \neq 1$, the discriminant for the quadratic equation is written as follows;

$$B_i^2 - 4A_iC_i = b_{i+1}^2 + 2\frac{\gamma}{C\ell} h_i^2 (1-\theta) n_i + 2\frac{\gamma}{C\ell} h_i^2 (1-\theta)^2 \sum_{j=1}^{J} (e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}}) S_{ij}. \tag{39}$$

On the right hand side of Eq. (39), since $0 \leq \theta < 1$, the second term is positive when $n_i > 0$ and the third term is also positive. Thus it follows immediately that for $\theta = 0$

$$-B_i + \sqrt{B_i^2 - 4A_iC_i} > 0. \tag{40}$$

This gives the proof to the theorem which states that <u>the fully implicit method is unconditionally stable</u>.

For the Crank-Nicolson method it is not self-evident that Eq. (40) holds always. However, from our computational experiences we know that it does hold almost always for our problems considered here.

(2) The linear interpolation method

$$A_i = \frac{\gamma}{C\ell}\frac{h_i^2}{8},$$

$$B_i = \frac{\gamma}{C\ell}\frac{h_i^2}{4} n_i + b_{i+1}, \tag{41}$$

$$C_i = -b_i n_i + \frac{\gamma}{C\ell}\frac{h_i^2}{8} n_i^2 - \sum_{j=1}^{J} (e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}}) S_{ij},$$

where

$$b_{i+1} = 1 - a_{i+1}, \qquad b_i = 1 + a_i,$$

$$a_{i+1} = \left[\frac{\alpha(2t_{i+1}+t_i)}{3} - \beta\right]\frac{h_i}{2\ell} - \frac{\gamma}{C\ell}\frac{h_i}{4}H_i + \frac{1}{\ell}\sum_{j=1}^{J}\frac{\beta_j}{h_i}\alpha_{i+1,j}, \tag{42}$$

$$a_i = \left[\frac{\alpha(t_{i+1}+2t_i)}{3} - \beta\right]\frac{h_i}{2\ell} - \frac{\gamma}{C\ell}\frac{h_i}{4}H_i + \frac{1}{\ell}\sum_{j=1}^{J}\frac{\beta_j}{h_i}\alpha_{i,j},$$

with

$$H_i = \sum_{k=1}^{i-1} h_k \left(n_k + n_{k+1}\right), \tag{43}$$

$$\left.\begin{aligned}
\alpha_{i+1,j} &= \frac{1}{\lambda_j^2}\left(1 - e^{-\lambda_j h_i}\right) + h_i\left(\frac{h_i}{2} - \frac{1}{\lambda_j}\right), \\
\alpha_{i,j} &= \left(h_i + \frac{1}{\lambda_j}\right)\frac{1}{\lambda_j}\left(1 - e^{-\lambda_j h_i}\right) - \left(\frac{h_i}{2} + \frac{1}{\lambda_j}\right),
\end{aligned}\right\} \tag{44}$$

$$S_{ij} = R_{j0} + \frac{\beta_j}{\lambda_j}\sum_{k=1}^{i-1}\frac{1}{h_k}\left\{\left[e^{\lambda_j t_{k+1}}\left(h_k - \frac{1}{\lambda_j}\right) + \frac{1}{\lambda_j}e^{\lambda_j t_k}\right]n_{k+1} - \left[e^{\lambda_j t_k}\left(h_k + \frac{1}{\lambda_j}\right) - \frac{1}{\lambda_j}e^{\lambda_j t_{k+1}}\right]n_k\right\}. \tag{45}$$

For the coefficients given by Eq. (41) the discriminat is so complicated that it is not easy to see whether the positive definiteness of the solution can be maintained or not. In this case also computational experiences tell us that the linear interpolation method is as good as the Crank-Nicolson method.

(3) The quadratic spline method

$$A_i = \frac{\gamma}{C\ell}\frac{h_i^2}{18},$$

$$B_i = \frac{\gamma}{C\ell}\frac{2}{9}h_i^2\left(n_i + \frac{h_i}{4}\dot{n}_i\right) + b_{i+1}, \tag{46}$$

$$C_i = -b_i n_i - c_i \dot{n}_i + \frac{\gamma}{C\ell}\frac{h_i^2}{9}\left[2(n_i)^2 + \frac{h_i^2}{8}(\dot{n}_i)^2\right] - \sum_{j=1}^{J}(e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}})\,\delta_{i,j},$$

where

$$b_{i+1} = 1 - a_{i+1},$$

$$b_i = 1 + a_i,$$

$$a_{i+1} = \left[\alpha\left(t_{i+1} - \frac{h_i}{4}\right) - \beta\right]\frac{h_i}{3\ell} - \frac{\gamma}{C\ell}\frac{h_i}{3}H_i + \frac{1}{\ell}\sum_{j=1}^{J}\beta_j\alpha_{i+1,j},$$

$$a_i = \left[\frac{\alpha(3t_{i+1} + 5t_i)}{4} - 2\beta\right]\frac{h_i}{3\ell} - \frac{\gamma}{C\ell}\frac{2}{3}h_i H_i + \frac{1}{\ell}\sum_{j=1}^{J}\beta_j\alpha_{i,j} + \frac{\gamma}{C\ell}\frac{h_i^3}{9}\dot{n}_i, \qquad (47)$$

$$c_i = \left[\alpha\left(t_{i+1} - \frac{h_i}{2}\right) + \beta\right]\frac{n_i^2}{6\ell} - \frac{\gamma}{C\ell}\frac{h_i^2}{6}H_i + \frac{1}{\ell}\sum_{j=1}^{J}\beta_j\left[\frac{h_i^2}{2} - \frac{h_i}{\lambda_j} + \frac{1}{\lambda_j^2}(1 - e^{-\lambda_j h_i}) - h_i\alpha_{i,j}\right],$$

with

H_i being defined in Eq. (15),

$$S_{i,j} = R_{j0} + \frac{\beta_j}{\lambda_j\ell}\sum_{k=1}^{i-1}\left\{(e^{\lambda_j t_{k+1}} - e^{\lambda_j t_k})n_k + \left[e^{\lambda_j t_{k+1}}\left(h_k - \frac{1}{\lambda_j}\right) + \frac{1}{\lambda_j}e^{\lambda_j t_k}\right]\dot{n}_k + \left[e^{\lambda_j t_{k+1}}\left(h_k^2 - \frac{2h_k}{\lambda_j} + \frac{2}{\lambda_j^2}\right) - \frac{2}{\lambda_j^2}e^{\lambda_j t_k}\right]\frac{(n_{k+1} - n_k - h_k\dot{n}_k)}{h_k^2}\right\},$$

$$\alpha_{i+1,j} = \frac{1}{n_i^2}\left[\frac{h_i^3}{3} - \frac{h_i^2}{\lambda_j} + \frac{2h_i}{\lambda_j^2} - \frac{1}{\lambda_j^3}(1 - e^{-\lambda_j h_i})\right],$$

$$\alpha_{i,j} = h_i - \frac{1}{\lambda_j}(1 - e^{-\lambda_j h_i}) - \alpha_{i+1,j}.$$

In the algorithm II the equation to determine the value of $\dot{n}_{i+1}$ from n_{i+1} now can be derived from Eq. (30) to give

$$\dot{n}_{i+1} = \frac{\alpha t_{i+1} - \beta}{\ell} n_{i+1} + \sum_{j=1}^{J} \lambda_j \left[R_{j0} e^{-\lambda_j t_{i+1}} + \frac{\beta_j}{\ell} \int_0^{t_{i+1}} ds\, n(s) e^{-\lambda_j (t-s)} \right]$$

$$- \frac{\gamma}{C\ell} n_{i+1} \int_0^{t_{i+1}} ds\, n(s), \tag{48}$$

where the integration of the last term on the right hand side of Eq. (48) has been given already by Eq. (14), which can be rewritten simply as $n_{i+1} H_{i+1}$. For the second term in the bracket on the right hand side of Eq. (48), we have

$$\int_0^{t_{i+1}} ds\, n(s) e^{-\lambda_j (t_{i+1} - s)} = \frac{1}{\lambda_j} \sum_{k=1}^{i-1} \Bigg\{ \left(e^{\lambda_j t_{k+1}} - e^{\lambda_j t_k} \right) n_k$$

$$+ \left[e^{\lambda_j t_{k+1}} \left(h_k - \frac{1}{\lambda_j} \right) + \frac{1}{\lambda_j} e^{\lambda_j t_k} \right] \dot{n}_k + \frac{1}{h_k^2} \left[e^{\lambda_j t_{k+1}} \left(h_k^2 - \frac{2h_k}{\lambda_j} + \frac{2}{\lambda_j^2} \right) \right.$$

$$\left. - \frac{2}{\lambda_j^2} e^{\lambda_j t_k} \right] \left(n_{k+1} - n_k - h_k \dot{n}_k \right) \Bigg\} + \frac{1}{\lambda_j} \left[\left(1 - e^{-\lambda_j h_i} \right) + \left(h_i - \frac{1}{\lambda_j} + \frac{1}{\lambda_j} e^{-\lambda_j h_i} \right) \dot{n}_i \right.$$

$$\left. + \frac{1}{h_i^2} \left(h_i^2 - \frac{h_i}{\lambda_j} + \frac{2}{\lambda_j^2} - \frac{2}{\lambda_j^2} e^{-\lambda_j h_i} \right) \left(n_{i+1} - n_i - h_i \dot{n}_i \right) \right].$$

As can be seen immediately, the right hand side of Eq. (48) does not contain $\dot{n}_{i+1}$. Thus $\dot{n}_{i+1}$ can be calculated easily, once n_{i+1} is determined.

