Book Chapter

High-Order Accurate and High-Speed Calculation System of 1D Laplace and Poisson Equations Using the Interpolation Finite Difference Method

Tsugio Fukuchi*

Tsubokura Ground-Survey and Design Ltd., Japan

***Corresponding Author:** Tsugio Fukuchi, Tsubokura Ground-Survey and Design Ltd., Fukushima 960-0113, Japan

Published **June 15, 2020**

This Book Chapter is a republication of an article published by Tsugio Fukuchi at AIP Advances in May 2019. (Tsugio Fukuchi. High-Order Accurate and High-Speed Calculation System of 1D Laplace and Poisson Equations Using the Interpolation Finite Difference Method. AIP Advances. 9, 055312 (2019); <https://doi.org/10.1063/1.5096395>.)

How to cite this book chapter: Tsugio Fukuchi. High-Order Accurate and High-Speed Calculation System of 1D Laplace and Poisson Equations Using the Interpolation Finite Difference Method. In: Vishnu Gopal, editor. Prime Archives in Physical Sciences. Hyderabad, India: Vide Leaf. 2020.

© The Author(s) 2020. This article is distributed under the terms of the Creative Commons Attribution 4.0 International License(<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Among the methods of the numerical analysis of the physical phenomena of the continuum, the finite difference method (FDM) is the first examined method and has been established as a full numerical calculation system over the regular domain. However, there is a general perception that generality in numerical calculations cannot be expected over complex irregular domains. As using the FDM, the development of computational methods that are applicable over any irregular domain is considered to be a very important contemporary problem. In the FDM, there is a marked characteristic that the theory developed by the (spatial) one-dimensional (1D) problem is naturally applied to the 2D and 3D problems. The calculation method is called the interpolation FDM (IFDM). In this paper, attention is paid to 1D Laplace and Poisson equations, and the whole image of the IFDM using the algebraic polynomial interpolation method (APIM), the IFDM-APIM, is described. Based on the Lagrange interpolation function, the spatial difference schemes from 2nd order to 10th order including odd order are calculated and defined instantaneously over equally/unequally spaced grid points, then, high-order accurate and high-speed computations become possible.

Introduction

The FDM is a major numerical calculation method mainly used in the field of fluid dynamics. The common recognition of the FDM up to around 2000 was as follows.

“FDM is the oldest method for numerical solutions of partial differential equations (PDEs). On structured grids, FDM is very simple and effective. It is especially easy to obtain higher-order schemes on regular grids. A disadvantage of FDM is that certain conservation laws are not enforced unless special care is taken. Furthermore, the restriction to simple geometries is a significant disadvantage if complex flows are specified” [1].

Hoffmann and Chiang [2] issued a comprehensive guide on the FDM and stated the problems when applying the FDM to numerical analysis in arbitrary domains as follows.

“Simple rectangular boundaries rarely occur in nature; indeed, most boundaries are irregular. Irregular boundaries create tremendous difficulties in implementation of the boundary conditions. Various schemes are available to treat irregular boundaries. One may consider using variable step sizes in the neighborhood of the irregular boundaries. This procedure is cumbersome and most cases, very inefficient.” [Ref. 2, p.94]

Many ideas have been proposed to overcome the aforementioned difficulties with the FDM. The ghost cell method [3], which has been attempted in particular in the field of fluid dynamics, is one noteworthy example. Fukuchi's research [4-8] may be considered an attempt to exceed the conventional limits of the FDM. The IFDM was proposed as a general system subsuming the conventional FDM. Regarding the handling of spatial differences, the IFDM is categorized into the (i) algebraic polynomial interpolation method (APIM) and (ii) boundary polynomial interpolation method (BPIM) [6]. In the IFDM-BPIM calculation system, a method is adopted in which a dummy element is set at the boundary and a dummy value is obtained by successive interpolation in the calculation process so as to match the boundary condition. This method is in common with the idea of the ghost cell method [9-11]. From this perspective, the verification of the second-order accurate difference has been investigated with respect to its generality, and the effectiveness of numerical calculations over arbitrary domains has been confirmed [7,8]. There are two cases where all of the Dirichlet boundary conditions are present and the Dirichlet-non-Dirichlet mixed conditions exist as the boundary conditions. In any case of non-Dirichlet conditions in 2D problems, by successively setting the local curved surface at the boundary in the calculation process, the boundary condition comes down to the Dirichlet boundary condition. The mixed boundary conditions method is also incorporated into the IFDM-APIM calculation system, but since the subject is the 1D calculation method, it is not mentioned in this paper. The IFDM-

BPIM is applied in the case of equidistant grid points, but the IFDM-APIM can be used freely over regular and irregular grid points. Although its basic idea was previously reported [6], detailed examination results have not yet been demonstrated.

The Laplace equation is the governing equation in the study of steady-state thermal fields, electrostatic fields, potential flows in fluid dynamics, etc. Including the Poisson equation, which is a generalization of Laplace equation, both equations are the simplest examples of elliptic PDEs. In the paper, the Poisson equation in 3D space is expressed as follows:

$$\nabla_{(3)}^2 u \equiv \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = f(x, y, z) \quad (1)$$

As $f(x, y, z) = 0$, it reduces to the 3D Laplace equation. (The variable u is indicated in a conventional notation specific to each of the analysis fields, but in this paper, u is uniformly used.) The ultimate goal is to numerically analyze Eq. (1) over arbitrary domains, but in order to do so, we must first consider the 1D problem as comprehensively as possible.

In this paper, we will compare and verify basic conventional FDM numerical analysis methods that numerically analyze 1D Laplace and Poisson equations. The following seven calculation methods are systematically verified: (i) the forward time explicit (FTEX) method, (ii) the time marching successive displacement (TMSD) method, (iii) the successive over-relaxation (SOR) method, (iv) the direct method using the Gauss elimination method (GEM), (v) the direct method using the band diagonal matrix algorithm (BDMA), (vi) the implicit Crank-Nicolson method (CNM), and (vii) the full implicit method (FIM). In section II, focusing on the second-order accurate difference, we comprehensively examine the various calculation methods by IFDM-APIM, and confirm the convergence characteristics. In section III, based on the Lagrange interpolation function, we formulate a method to calculate high-order difference schemes instantly. At the same time, it is confirmed that various calculation methods based on the second-order accurate difference naturally develop to the high-order difference

calculation methods. In section IV, we numerically analyze four patterns of Poisson's equation using direct method with BDMA and investigate numerical analysis accuracy when using the second-order to tenth-order difference equations.

The Laplace Equation and its Finite Difference Equation

In the paper, the Laplace equation is generally expressed as $\nabla_{(n)}^2 u = 0$, where n is the dimension of the domain. As previously described, to progressively understand the simple equation compared to the complicated one, it is better to examine the calculation methods in the order of the 1D Laplace equation $\nabla_{(1)}^2 u = 0$, 2D Laplace equation $\nabla_{(2)}^2 u = 0$, and 3D Laplace equation $\nabla_{(3)}^2 u = 0$. This paper comprehensively considers the numerical calculation method by the IFDM-APIM, limited to 1D problems.

PDEs are classified into (i) parabolic PDEs, (ii) elliptic PDEs, and (iii) hyperbolic PDEs. This paper considers the first two. Conventionally, both numerical methods have each developed as independent numerical calculation methods, but from the perspective that the elliptic PDE is considered the limiting form of the time marching of parabolic PDEs, there is no distinction between them in numerical calculations. Therefore, in this paper, the basic equation for the numerical analysis of 1D Laplace equations is expressed as follows.

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \left(\equiv \nabla_{(1)}^2 u \right) \quad (2)$$

Under a fixed boundary condition, as $t \rightarrow \infty$, it becomes $\partial u / \partial t \rightarrow 0$, then $\nabla_{(1)}^2 u = 0$. The time differential term of the parabolic PDE has a physical meaning. However, the time differential term for solving the elliptic PDE is introduced as a means of numerical analysis. Let the time differential term in this case be defined as the pseudo-acceleration term and call this method the pseudo-acceleration method.