It is shown in the next chapter that the solutions obtained by the algorithm II are stable for problems we are considering. The algorithm I, however, gives oscillatory or sometimes divergent solutions, depending on the values of the prompt neutron lifetime, when the coarse mesh widths are used.

(4) The cubic spline method

We have been unsuccessful in obtaining reasonable results with this method. Discriminants become negetive too many times to continue meaningful calculations on a computer. Therefore, we don't reproduce here the lengthy expressions for A_i, B_i and C_i, of which interested readers may refer to Appendix 2.

4. Benchmark and Model Problems

To demonstrate the practical effectiveness of our algorithms, the Argonne benchmark problem ID.7-A1 has been adopted as an example of a model thermal reactor. This problem has been also used to simulate the temporal behavior of a model fast reactor by modifying only the value of the prompt neutron lifetime to be consistent with a fast reactor.

(1) Benchmark problem for a thermal reactor

The point reactor kinetics equation in the prompt approximation with ramp reactivity insertion and linear temperature feedback takes a simple form

$$\frac{d}{dt} n(t) = \frac{\alpha}{\ell} t\, n(t) - \frac{\gamma}{C\ell} n(t) \int_0^t ds\, n(s), \tag{49}$$

which can be solved by using the algorithms formulated in Chapter 3 with all parameters related to the delayed neutrons being put to zero.

Froehlich and Johnson have succeeded in transforming Eq. (49) to a simple integration problem.[2] Defining the dimensionless parameters: $a = \frac{\alpha}{\ell}$, $b = \frac{\gamma}{C1}$, $V(t) = \frac{b}{a} n(t)$, $x = \sqrt{a}\, t$ and introducing the definition

$$W = \ln V(x),$$

the function x(w) is calculated from the integral:

$$x(w) = \int_{w(0)}^{w} du \, \frac{1}{\sqrt{2(u - e^{u} - w(0) + e^{w(0)})}}, \tag{50}$$

which is integrated by Simpson's rule.

Relations between x and V have been tabulated by Meneley for the following input data[3]:

Initial power density $n(0) = 10\ w/cm^3$,

Initial temperature $T(0) = 0\ ^\circ C$,

Prompt neutron lifetime $l = 5.10^{-4} sec$,

Input reactivity ramp rate $= 5.10^{-2} sec^{-1}$,

Temperature coefficient $= 1 \times 10^{-4} (^\circ C)^{-1}$

(taken positive for negative feedback),

Heat capacity $C = 2\ w/sec/cm^2.\ ^\circ C$.

Numerical solutions due to Meneley are taken as standard values, with which approximate solutions from our algorithms are to be compared.

For simplicity, time mesh widths are taken to be uniform, i.e., $h_i = \Delta t$(constant). Table 1 shows the approximate solutions, at several time steps after the start of reactivity insertion, obtained by the present algorithms using lower degree interpolates for relatively coarse mesh widths. It is seen from the table that the fully explicit method ($\theta = 1$) with $\Delta t = 0.05$ sec gives unphysical negative solutions. In Fig. 1 the quadratic spline algorithms are compared with the Crank-Nicolson and linear interpolation methods for $\Delta t = 0.05$ sec, which is 100 times of l. The Crank-Nicolson and linear interpolation methods show almost converged results, as can be noticed by referring to Fig. 2, which demonstrates the converging behavior of the solutions by the linear interpolation method. The Crank-Nicolson method produces almost the same as or somewhat better convergence than the linear inter-

polation method. Unphysical oscillations occur also in the quadratic spline algorithm I, as seen from Fig. 1. Fig. 3 indicates that oscillations disappear for $\Delta t \leq 10$ msec. Fig. 4 shows the convergency of the solutions obtained by the quadratic spline algorithm II. Convergencies of the solutions at $t = 0.5$ sec by the implicit, Crank-Nicolson, linear interpolation and quadratic spline (II) methods are compared in Fig. 5, from which it is seen that the solutions by the former three methods converge from the lower side, while the last one does from the upper side.

In Table 2 comparisons are made of $V(x)$ at $x = 3.1$ ($t = 0.31$ sec) between approximate values. The Crank-Nicolson method may be said to be the best in accuracy. The linear interpolation method is as accurate as it. Although the fully implicit method has been proved unconditionally stable, the large errors are remarkable for it in comparison with the other methods. Eq. (50) makes it difficult to tabulate $V(x)$ versus x of a regular step. Values for x in the Meneley's table are so irregularly scattered that $x = 3.1$ is the only value which is common to all Δt's found in the Meneley's table. This is the reason why we picked up this particulat value of x.

Ranking of the accuracy is illustrated by the number of asterisks in Table 2.

Numerical experiments presented above can be summarized in the ranking of the stability as shown in Table 3, where the results of estimations due to Kang and Hansen[11] for a linear problem without temperature feedback but with delayed neutrons are compared with those for our nonlinear problem. It is

interesting to see that the stabilities of fully explicit and implicit methods are reversed for the linear and nonlinear problems. The proof of the unconditional stability of the fully implicit method for the nonlinear problem has been given in Chapter 3. We show here why the fully implicit method is not unconditionally stable for the linear problem.

For simplicity, we neglect the delayed neutrons. This simplification does not reduce the generality of our arguments. Consider the equation

$$\frac{d}{dt} n(t) = (a + bt)\, n(t), \quad a > 0, \quad b > 0. \tag{51}$$

The integration of Eq. (51) over the interval $[t_i, t_{i+1}]$ by making use of the weighted step function gives

$$n_{i+1} - n_i = a h_i [\theta n_i + (1-\theta) n_{i+1}]$$
$$+ b h_i \left(t_i + \frac{h_i}{2}\right) [\theta n_i + (1-\theta) n_{i+1}],$$

which is rewritten as follows,

$$b_{i+1} n_{i+1} = b_i n_i, \tag{52}$$

where

$$b_{i+1} = 1 - \left[a + b\left(t_i + \frac{h_i}{2}\right)\right] h_i (1-\theta),$$

$$b_i = 1 + \left[a + b\left(t_i + \frac{h_i}{2}\right)\right] h_i \theta.$$

When $\theta = 0$, Eq. (52) becomes

$$b_{i+1} n_{i+1} = n_i .$$

Thus the condition

$$b_{i+1} = 1 - [a + b(t_i + \frac{h_i}{2})]h_i > 0$$

must be satisfied in order to get a positive solution.

When $\theta = 1$, we have

$$n_{i+1} = b_i n_i. \tag{53}$$

Since b_i is positive definite, the fully explicit method is unconditionally stable for the linear problem.

The remarkable contrast is also seen for the interpolates of higher degree. In the linear problem, higher the degree of the interpolate, more stable the algorithm. Contrary is true in the nonlinear problem. As has been mentioned already in Chapter 3, in the nonlinear problem minor errors of higher derivatives contained in the higher degree interpolate may bring about the undesirable amplification of errors in solutions resulting in catastrophic divergence.

(2) Model problem for a fast reactor

In order to see the effects of the values of the prompt neutron lifetime on the relation of the convergence and stability to the time mesh width, we consider the following simple problem:

Prompt neutron lifetime $\ell = 5.10^{-7}$ sec,

Values of all other parameters are the same as in the benchmark problem mentioned above.

Fig. 6 summarizes the computational results for lower degree interpolates. At the power increasing stage the solutions are not so bad as one might imagine for a very coarse mesh of 10^4 times of ℓ, but at the decreasing stage these three

methods are seen to violate the stability and convergency. For Δt = 1 msec = 1,000 ℓ, the Crank-Nicolson and linear interpolation methods give almost converged solutions. The fully implicit method is not good enough to get converged solutions with this time step.

The quadratic spline algorithms I and II are still worse for this mesh width of 1 msec , as shown in Fig. 7.

Finally we make a comparison of CPU (Central Processing Unit) time for the seemingly favourable four methods in Table 4. Again the Crank-Nicolson method takes the first rank. It is natural that higher the degree of interpolate, more computational time required per time step.

From all evidences given above, it can be concluded that the Crank-Nicolson method is the best in our direct method and the linear interpolation is a good competitor to it.

In the problems condidered above the delayed neutrons have been neglected. Inclusion of them improves the stability of our algorithm. For example, refer to Eq. (39). The last term on the right hand side of it is due to the delayed neutrons, which gives a positive contribution to the discriminant. Thus the positivity of solutions is enhanced by including the delayed neutrons.

Table 1. Comparison of approximate solutions by lower degree interpolation methods

Methods	Time step width h(sec.)	Time (sec.)				
		0.2	0.4	0.5	0.6	0.8
θ=1.0	0.05	4.6430×10	1.5643×10^3	9.2384×10^3	-7.3797×10^3	-1.5140×10^2
	0.01	6.3820×10	5.5166×10^3	1.8673×10^3	6.4674×10	2.2044
	0.005	6.7457×10	6.0901×10^3	1.5232×10^3	8.6213×10	5.1289
	0.001	7.0776×10	6.3046×10^3	1.3363×10^3	1.0313×10^2	8.9716
θ=0.5	0.05	7.7705×10	5.8231×10^3	1.0322×10^3	7.5292×10	1.0001×10
	0.01	7.1891×10	6.3039×10^3	1.2847×10^3	1.0571×10^2	1.0177×10
	0.005	7.1727×10	6.3119×10^3	1.2946×10^3	1.0691×10^2	1.0190×10
	0.001	7.1764×10	6.3143×10^3	1.2978×10^3	1.0730×10^2	1.0195×10
θ=0.0	0.05	2.6355×10^2	2.1246×10^3	8.0616×10^2	3.7005×10^2	4.3078×10^2
	0.01	8.2471×10	5.5581×10^3	1.0329×10^3	1.4943×10^2	3.0470×10
	0.005	7.6610×10	6.1009×10^3	1.1430×10^3	1.2826×10^2	1.0274×10
	0.001	7.2596×10	6.3060×10^3	1.2624×10^3	1.1150×10^2	1.1538×10
Linear interpola-tion	0.05	8.1449×10	5.7807×10^3	8.8193×10^2	6.5583×10	1.0053×10
	0.01	7.2009×10	6.3092×10^3	1.2775×10^3	1.0515×10^2	1.0171×10
	0.005	7.1756×10	6.3133×10^3	1.2928×10^3	1.0677×10^2	1.0189×10
	0.001	7.1675×10	6.3144×10^3	1.2977×10^3	1.0729×10^2	1.0195×10

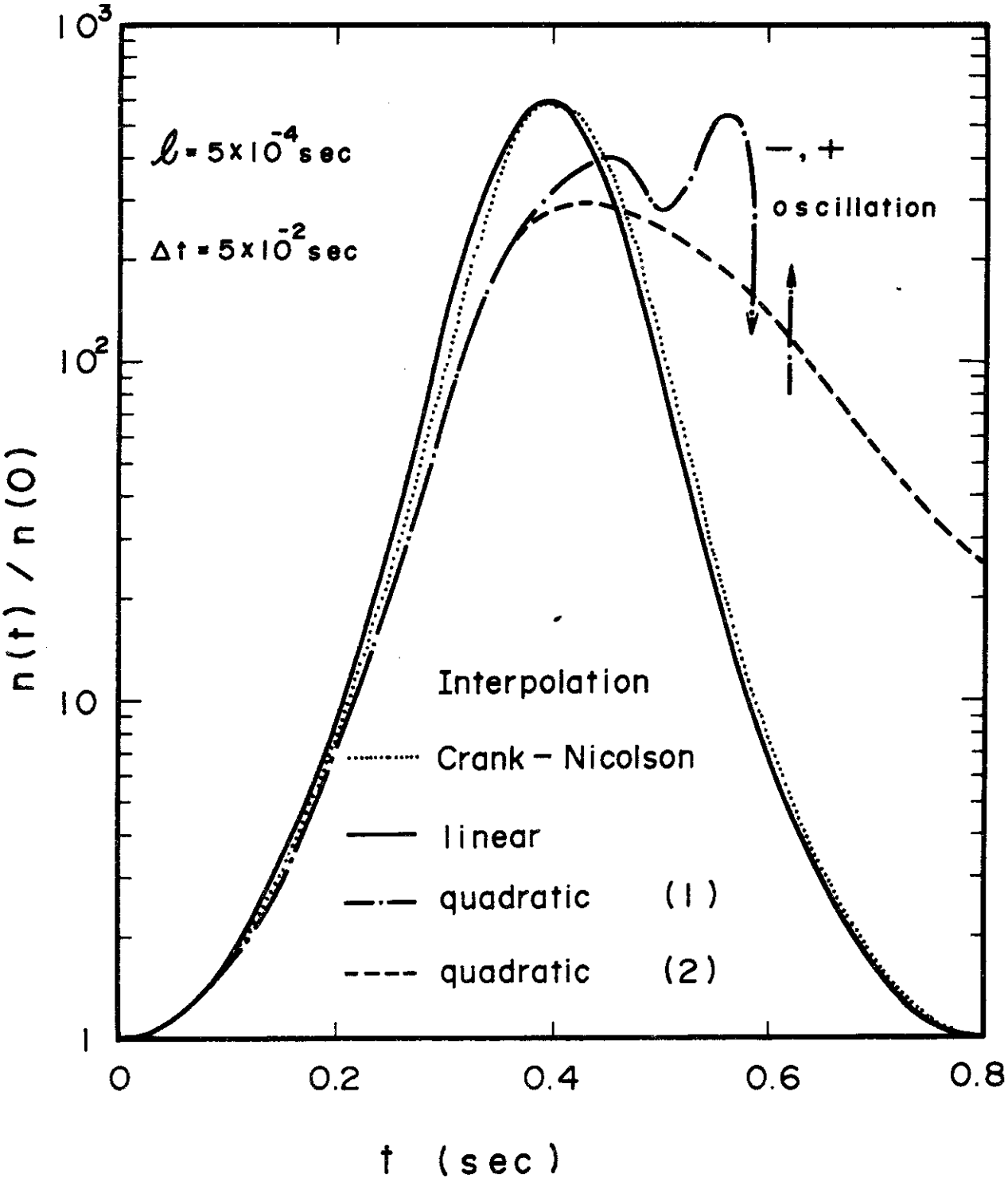

Fig. 1 Comparison of approximate solutions of the benchmark problem for coarse mesh width of 100 times of ℓ

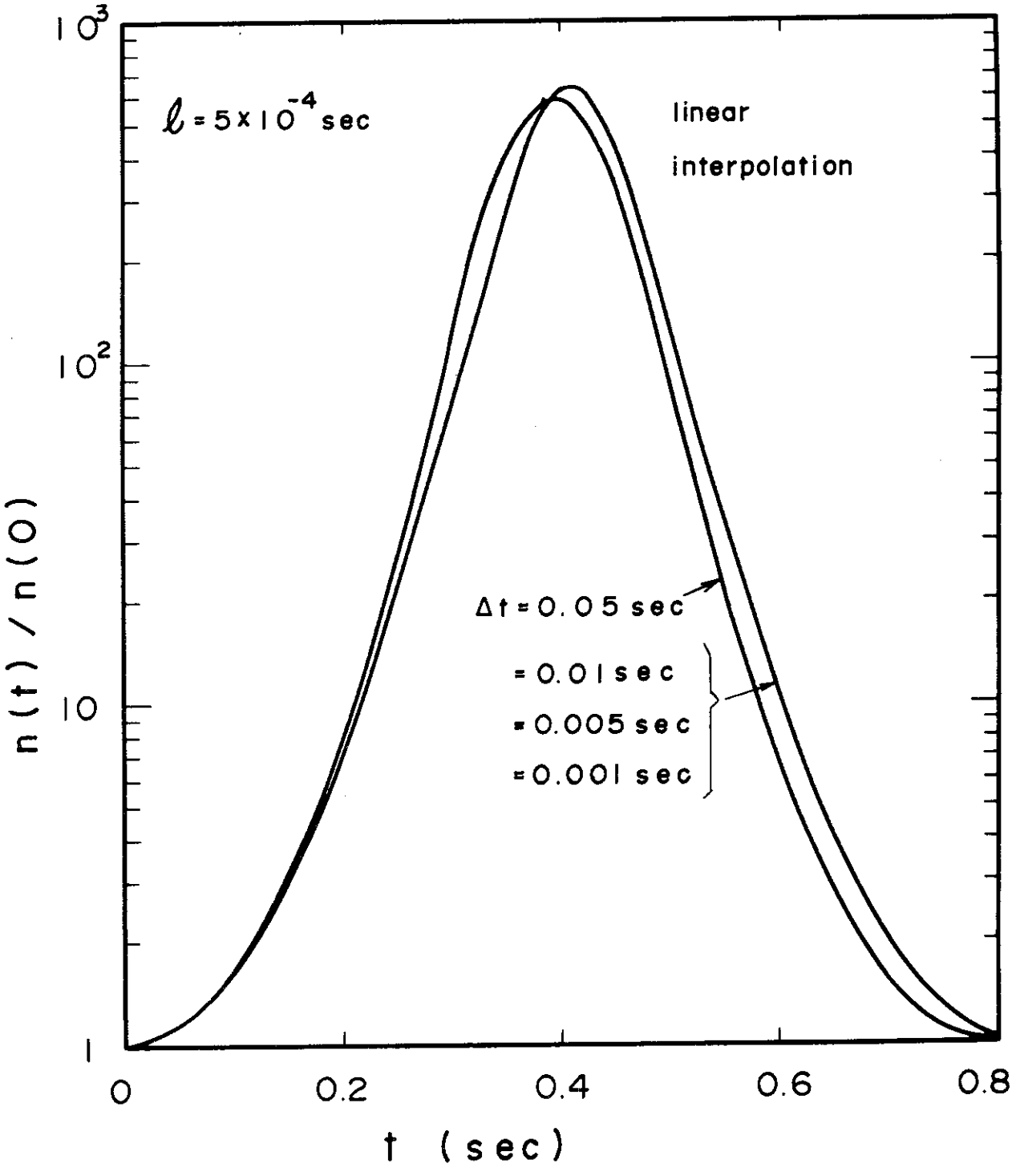

Fig. 2 Convergence behaviour of the approximate solutions by the linear interpolation method

The solutions by the Crank-Nicolson method show almost the same behaviour.