The method of solving parabolic PDEs is divided into an explicit method and an implicit method. At first, the forward time-centered space (FTCS) scheme expressed by the finite difference equation (FDE) below is adopted as a representation of the explicit method.

$$u_i^{n+1} = u_i^n + \Delta t \left(\frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{(\Delta x)^2} \right) \quad (3)$$

The time difference in the FTCS scheme is defined by the following equation [2,5].

$$\Delta t = \alpha_a \Delta t_c, \Delta t_c = 1 / \frac{2}{(\Delta x)^2}, 0 < \alpha_a \leq 1 (= \alpha_{amax}) \quad (4)$$

In this case, the criteria time difference Δt_c is equal to the reciprocal of the absolute value of the coefficient of u_i^n in the FDE of Eq. (3). This is evidence to define Δt_c in a high-order accurate difference scheme. The stable condition for which the numerical calculation does not diverge is to specify the acceleration factor α_a in the range of $0 < \alpha_a \leq 1$. In the case of the second-order accurate difference scheme, the maximum value $\alpha_{amax} = 1$ is theoretically derived by the discrete perturbation stability analysis or the Von Neumann stability analysis [Ref. 2, pp. 113-141].

The variable u depends on the time and place. Let us denote this as $u(t, x)$. The fundamental equation of the 1D thermal conduction problem is expressed by Eq. (2) due to deriving dimensionless equation [12]. In Figure 1(a), the problem of a so-called cooling rod is the problem of setting the temperature of A' and B' to zero, giving the initial condition $u(0, x)$ and obtaining the time change $u(t, x)$ of the temperature. This general solution and the solution when the initial condition is A'C'B' in Figure 1(a) are also known. Here, for the sake of explanation, let us treat the problem of the cooling rod as a heating rod problem. A boundary condition of $u_A(t, 0) = u_B(t, 1) = 1.00$ is given to the end points of A and B, and an initial condition is provided as

shown; the initial condition at the center point C is $u_C(0,0.5) = 0$. Figure 1(a) shows the numerical calculation result using the FTCS. The exact solution to this problem may be expressed as follows using a Fourier series.

$$u(x,t) = u_0 - \frac{8u_0}{\pi^2} \sum_{n=1}^{\infty} \frac{1}{n^2} \sin \frac{n\pi}{2} \sin \frac{n\pi x}{L} \exp \left[-u_0 \left(n\pi / L \right)^2 t \right] \quad (5)$$

where $u_0 = 1, L(= |\overline{AB}|) = 1$. (We attempted to calculate the case of the initial value of zero between points A and B, but a significant deviation occurs between the numerical solution and the theoretical solution at the initial stage of the calculation. This is thought to be due to the nature of the Fourier series. Therefore, the above initial condition is adopted.)

Consideration starts from the case of a regular grid disposition. Figure 1(a) is a calculation example in which the number of divisions is $N_x = 5 (\Delta x = 0.2)$. The numbers of grid points are defined as shown. There is a clear deviation between the numerical solution (dot and interpolation curve) and the theoretical solution (dashed line). The calculation accuracy needs to be evaluated separately for calculation accuracy of the time progression and the calculation accuracy of the convergence value. The error occurring in Figure 1(a) is the error with respect to time progression. However, regarding the convergence value, it is meaningless to compare and verify the accuracy of the spatial difference because the 1D problem here is a problem of the intrinsic-error-zero calculation [6], as described in detail later. This treatment has the advantage that the convergence characteristics of several numerical analysis methods can be clearly compared. Furthermore, if the number of divisions is $N_x = 6$, the result is strange as shown in Figure 1(b). This is a result of setting the singular point related to the initial condition at the center point C. Usually, since such a singular initial condition is not set and there may be no problem, it shows one characteristic of the numerical calculation by an explicit method. Even if $N_x = 10, \alpha_a = 1$ in Figure 1(c), there is a calculation point at the singular point, so unnaturalness remains. However, in the case of Figure 1(d), $N_x = 10, \alpha_a = 0.5$, the calculation result is sufficiently reliable. By choosing the appropriate

number of calculation points and time differences, we obtain reasonable calculation results.

Figure. 1: Time-dependent solution using the FTCS scheme. (a) $N_x = 5$, $\alpha_a = 1$, $N_C(3)=32$, $N_C(10)=108$, (b) $N_x = 6$, $\alpha_a = 1$, $N_C(3)=47$, $N_C(10)=159$, (c) $N_x = 10$, $\alpha_a = 1$, $N_C(3)=135$, $N_C(10)=455$, and (d) $N_x = 10$, $\alpha_a = 0.5$, $N_C(3)=271$, $N_C(10)=922$.