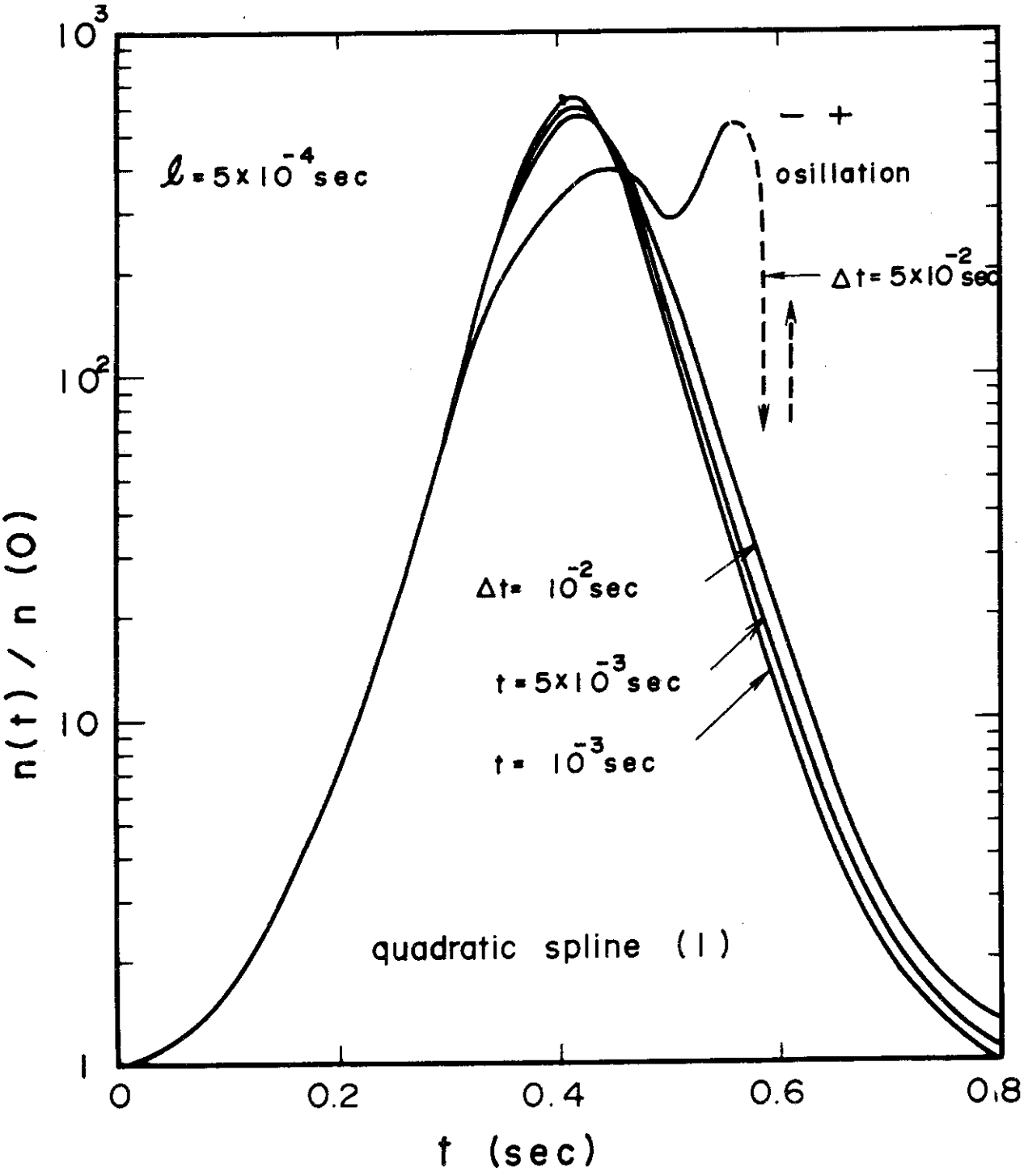

Fig. 3 Convergence behaviour of the approximate solutions by the quadratic spline method with the algorithm I

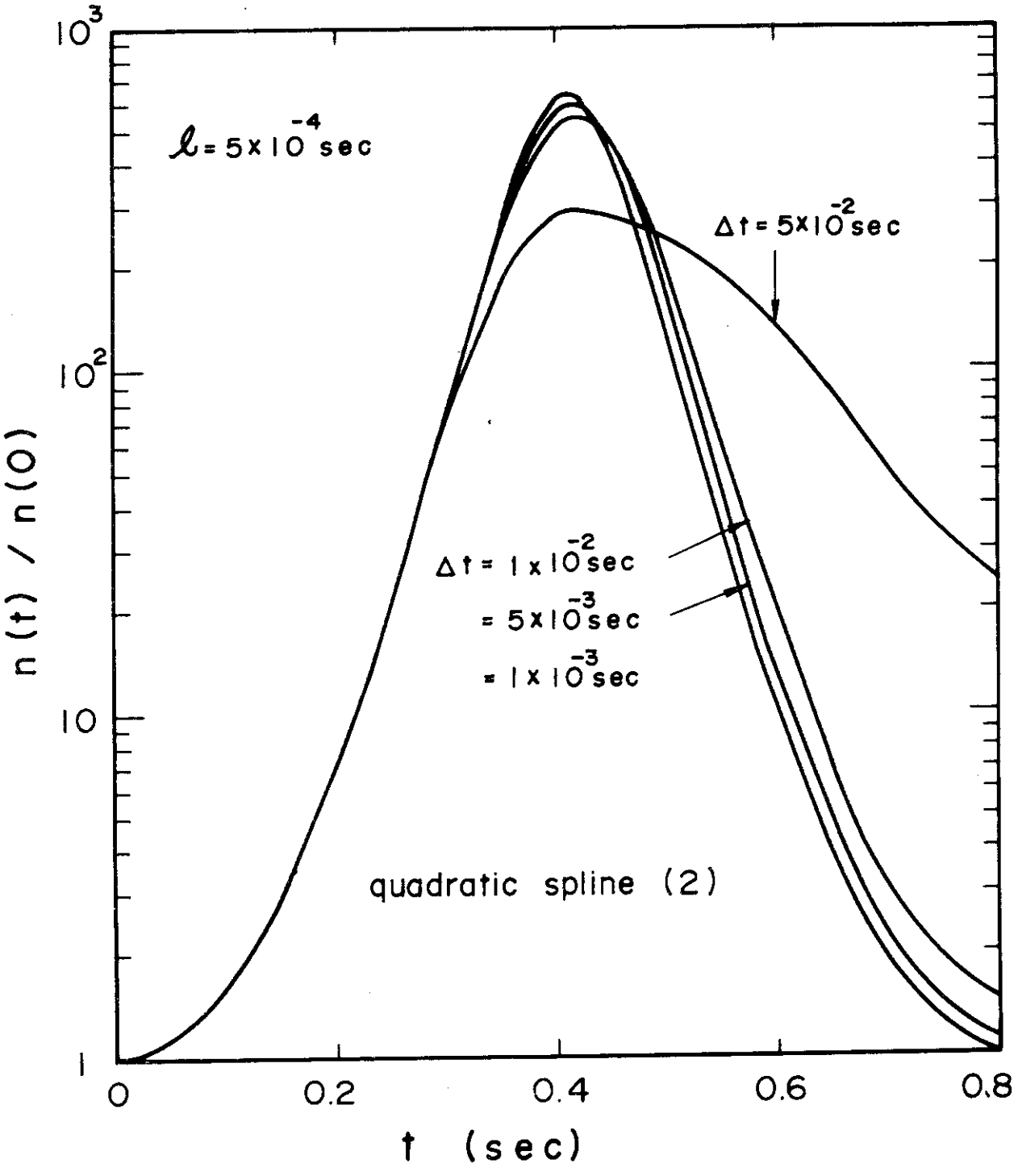

Fig. 4 Convergence behaviour of the approximate solutions by the quadratic spline method with the algorithm II