In Figures 1 (a), (b), and (c), since $\alpha_a = 1$, it becomes $\Delta t = \Delta t_c = (\Delta x)^2/2$, and then Eq. (3) leads to the following equation.

$$u_i^{n+1} = \frac{1}{2} (u_{i-1}^n + u_{i+1}^n) \quad (6)$$

This is the Jacobi iteration method (Ref. 2, pp. 159-162) in the 1D calculation. If we are interested only in the convergence value, let $\alpha_a = 1$ and check the number of iterations in which the

value at the (near) center point is within the specified range. The convergence confirmation point is $i_c = 2(x = 0.4)$ in Figure 1(a), $i_c = 3(x = 0.5)$ in Figure 1(b), and $i_c = 5(x = 0.5)$ in Figures 1(c) and (d). The convergence value is $u_{i_c}^c = 1$. Let us designate the convergence index $m_c (= 3 \text{ or } 10)$ and check the calculation step $N_c(m_c)$ that reaches the condition of $(u_{i_c}^c - u_{i_c}^n)/u_{i_c}^c < 10^{-m_c}$. The results are shown in the caption in Figure 1. These indexes indicate the convergence characteristic of the calculated sequence. Since the PC calculates with a 15-digit floating point, the maximum value of m_c is 15. From an engineering perspective, it can be considered sufficient convergence with $m_c = 3$. When evaluating precise errors, $m_c = 10$ is adopted. Table I (I) shows the calculation results including cases of other division numbers.

Let us consider the stability of numerical calculation. In Eq. (4), is $\alpha_{amax} = 1$ a reasonable definition in any case? α_{amax} is interpreted as the limit value at which the numerical calculation diverges if it is set larger than the value. This is theoretically determined by slightly disturbing the system in the equilibrium (stationary) state and confirming whether the disturbance decreases; however, it is possible to systematically change the value of α_a to check and determine the maximum value without divergence. The results are shown in Table II, α_{amax} changes depending on whether the number of divisions is an odd number or an even number and converges to 1 as N_x increases. In the method of deriving $\alpha_{amax} = 1$ theoretically [Ref. 2, pp. 113-141], the conclusion comes from the fact that the influence of finite division cannot be evaluated. Therefore, even in the case of adopting a high-order accurate scheme, N_x should be approximately 50 to obtain α_{amax} as a limit value by

Table I: Convergence characteristics of each calculation method.

(I)	FTCS, $\alpha_a = 1$				
N_x	5	10	20	50	100
α_a	1	1	1	1	1
$N_c(3)$	32	135	541	3,391	13,571
$N_c(10)$	108	455	1843	11,551	92,467
(II)	TMSD, $\alpha_b = 1$				
N_x	5	10	20	50	100
α_b	1	1	1	1	1
$N_c(3)$	18	72	289	1,811	7,244
$N_c(10)$	56	233	940	5,891	23,572
(III)	TMSD, $\alpha_b = \alpha_{bopt}$				
N_x	5	10	20	50	100
α_{bopt}	1.26	1.53	1.73	1.88	1.94
$N_c(3)$	8	16	33	82	165
$N_c(10)$	21	43	87	219	438

Table II: Maximum acceleration factor α_{amax} depending on the division number N_x .

N_x	5	6	7	8	9	10	...	49	50
α_{amax}	1.20	1.04	1.14	1.02	1.07	1.01	...	1.00	1.00

numerical stability analysis.

We can formally develop the discussion into the numerical calculation of the 2D Laplace equation, $\nabla_{(2)}^2 u = 0$. This paper assesses PDEs with higher-order accuracy. Therefore, let us represent the FDE more generally as follows.

$$u_i^{n+1} = u_i^n + \Delta t \left[P''_{m,x} (u_i^n) \right] \quad (7)$$

In $P''_{m,x}(u_i^n)$, m expresses the order of the algebraic polynomial interpolation, and in the high-order FDE, not necessarily the centered space FDE, $m = 3, 5$, etc., are also allowed (the meaning is described later). In the case of the quadratic interpolation $m = 2$, the following FTCS scheme is obtained over equidistant grid points.

$$P''_{2,x}(u_i^n) = \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{(\Delta x)^2} \quad (8)$$

In the 2D domain, Eq. (8) changes to the following.