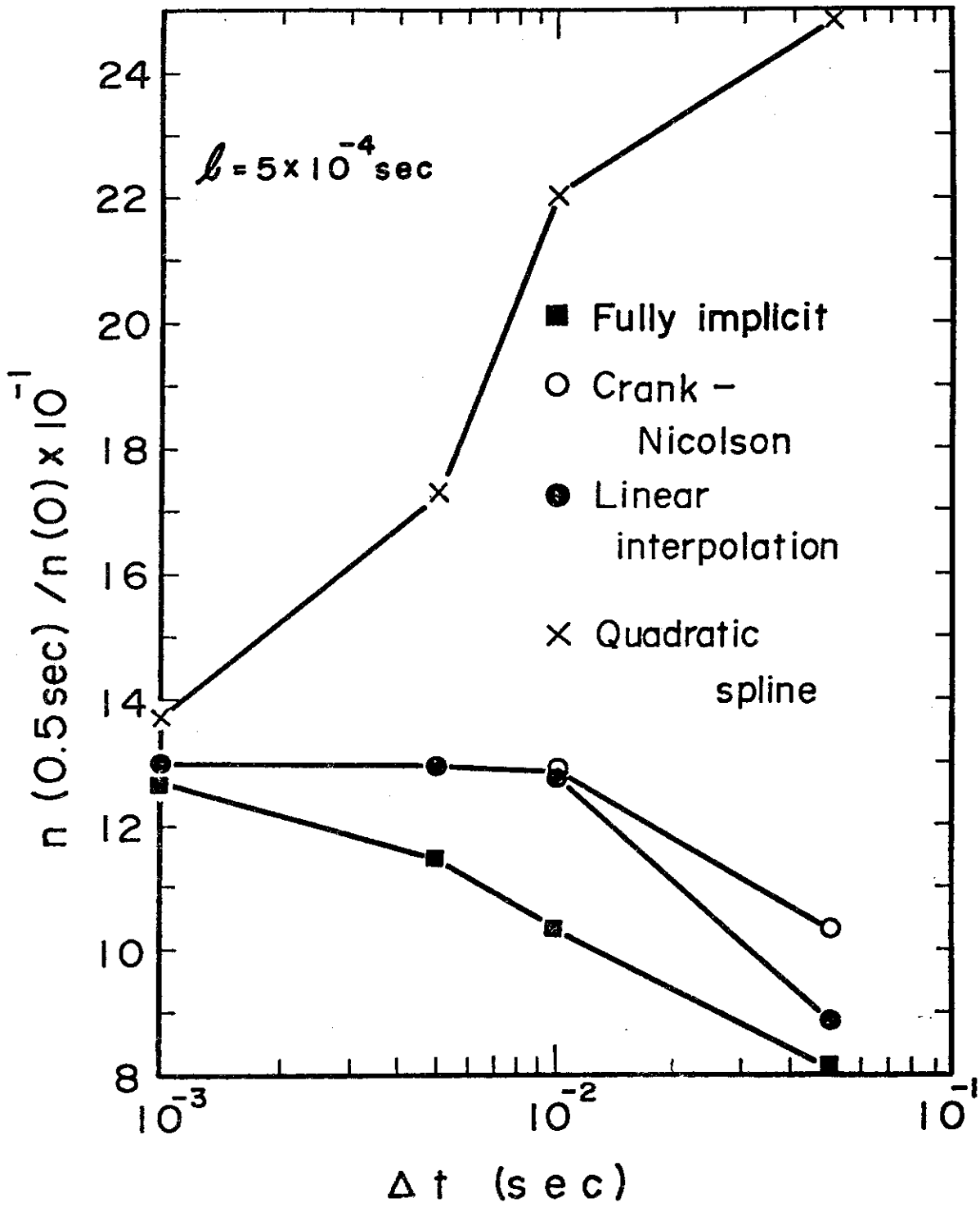

Fig. 5 Comparison of convergency of approximate solutions at 0.5 sec after the start of the reactivity insertion

Table 2. Comparison of accuracies of direct methods for a thermal reactor problem in values of $V(x) = (\gamma/\alpha C)n(t)$ at $x = 3.1$

Methods	Time step width (sec)			Accuracy
	0.01	0.005	0.001	
Fully implicit	1.5320 (53.20%)	1.2276 (22.76%)	1.0397 (3.97%)	*
Crank-Nicolson	1.0127 (1.27%)	1.0025 (0.25%)	0.99924 (-0.076%)	* * * *
Linear interpolation	1.0162 (1.62%)	1.0033 (0.33%)	0.99927 (-0.073%)	* * * *
Quadratic spline(1)	0.9517 (-4.83%)	0.9740 (-3.60%)	1.049 (4.9%)	* *
Quadratic spline(2)	0.9532 (-4.68%)	0.9742 (-2.58%)	0.9939 (-0.61%)	* * *

$x = \sqrt{\alpha/l}\, t$.
Standard value: $V(x=3.1) = 1.000$.
Values in parentheses are relative errors.
More * ' s, more accurate.

Table 3. Comparison of stability of linear and nonlinear problems

Methods	Temperature feedback	
	Without(1)	With(2)
Fully explicit	Unconditionally stable	*
Fully implicit	*	Unconditionally stable
Crank-Nicolson	* *	* * * *
Linear interpolation	* * *	* * *
Quadratic spline(2)	———	* *
Cubic Hermite spline	* * * * ———	——— Unstable

Rank of conditional stability : More * s, more stable.
(1) C. M. Kang and K. H. Hansen; (2) Y. Nakahara et al.

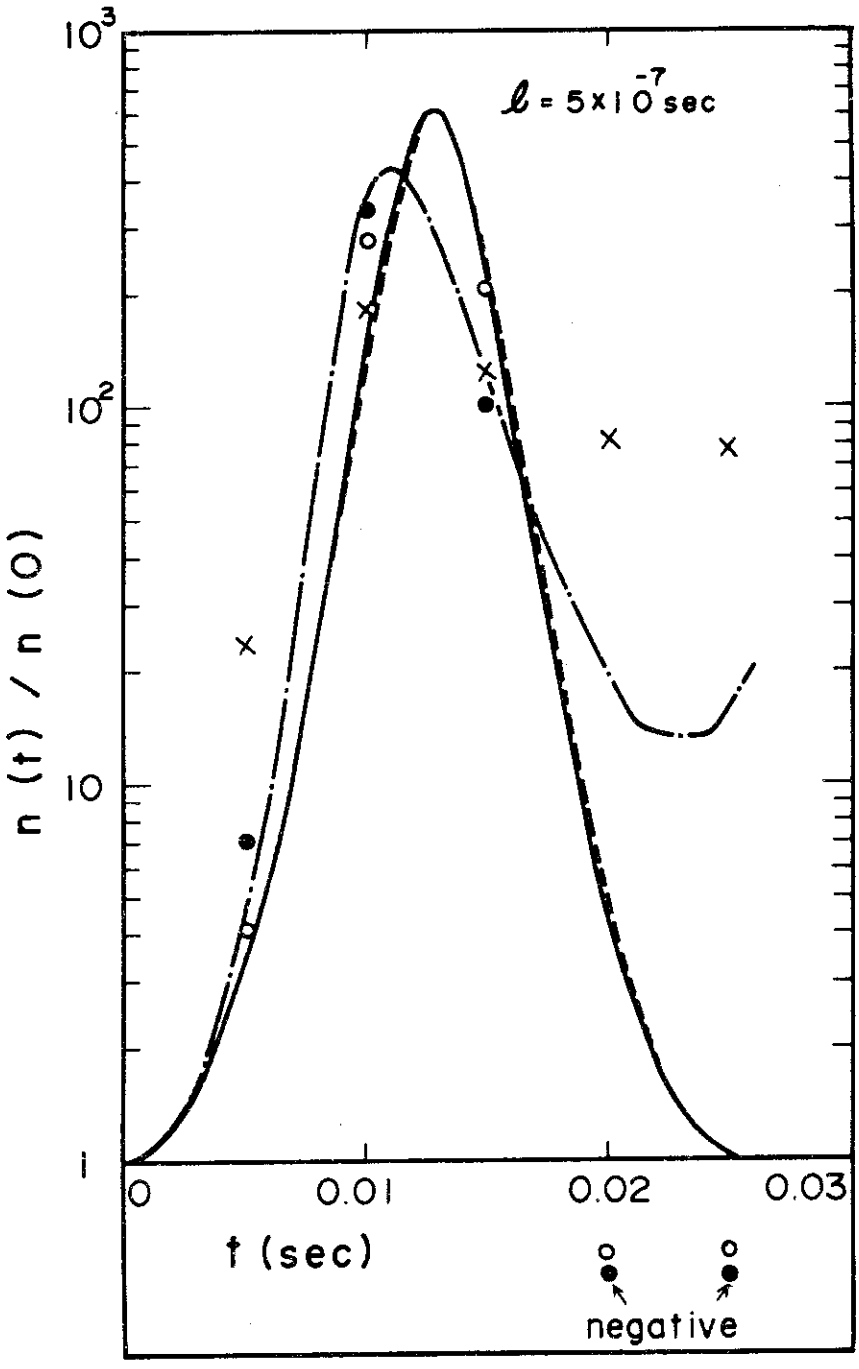

Fig. 6 Comparison of approximate solutions of the fast reactor model problem for coarse mesh widths of 2,000 and 10,000 times of ℓ

Methods	$t = 10^{-3}$ sec	$t = 5.10^{-3}$ sec
Fully implicit	—— - ——	x
Crank-Nicolson	----------	o
Linear	————	•