$$u_{i,j}^{n+1} = u_{i,j}^n + \Delta t \left[P''_{2,x}(u_{i,j}^n) + P''_{2,y}(u_{i,j}^n) \right] \quad (9)$$

$$P''_{2,x}(u_{i,j}^n) = \frac{u_{i-1,j}^n - 2u_{i,j}^n + u_{i+1,j}^n}{(\Delta x)^2} \quad (9-1)$$

$$P''_{2,y}(u_{i,j}^n) = \frac{u_{i,j-1}^n - 2u_{i,j}^n + u_{i,j+1}^n}{(\Delta y)^2} \quad (9-2)$$

Equation (4) also changes to the following.

$$\Delta t = \alpha_a \Delta t_c, \Delta t_c = 1 / \left[\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2} \right], 0 < \alpha_a \leq 1 (= \alpha_{amax}) \quad (10)$$

When $\Delta t = \Delta t_c$ is adopted, Eq. (9) is reduced to the scheme of the following 2D Jacobi iteration method [Ref. 2, pp. 159-162].

$$u_i^{n+1} = \frac{1}{4} (u_{i-1,j}^n + u_{i+1,j}^n + u_{i,j-1}^n + u_{i,j+1}^n) \quad (11)$$

It is easy to extend the above definitions to 3D FDE. Equations (6) and (11) guarantee the fastest convergence based on the premise of simultaneous displacement of variables in the calculation process. Let us change Eq. (3) as follows.

$$u_i^{n+1} = u_i^n + \Delta t \left(\frac{u_{i-1}^{n+1} - 2u_i^n + u_{i+1}^n}{(\Delta x)^2} \right) \quad (12)$$

This is defined the time marching successive displacement (TMSD) scheme [5]. The definition of the TMSD method is inseparably close to the definition of the FTCS scheme. The concept of the TMSD should be applied to parabolic and elliptic PDEs in general, and it will be able to quickly grasp the convergence value. In this scheme, Eq. (4) is changed as follows.

$$\Delta t = \alpha_b \Delta t_c, \Delta t_c = 1 / \frac{2}{(\Delta x)^2}, 0 < \alpha_b < 2 (= \alpha_{bmax}) \quad (13)$$

As $\Delta t = \Delta t_c = (\Delta x)^2/2$, it becomes:

$$u_i^{n+1} = \frac{1}{2} (u_{i-1}^{n+1} + u_{i+1}^n) \quad (14)$$

This is the 1D Gauss-Seidel iteration method [Ref. 2, pp. 162-164]. The calculation result when $N_x = 5$ is shown in Figure 2(a). In the TMSD calculation, since only the convergence value is concerned, the initial value between points A and B is defined as zero. The convergence characteristics are shown in Table I (II), including the cases of other divisions. The Gauss-Seidel iteration method has meaning as an iteration calculation method of elliptic PDE, but it is a special case of Eq. (12). To execute the calculation of Eq. (12), it is sufficient to successively change the variable in the FTCS calculation. Let us use $\Delta t = \alpha_b \Delta t_c = \alpha_b (\Delta x)^2/2$ in Eq. (12), then we obtain the following results.

$$u_i^{n+1} = u_i^n + \frac{\alpha_b}{2} (u_{i-1}^{n+1} - 2u_i^n + u_{i+1}^n) = (1 - \alpha_b) u_i^n + \frac{\alpha_b}{2} (u_{i-1}^{n+1} + u_{i+1}^n) \quad (15)$$

Replacing the acceleration factor of the TMSD, α_b , with the relaxation parameter, ω , it becomes the expression of the successive over-relaxation (SOR) method [Ref. 2, pp. 166-167]. Therefore, α_b and ω are completely equivalent. The equivalence of the TMSD calculation and the SOR calculation is limited to calculations based on the second-order accurate difference, which does not hold in the calculation with a higher-order difference, as described in detail later. In the case of the 2D Laplace equation, the optimum relaxation parameter ω_{opt} that

ensures the fastest convergence is limited to all of the Dirichlet conditions and $\Delta x = \Delta y$, as represented by the following equation [Ref. 2, pp. 166-167 and Ref. 5].