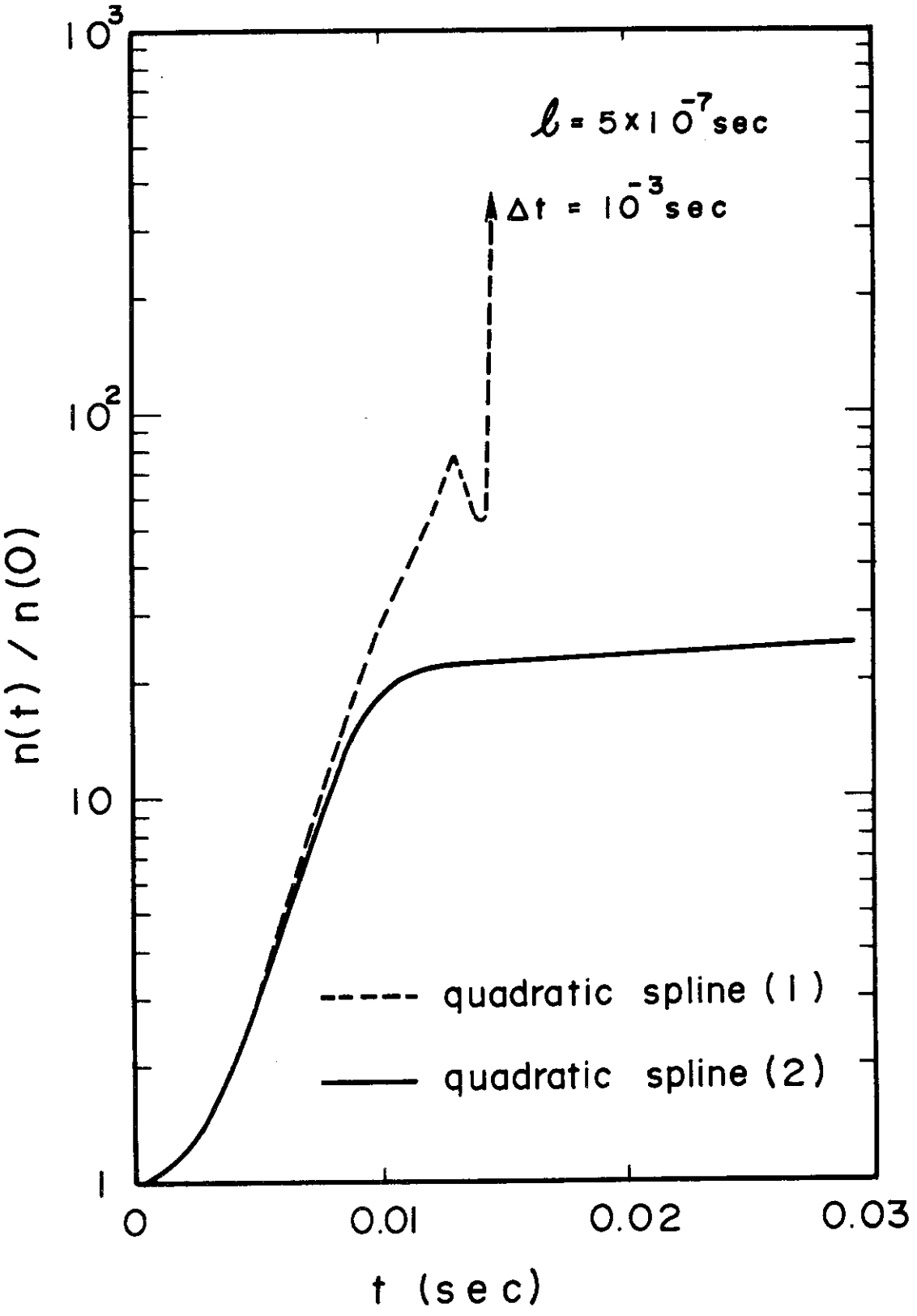

Fig. 7 Divergent behavior of the approximate solutions by the quadratic spline method for coarse mesh width of 2,000 times of ℓ

Table 4. Comparison of the CPU time

Methods	msec/step
Fully implicit	4.81
Crank-Nicolson	4.52
Linear interpolation	9.09
Quadratic spline (2)	10.25

FACOM 230/60

5. Application to a Nonlinear Fusion Reactor Kinetics Equation

In a one dimensional axisymmetric model for a 50-50 D-T burning Tokamak, in which particle loss is due to radial drift diffusion, the particle continuity equation is of the form[14),15),16)]

$$\frac{\partial}{\partial t} n(r,t) = \frac{1}{r}\frac{\partial}{\partial r}\left(Dr\frac{\partial}{\partial r} n\right) + S - n^2 Q. \tag{54}$$

Although Eq. (54) is not a Volterra equation but a partial differential equation, we briefly discuss about it here as one of other examples to which our direct method can be applied. Definitions of symbols in Eq. (54) are as follows;

$n(r,t)$ = ion density (cm^{-3})

D = drift diffusion coefficient (cm^2/sec)

S = ion source (ionization of injected neutrals)

Q = rate coefficient for recombination.

When we solve Eq. (54), D, S and Q are treated as constants.

As an example, we use the weighted step function method for the time interval. Integration of both sides of Eq. (54) over a subinterval $[t_i, t_{i+1}]$ and subregion $[r_j, r_{j+1}]$ with the aid of the interpolate

$$n(r,t) = \theta\, n_{i+1}(r) + (1-\theta) n_i(r) \tag{55}$$

can be written as follows,

$$\begin{aligned}\int_{r_j}^{r_{j+1}} [n_{i+1}(r) - n_i(r)]\, r\,dr = h_i \Big\{ & \theta\, Dr \frac{\partial}{\partial r} n_{i+1}(r)\Big|_{r_j}^{r_{j+1}} \\ & + (1-\theta)\, Dr \frac{\partial}{\partial r} n_i(r)\Big|_{r_j}^{r_{j+1}} + \frac{1}{2}(r_{j+1} + r_j)\Delta r_j S \\ & - Q \int_{r_j}^{r_{j+1}} [\theta\, n_{i+1}(r) + (1-\theta)\, n_i(r)]^2\, r\,dr. \end{aligned} \tag{56}$$

Eq. (56) contains the first space derivatives, so that the weighted step function method cannot be applied to spatial subregions. One of the lowest possible degree is the linear interpolation method. Then we have for the derivatives,

$$D r \frac{\partial}{\partial r} n_{i+1}(r)\Big|_{r_j}^{r_{j+1}} = D\left(r_{j+1} n_{i+1,j+1} - r_j n_{i+1,j}\right),$$

$$D r \frac{\partial}{\partial r} n_i(r)\Big|_{r_j}^{r_{j+1}} = D\left(r_{j+1} n_{i,j+1} - r_j n_{i,j}\right), \tag{57}$$

and for the integrals

$$\int_{r_j}^{r_{j+1}} [n_{i+1}(r) - n_i(r)] r\,dr = \frac{\Delta r_j}{6}\left[(2r_{j+1} + r_j) n_{i+1,j+1} + (r_{j+1} + 2r_j) n_{i+1,j} - (2r_{j+1} + r_j) n_{i,j+1} - (r_{j+1} + 2r_j) n_{i,j}\right], \tag{58}$$

$$\begin{aligned}\int_{r_j}^{r_{j+1}} [\theta n_{i+1}(r) + (1-\theta) n_i(r)]^2 r\,dr = \frac{\Delta r_j}{12}\Big\{&\theta^2\left[(3r_{j+1} + r_j)(n_{i+1,j+1})^2 + 2(r_{j+1} + r_j) n_{i+1,j+1} n_{i+1,j} + (r_{j+1} + 3r_j)(n_{i+1,j})^2\right] \\ &+ 2\theta(1-\theta)\left[(3r_{j+1} + r_j) n_{i+1,j+1} n_{i,j+1} + (r_{j+1} + r_j)(n_{i+1,j+1} n_{i,j} + n_{i+1,j} n_{i,j+1}) + (r_{j+1} + 3r_j) n_{i+1,j} n_{i,j}\right] \\ &+ (1-\theta)^2\left[(3r_{j+1} + r_j)(n_{i,j+1})^2 + 2(r_{j+1} + r_j) n_{i,j+1} n_{i,j} + (r_{j+1} + 3r_j)(n_{i,j})^2\right]\Big\}.\end{aligned} \tag{59}$$

Substituting Eqs. (57), (58) and (59) in Eq. (56) and performing some tedious arithemetic exercises, we finally get an algebraic quadratic equation for $n_{i+1,j+1}$, i.e.,

$$A_j (n_{i+1,j+1})^2 + B_{ij} n_{i+1,j+1} + C_{ij} = 0, \tag{60}$$

where putting $\Delta r_j = s_j$,

$$A_j = \theta^2 \frac{s_j^2}{12} (3r_{j+1} + r_j) Q,$$

$$B_{ij} = \frac{s_j}{6h_i} (2r_{j+1} + r_j) - \theta D r_{j+1} + \theta^2 \frac{s_j}{6} (r_{j+1} + r_j) Q n_{i+1,j}$$
$$+ \theta(1-\theta) \frac{s_j}{6} Q [(3r_{j+1} + r_j) n_{i,j+1} + (r_{j+1} + r_j) n_{ij}],$$

$$C_{ij} = \frac{s_j}{6h_i} [(r_{j+1} + 2r_j) n_{i+1,j} - (2r_{j+1} + r_j) n_{i,j+1}$$
$$- (r_{j+1} + 2r_j) n_{ij}] - \frac{1}{2} (r_{j+1} + r_j) s_j S + \theta D r_j n_{i+1,j}$$
$$- (1-\theta) D (r_{j+1} n_{i,j+1} - r_j n_{ij}) + \theta^2 \frac{s_j}{12} (r_{j+1} + 3r_j) Q (n_{i+1,j})^2$$
$$+ \theta(1-\theta) \frac{s_j}{6} [(r_{j+1} + r_j) n_{i+1,j} n_{i,j+1} + (r_{j+1} + 3r_j) n_{i+1,j} n_{ij}] Q$$
$$+ (1-\theta)^2 \frac{s_j}{12} [(3r_{j+1} + r_j)(n_{i,j+1})^2 + (2r_{j+1} + 2r_j) n_{i,j+1} n_{ij}$$
$$+ (r_{j+1} + 3r_j)(n_{ij})^2] Q. \tag{61}$$

Thus the solution we want to get is given by

$$n_{i+1,j+1} = \frac{-B_{ij} + \sqrt{(B_{ij})^2 - 4A_i C_{ij}}}{2A_i}. \tag{62}$$

6. Conclusions and Discussions

In the application of our algorithm to point reactor kinetics problems for thermal and fast fission reactor, it is concluded from the estimation of the stability, convergency and CPU time that the best is the Crank-Nicolson method and the linear interpolation comes closely next to it. These method gives sufficiently accurate solutions even with very coarse mesh widths of 100 $\sim$ 1,000 times of the prompt neutron lifetime.