$$\omega_{opt} (= \alpha_{bopt}) = \frac{2 - 2\sqrt{1-a}}{a} \quad (16)$$

$$a = \left\{ \frac{1}{2} \left[\cos\left(\frac{\pi}{N_x}\right) + \cos\left(\frac{\pi}{N_y}\right) \right] \right\}^2 \quad (16-1)$$

Let us define α_{bopt} as the optimum acceleration factor (OAF). In Eq. (16-1), as $N_x, N_y \rightarrow \infty$, it becomes $a \rightarrow 1$, then $\alpha_{bopt} \rightarrow 2$. Regarding Eq. (16-1), there is an important fact confirmed by the numerical stability analysis. In the case of the 1D Laplace equation, when Eq. (16-1) is changed to $a = \cos^2(\pi/N_x)$, Eq. 16 gives α_{bopt} . In Eq. (16-1), if it is $N_x = N_y$, it becomes $a = \cos^2(\pi/N_x)$. Also, in a 3D problem, it has been confirmed that the following expression holds.

$$a = \left\{ \frac{1}{3} \left[\cos\left(\frac{\pi}{N_x}\right) + \cos\left(\frac{\pi}{N_y}\right) + \cos\left(\frac{\pi}{N_z}\right) \right] \right\}^2 \quad (16-2)$$

Therefore, even when the domain is a cube and the division number is $N_x = N_y = N_z$, it also reduces to a one-dimensional expression. The author confirmed that this relation holds in a 4D problem by numerical calculation. From this, the general form of a in Eq. (16) is expressed by the following equation.

$$a = \left[\frac{1}{n} \sum_{i=1}^n \cos\left(\frac{\pi}{N_i}\right) \right]^2 \quad (16-3)$$

where n is the dimension of the domain.

Figure 2: Convergence process of the TMSD calculation. (a) $N_x = 5, \alpha_b = 1.0, N_C(3)=18, N_C(10)=56$, (b) $N_x = 5, \alpha_b = \alpha_{opt} = 1.26, N_C(3)=8, N_C(10)=21$, (c) $N_x = 5, \alpha_b = 1.53, N_C(3)=7, N_C(10)=33$, and (d) $N_x = 20, \alpha_b = 1.83, N_C(3)=36, N_C(10)=121$, cf. $\alpha_b = \alpha_{opt} = 1.729, N_C(3)=33, N_C(10)=108$.

Even if $N_x \neq N_y$ in a 2D problem, it is convenient if α_{opt} can be defined by reducing to an equivalent 1D calculation. Let us confirm this with Eq. (16-1). It holds that $\cos \theta = 1 - \theta^2/2 + \theta^4/4! + \dots$. When θ is small, the third and subsequent terms on the right side are ignored. Then Eq. (16-1) is expressed as follows.

$$a = \left\{ \frac{1}{2} \left[2 - \frac{1}{2} \left(\left(\frac{\pi}{N_x} \right)^2 + \left(\frac{\pi}{N_y} \right)^2 \right) \right] \right\}^2 \quad (16-4)$$

This is equivalent to the following equation.

$$a = \left\{ \frac{1}{2} \left[2 - \left(\frac{\pi}{N} \right)^2 \right] \right\}^2 \approx \cos^2 \frac{\pi}{N} \quad (16-5)$$

Therefore, the following equation holds.

$$\frac{2}{N^2} = \frac{1}{N_x^2} + \frac{1}{N_y^2} \quad (16-6)$$

N^2 is the harmonic mean of N_x^2 and N_y^2 . When $N_x = 10$ and $N_y = 20$, $N = 12.649 \dots$, then $a = \cos^2(\pi/N)$ gives the result, $\alpha_{bopt} = 1.6053 \dots$. On the other hand, Eq. (16) and (16-1) give the result $\alpha_{bopt} = 1.6056 \dots$. The OAF in multidimensional TMSD calculation leads to the equivalent OAF of the 1D TMSD calculation. As will be described later, the concept of the OAF is also effective when using higher-order differences in the TMSD calculation. There, the theoretical solution is unknown and is obtained by numerical calculation. Then it is convenient if it is possible to use the OAF of the 1D problem for the OAF in the 2D and 3D problems. Equation (16-1) is the theoretical solution in the case of a rectangular cross-section, but also in the case of an arbitrary section, it can be applied accordingly by adopting the maximum value for the division number of each N_x and N_y .

As in the case of the SOR method, let us define $0 < \alpha_b < 1$ as under-relaxation and $1 < \alpha_b < 2 (= \alpha_{bmax})$ as over-relaxation. There is an OAF, α_{bopt} , that guarantees the fastest convergence in the range of over-relaxation. It has been confirmed that the following holds [5].

(i) The convergence process at each calculation point progresses while affecting each other, then the state of convergence progress can be confirmed by specifying one calculation point near the center of the domain.

(ii) In the process of convergence, there are two cases where the calculated sequence is (I) converging monotonously and (II) converging while oscillating.

(iii) At that time, the OAF, $\alpha_{b_{opt}}$, is a limiting pattern of case (I), at which monotonicity is maintained.

Therefore, the range of the over-relaxation is divided into (I) $1 < \alpha_b \leq \alpha_{b_{opt}}$ (the range of monotonous convergence) and (II) $\alpha_{b_{opt}} < \alpha_b < 2$ (the range of oscillation convergence). Suppose that it is a monotonous increase in (I) the range of monotonous convergence. If it is assumed that $\alpha_b = \alpha_{b_{opt}} + \epsilon$, the calculated sequence monotonously increases to the maximum value exceeding the convergence value and follows the process of converging with oscillation. At this time, the oscillation amplitude decreases monotonously. If the convergence range $\pm\epsilon$ is designated beforehand, there are acceleration factors arriving at this range with the smallest calculation steps. This is the allowable oscillation OAF, expressed as $\alpha'_{b_{opt}}$. As the convergence range $\pm\epsilon \rightarrow 0$, it becomes $\alpha'_{b_{opt}} \rightarrow \alpha_{b_{opt}}$.

Figure 2(b) shows the result of the TMSD calculation using the OAF of $N_x = 5$, $\alpha_{b_{opt}} = 1.26$. It becomes $N_C(3)=8$, $N_C(10)=21$; the convergence is faster than in Figure 2(a). Table I (III) shows the calculation results including cases of other divisions. As the number of divisions increases, the advantage of using $\alpha_{b_{opt}}$ increases. In both Table I (I) (FTCS) and Table I (II) (TMSD), the convergence speed will be expressed as $C_s = cN_x^b$, $b = 2$. c is a proportional constant inherent to each problem. This relationship always holds if the acceleration factor is a fixed value. However, in the method that adopts $\alpha_b = \alpha_{b_{opt}}$, depending on N_x , it becomes $C_s = cN_x^b$, $b = 1$. It is ideal to adopt $\alpha_b = \alpha_{b_{opt}}$ whenever possible.

Figure 2(c) shows the calculation result of the TMSD with $\alpha_b = 1.53$. Once the calculated sequence exceeds the convergence value, the oscillation is decreases. If the convergence range is specified as 0.8 to 1.2, it reaches the specified range in three steps, so $\alpha_b = 1.53$ is one allowable

oscillation solution of the acceleration factor under this condition. However, $N_c(3)=8$ and $N_c(10)=34$ are larger than those in Figure 2(b), where $\alpha_b = \alpha_{bopt} = 1.26$. In this case, the maximum amplitude occurs at $x = 0.2(i = 1)$, but the definition of the convergence-confirmation grid point defined previously is not changed. The reason is that even if the convergence check is executed at all of the calculation points, there is almost no change in the convergence index evaluation. For example, Figure 2(d) shows the case of $N_x = 20$ and $\alpha_b = 1.83$. The result is $N_c(3)=36$, $N_c(10)=121$. At this time, if we check all of the calculation points, it becomes $N_c(3)=40$, $N_c(10)=129$. The difference between a one-point confirmation and an all-point confirmation will not be a problem. If we adopt $\alpha_b = \alpha_{bopt} = 1.729$, it becomes $N_c(3)=33$, $N_c(10)=108$. From this, it is understood that even if the approximate value is adopted for the acceleration factor, it does not become a problem.