It is interesting and perhaps favourable for us to have got the conclusion that the lower degree methods are most suited to our algorithm for nonlinear equations, which are cumbersome in solving but stimulative in researching. The Volterra integro-differential, partial differential and ordinary differential equations with the quadratic nonlinearity appear in many fields of science and technology. Many other examples will be found, to which our algorithm is applicable. Equations of cubic nonlinearity can also be solved analogously. In this case we derive a piecewise algebraic cubic equation, which has three roots in general. When these three roots are real and positive, some care will be needed in choosing only one root which has a plausible physical meaning.

Perhaps, we cannot pass without commenting on the space and time dependent fission reactor kinetics. In space dependent kinetics equations nonlinear feedback effects may not be expressed explicitly by nonlinear terms of definite degree of nonlinearity. Neutron cross sections have complex

dependence on temperature, such as the Doppler effect. The neutron fluxes and spectra also undergo different changes pertaining to the material composition of each reactor zone. A simple feedback model would be of limited usefulness to describe the whole aspect of possible reactor transients and accidents. Thus our algorithm does not offer a direct application to the space dependent fission reactor problems. Some other effective algorithms have to be exploited. In any way, some iterations would be necessary for numerical calculations, not from the mathematical point of view but rather from the physical point of view, that is to say, because space and time dependence of temperature must be given by solving simultaneously the heat balance equation.

As was shown in Chapter 5, our algorithm finds its further extention to the space dependent fusion rather than fission reactor kinetics.

References

1) Nakahara, Y., Ise, T. and Kobayashi, K. : "A Direct Method for the Numerical Solution of a Class of Nonlinear Volterra Integro-Differential Equations", (1975), not yet published

2) Froehlich, R. and Johnson, S.R. : Nukleonik, 12, 93 (1969)

3) Argonne Code Center : "Benchmark Problem Book", ANL-7416 (Suppl. 1), 164 (1972)

4) See for example Anselone, P.M. (ed.) : "Nonlinear Integral Equations", The Univeristy of Wisconsin Press, Madison, Wis. (1964)
Ortega, J.M. and Rheinboldt, W.C. : "Iterative Solution of Nonlinear Equations in Several Variables", Academic Press (1970)

5) Amann, H. : Numer. Math., 19, 29 (1972)

6) Prenter, P.M. : "Splines and Variational Method", A Wiley-Interscience Publ. (1975)
There are several other texts on the subject, but this book seems to be not only most comprehensive but readable. References cited in it are also very helpful for many readers.
See also

7) Strang, G. and Fix, G. : "Analysis of the Finite Element Method", Prentice-Hall (1973)

8) Guzek, J.A. and Kemper, G.A. : Math. Comput., 27, 563 (1973)

9) Loscalzo, F.R. and Talbot, T.D. : SIAM J. Numer. Anal., 4, 433 (1967)

10) Kemper, G.A. : SIAM J. Numer. Anal., 12, 73 (1975)

11) Kang, C.M. and Hansen, K.F. : "Finite Element Methods for Space-Time Reactor Analysis", MIT-3903-5 (1971) or Nucl. Sci. Eng., 51, 456 (1973)

12) Feldstein, A. and Sopka, J.R. : SIAM J. Numer. Anal., 11, 827 (1974)

13) Garey, L. : "Proceedings of the Fourth Manitoba Conference on Numerical Mathmatics", p. 253, University of Manitoba, Winnipeg (1974)
BIT, 14, 33 (1974)
SIAM. J. Numer. Anal., 12, 501 (1975)

14) Steuerwald, J. : "Untersuchungen über die Monotonie von Differenzen-Schemata für Diffusionsgleichungen", IPP 6/135 (1975)

15) Keilhacker, M. : Numerical Calculations of the Density and Temperature Variations in Cylindrical Plasmas Preionized by UV-Radiation", IPP-1/106 (1970)

16) Stacey, W.M. Jr. : Nuclear Fusion, 15, 163 (1975)

Appendix 1

Integration of $\int_{t_i}^{t_{i+1}} dt \int_0^t ds\, K(t,s,n(s))$

Defining following integrals

$$I_1 = \int_{t_i}^{t_{i+1}} dt\; t n(t),$$

$$I_{2j} = \int_{t_i}^{t_{i+1}} dt\; e^{-\lambda_j t},$$

$$I_{3j} = \int_{t_i}^{t_{i+1}} dt \int_0^t ds\, n(s) e^{-\lambda_j (t-s)},$$

we write

$$\int_{t_i}^{t_{i+1}} dt \int_0^t ds\, K(t,s,n(s)) = \frac{\alpha}{\ell} I_1 + \sum_{j=1}^{J} \lambda_j \left[R_{j0} I_{2j} + \frac{\beta_j}{\ell} I_{3j} \right].$$

I_{2j} does not depend on n(t), so that

$$I_{2j} = \frac{1}{\lambda_j} \left(e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}} \right)$$

for any algorithms.

(1) Fully implicit, fully explicit and Crank-Nicolson methods

$$I_1 = h_i \left(t_i + \frac{h_i}{2} \right) \left[\theta n_i + (1-\theta) n_{i+1} \right].$$

$$I_{3j} = \int_{t_i}^{t_{i+1}} dt\, e^{-\lambda_j t} \left\{ \frac{1}{\lambda_j} \sum_{k=1}^{i-1} \left[\theta n_k + (1-\theta) n_{k+1} \right] \left(e^{\lambda_j t_{k+1}} - e^{\lambda_j t_k} \right) \right.$$

$$\left. + \frac{1}{\lambda_j} \left[\theta n_i + (1-\theta) n_{i+1} \right] \left(e^{\lambda_j t} - e^{\lambda_j t_i} \right) \right\}$$

$$= \frac{1}{\lambda_j^2}(e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}})\left\{\sum_{k=1}^{i-1}[\theta n_k + (1-\theta)n_{k+1}](e^{\lambda_j t_{k+1}} - e^{\lambda_j t_k})\right.$$

$$\left. + \frac{1}{\lambda_j}[\theta n_i + (1-\theta)n_{i+1}]\left[h_i - \frac{1}{\lambda_j}(1 - e^{-\lambda_j h_i})\right]\right.$$

(2) The linear interpolation method

$$I_1 = \frac{h_i}{6}(2t_{i+1} + t_i)n_{i+1} + \frac{h_i}{6}(t_{i+1} + 2t_i)n_i .$$

$$I_{3j} = \frac{1}{\lambda_j}\sum_{k=1}^{i-1}\frac{1}{h_k}\left\{\left[e^{\lambda_j t_{k+1}}\left(h_k - \frac{1}{\lambda_j}\right) + \frac{1}{\lambda_j}e^{\lambda_j t_k}\right]n_{k+1}\right.$$

$$\left. - \left[e^{\lambda_j t_k}\left(h_k + \frac{1}{\lambda_j}\right) - \frac{1}{\lambda_j}e^{\lambda_j t_{k+1}}\right]n_k\right\}(e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}})\frac{1}{\lambda_j}$$

$$+ \frac{1}{\lambda_j h_i}\left\{\left[\frac{1}{\lambda_j^2}(1 - e^{-\lambda_j h_i}) + h_i\left(\frac{h_i}{2} - \frac{1}{\lambda_j}\right)\right]n_{i+1}\right.$$

$$\left. - \left[\left(h_i + \frac{1}{\lambda_j}\right)\frac{1}{\lambda_j}(1 - e^{-\lambda_j h_i}) - h_i\left(\frac{h_i}{2} + \frac{1}{\lambda_j}\right)\right]n_i\right\}.$$

(3) The quadratic spline method

$$I_1 = \frac{h_i}{2}(t_{i+1} + t_i)n_i + \frac{h_i^2}{2}\left(t_{i+1} - \frac{h_i}{2}\right)\dot{n}_i + \frac{h_i}{3}\left(t_{i+1} - \frac{h_i}{4}\right)$$

$$\times (n_{i+1} - n_i - h_i\dot{n}_i).$$

$$I_{3j} = \frac{1}{\lambda_j^2}(e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}})\sum_{k=1}^{i-1}\left\{(e^{\lambda_j t_{k+1}} - e^{\lambda_j t_k})n_k\right.$$

$$+ \left[e^{\lambda_j t_{k+1}}\left(h_k - \frac{1}{\lambda_j}\right) + \frac{1}{\lambda_j}e^{\lambda_j t_k}\right]\dot{n}_k$$

$$\left. + \frac{1}{h_k^2}\left[e^{\lambda_j t_{k+1}}\left(h_k^2 - \frac{2h_k}{\lambda_j} + \frac{2}{\lambda_j^2}\right) - \frac{1}{\lambda_j^2}e^{\lambda_j t_k}\right](n_{k+1} - n_k - h_k\dot{n}_k)\right\}$$

$$+ \frac{1}{\lambda_j}\left\{\left[h_i - \frac{1}{\lambda_j}(1 - e^{-\lambda_j h_i})\right]n_i + \left[\frac{h_i^2}{2} - \frac{h_i}{\lambda_j} + \frac{1}{\lambda_j^2}(1 - e^{-\lambda_j h_i})\right]\dot{n}_i\right.$$

$$\left. + \frac{1}{h_i^2}\left[\frac{h_i^3}{3} - \frac{h_i^2}{\lambda_j} + \frac{2h_i}{\lambda_j^2} - \frac{1}{\lambda_j^3}(1 - e^{-\lambda_j h_i})\right](n_{i+1} - n_i - h_i\dot{n}_i)\right\}.$$

(4) The cubic spline method

$$I_1 = h_i\left(t_i + \frac{h_i}{2}\right) n_i + h_i^2\left(\frac{t_i}{2} + \frac{h_i}{3}\right)\dot{n}_i + \frac{h_i^3}{2}\left(\frac{t_i}{3} + \frac{h_i}{4}\right)\ddot{n}_i$$
$$+ \frac{h_i^2}{3}\left(\frac{t_i}{4} + \frac{h_i}{5}\right)\left(\dot{n}_{i+1} - \dot{n}_i - h_i\ddot{n}_i\right).$$

$$I_{3j} = \frac{1}{\lambda_j^2}\left(e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}}\right)\sum_{k=1}^{i-1}\left\{\left(e^{\lambda_j t_{k+1}} - e^{\lambda_j t_k}\right) n_k\right.$$
$$+\left[e^{\lambda_j t_{k+1}}\left(h_k - \frac{1}{\lambda_j}\right) + \frac{1}{\lambda_j}e^{\lambda_j t_k}\right]\dot{n}_k$$
$$+\frac{1}{2}\left[e^{\lambda_j t_{k+1}}\left(h_k^2 - \frac{2h_k}{\lambda_j} + \frac{2}{\lambda_j^2}\right) - \frac{2}{\lambda_j^2}e^{\lambda_j t_k}\right]\ddot{n}_k$$
$$\left.+\frac{1}{3h_k^2}\left[e^{\lambda_j t_{k+1}}\left(h_k^3 - \frac{3h_k^2}{\lambda_j} + \frac{6h_k}{\lambda_j^2} - \frac{6}{\lambda_j^3}\right) + \frac{6}{\lambda_j^3}e^{\lambda_j t_k}\right]\left(\dot{n}_{k+1} - \dot{n}_k - h_k\ddot{n}_k\right)\right\}$$
$$+\frac{1}{\lambda_j}\left\{\left[h_i - \frac{1}{\lambda_j}\left(1 - e^{-\lambda_j h_i}\right)\right] n_i + \left[\frac{h_i^2}{2} - \frac{h_i}{\lambda_j} + \frac{1}{\lambda_j^2}\left(1 - e^{-\lambda_j h_i}\right)\right]\dot{n}_i\right.$$
$$+\frac{1}{2}\left[\frac{h_i^3}{3} - \frac{h_i^2}{\lambda_j} + \frac{2h_i}{\lambda_j^2} - \frac{1}{\lambda_j^3}\left(1 - e^{-\lambda_j h_i}\right)\right]\ddot{n}_i$$
$$\left.+\frac{1}{3h_i^2}\left[\frac{h_i^4}{4} - \frac{h_i^3}{\lambda_j} + \frac{3h_i^2}{\lambda_j^2} - \frac{6h_i}{\lambda_j^3} + \frac{1}{\lambda_j^4}\left(1 - e^{-\lambda_j h_i}\right)\right]\left(\dot{n}_{i+1} - \dot{n}_i - h_i\ddot{n}_i\right)\right\}.$$

Appendix 2

Expressions of A_i, B_i and C_i for the cubic spline method

$$A_i = \frac{\gamma}{C\ell}\frac{h_i^4}{288},$$

$$B_i = \frac{h_i}{3} + \frac{\beta}{\ell}\frac{h_i^2}{12} - \frac{\alpha}{\ell}K_i^4 + \frac{\gamma}{C\ell}\left\{\frac{h_i^2}{12}P_{i-1} + \sum_{l=4}^{7}\frac{h_i^{l+1}}{l+1}M_l + \frac{h_i^3}{12}\left[\frac{1}{5}n_i + \frac{h_i}{6}\dot{n}_i + \frac{h_i^2}{14}\ddot{n}_i - \frac{h_i}{24}(\dot{n}_i + h_i\ddot{n}_i)\right]\right\} - \sum_{j=1}^{J}\frac{\beta_j\lambda_j}{\ell}I_{ij}^3,$$

$$C_i = \frac{\beta}{\ell}T_i - \frac{\alpha}{\ell}[K_i^1 n_i + K_i^2\dot{n}_i + K_i^3\ddot{n}_i - K_i^4(\dot{n}_i + h_i\ddot{n}_i)] + \frac{\gamma}{C\ell}\left(T_i P_{i-1} + \sum_{l=1}^{7}\frac{h_i^{l+1}}{l+1}N_l\right) - \sum_{j=1}^{J}\left\{(e^{-\lambda_j t_i} - e^{-\lambda_j t_{i+1}})[R_{j0} + \frac{\beta}{\ell}S_{i-1,j}] + \frac{\beta_j\lambda_j}{\ell}[I_{ij}^0 n_i + (I_{ij}^1 - I_{ij}^3)\dot{n}_i + (I_{ij}^2 - I_{ij}^3)\ddot{n}_i]\right\} + \frac{2}{3}h_i\dot{n}_i + \frac{h_i^2}{6}\ddot{n}_i,$$

where

$$P_i = P_{i-1} + h_i n_i + \frac{h_i^2}{2}\dot{n}_i + \frac{h_i^3}{6}\ddot{n}_i + \frac{h_i^2}{12}(\dot{n}_{i+1} - \dot{n}_i - h_i\ddot{n}_i),$$

$$T_i = h_i n_i + \frac{h_i^2}{2}\dot{n}_i + \frac{h_i^3}{6}\ddot{n}_i - \frac{h_i^2}{12}(\dot{n}_i + h_i\ddot{n}_i),$$

$$K_i^1 = \frac{h_i}{2}(t_{i+1} + t_i), \quad K_i^2 = \frac{h_i^2}{2}\left(t_{i+1} - \frac{h_i}{3}\right),$$

$$K_i^3 = \frac{h_i^3}{6}\left(t_{i+1} - \frac{h_i}{4}\right), \quad K_i^4 = \frac{h_i^2}{12}\left(t_{i+1} - \frac{h_i}{5}\right),$$

$$S_{k,j} = S_{k-1,j} + I^{0}_{k,j}\,n_k + I^{1}_{k,j}\,\dot{n}_i + I^{2}_{k,j}\,\ddot{n}_i + I^{3}_{k,j}\,(\dot{n}_{k+1} - \dot{n}_k - h_k\ddot{n}_k),$$

$$S_{1,j} = 0,$$

$$I^{o}_{k,j} = \frac{1}{\lambda_j}\left(e^{\lambda_j t_{k+1}} - e^{\lambda_j t_k}\right) \quad , \text{ for } k = 1, 2, \cdots, i-1$$

$$I^{1}_{k,j} = \frac{1}{\lambda_j}\left(h_k e^{\lambda_j t_{k+1}} - I^{o}_{k,j}\right) \quad ,$$

$$I^{2}_{k,j} = \frac{1}{\lambda_j}\left(h_k^2 e^{\lambda_j t_{k+1}} - I^{1}_{k,j}\right) \quad ,$$

$$I^{3}_{k,j} = \frac{1}{\lambda_j}\left(\frac{h_k}{3} e^{\lambda_j t_{k+1}} - \frac{2}{h_k^2} I^{2}_{k,j}\right),$$

$$I^{o}_{i,j} = \frac{1}{\lambda_j}\left[h_i + \frac{1}{\lambda_j}\left(e^{-\lambda_j n_i} - 1\right)\right],$$

$$I^{1}_{i,j} = \frac{1}{\lambda_j}\left(\frac{h_i^2}{2} - I^{o}_{i,j}\right) \quad ,$$

$$I^{2}_{i,j} = \frac{1}{\lambda_j}\left(\frac{h_i^3}{6} - I^{1}_{i,j}\right) \quad ,$$

$$I^{3}_{i,j} = \frac{1}{\lambda_j}\left(\frac{h_i^2}{12} - \frac{2}{h^2} I^{2}_{i,j}\right) \quad ,$$

$$N_1 = (n_i)^2 \quad , \quad N_2 = \frac{2}{3} \dot{n}_i n_i \quad , \quad N_3 = \frac{2}{3} \ddot{n}_i n_i + \frac{1}{2} (\dot{n}_i)^2 \quad ,$$

$$N_4 = -\frac{5}{12} \frac{n_i}{h_i^2} (\dot{n}_i + h_i \ddot{n}_i) + \frac{5}{12} \dot{n}_i \ddot{n}_i \quad ,$$

$$N_5 = -\frac{1}{4h_i^2} \dot{n}_i (\dot{n}_i + h_i \ddot{n}_i) + \frac{1}{12} (\ddot{n}_i)^2 \quad ,$$

$$N_6 = -\frac{7}{72} \ddot{n}_i (\dot{n}_i + h_i \ddot{n}_i) \quad , \quad N_7 = \frac{1}{36 h_i^4} (\dot{n}_i + h_i \ddot{n}_i)^2 \quad ,$$

$$M_4 = \frac{1}{3h_i^2} n_i \quad , \quad M_5 = \frac{1}{6h_i^2} \dot{n}_i \quad , \quad M_6 = \frac{1}{18 h_i^2} \ddot{n}_i \quad ,$$

$$M_7 = -\frac{1}{36 h_i^4} (\dot{n}_i + h_i \ddot{n}_i) \, .$$