Currently we deal with the ideal cases where we can use the theoretical solution of the OAF, but strictly the application of the theoretical solution is restricted to the regular domain. Furthermore, there is no known theoretical solution in the higher-order difference FDE. In relation to this, Fukuchi [5] proposed an algorithm to obtain the OAF by numerical calculation and confirmed the agreement with the theoretical solution. In that case, by adopting the “bisection method”, it was possible to numerically obtain a strict OAF with the specified number of digits. However, it is cumbersome to set a full-fledged routine. The precision of two decimal places is sufficient. Simple calculation by trial is performed as follows. To obtain an allowable oscillation OAF, we change the previously defined convergence definition $(u_{ic}^c - u_{ic}^n)/u_{ic}^c < 10^{-m}$ to $|u_{ic}^c - u_{ic}^n|/u_{ic}^c < 10^{-m}$. That is, under the condition $|u_{ic}^c - u_{ic}^n|/u_{ic}^c < 10^{-m}$, α'_{bopt} is determined, which gives the minimum number of convergence times. To clearly confirm the theoretical features of convergence, it is necessary to increase the division number to some extent. If we adopt $N_x = 50$, the result is as follows.

Theoretical OAF: $\alpha_{bopt} = 1.881 \dots$, $N_C(3) = 82, N_C(10) = 219$. Allowable oscillation OAF (i) $m = 3$; $\alpha'_{bopt} = 1.891$, $N_C(3) = 62$, (ii) $m = 10$; $\alpha'_{bopt} = 1.883, N_C(10) = 204$. As m increases, the allowable oscillation OAF converges to the theoretical solution, $\alpha_{opt} = 1.881 \dots$. The allowable oscillation OAF can be easily obtained by several trials.

Let us return to the definition of Eq. (7), $u_i^{n+1} = u_i^n + \Delta t [P''_{2,x}(u_i^n)]$. In the FTCS scheme, $P''_{2,x}(u_i^n)$ is expressed by the second-order accurate scheme of Eq. (8). The same calculation result is obtained by a second-order polynomial interpolation. Let the quadratic equation passing through three points $P_1(x_1, u_1), P_2(x_2, u_2)$, and $P_3(x_3, u_3)$ be $u = ax^2 + bx + c$. Each coefficient is determined and $u'' = 2a$. This corresponds as follows.

$$(x_1, y_1) \leftrightarrow (x_{i-1}, u_{i-1}), (x_2, y_2) \leftrightarrow (x_i, u_i), (x_3, y_3) \leftrightarrow (x_{i+1}, u_{i+1})$$

Since $u'' = 2a$ is different in the calculation points, it is expressed as $u''_i = 2a_i$. Assuming $P''_{2,x}(u_i^n) = 2a_i$, the result is consistent with the calculation using the FTCS scheme. This result is generalized in cases of unequally spaced grid points. This method is also effective in the TMSD scheme. This conclusion is obtained as a result of imparting the pseudo-acceleration term to an elliptic PDE. The above theory evolves into the derivation method of higher-order difference scheme. The polynomial interpolation is not limited to an algebraic polynomial. Any interpolation equation in which the derivative is defined could be used.

Let us refer to the method of directly obtaining the convergence value. In Figure 1, the following relationship holds at 4 calculation points, $i = 1, 2, 3, 4$.

$$(-u_{i-1} + 2u_i - u_{i+1}) = 0, \quad i = 1, 2, 3, 4 \quad (17)$$

Then the following simultaneous equations are obtained.

$$\begin{matrix} & \mathbf{M} & & \mathbf{u} & & \mathbf{f} \\ \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} & & \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} & = & \begin{bmatrix} u_0 \\ 0 \\ 0 \\ u_5 \end{bmatrix} \end{matrix} \quad (18)$$

Depending on the Dirichlet boundary condition, $u_0 = u_5 = 1$ and the solution is $u_i = 1$, $i = 1,2,3,4$. If the number of calculation points is small, the solution can be found instantly using the Gauss elimination method (GEM). However, as the number of calculation points increases, the computation time increases substantially. Equation (18) is a tridiagonal matrix, and when using the tridiagonal matrix algorithm (TDMA) [Ref. 13, pp. 50-54], the calculation time is shortened. In this paper, the expression for using the TDMA is as follows.

Table III: Convergence index of each calculation method; CT(s): actual calculation time.

TMSD-OPT				GEM		TDMA	
Nx	α_{bopt}	$N_c(10)$	CT(s)	Nx	CT(s)	Nx	CT(s)
100	1.939	438	0.023	100	0.155	10000	0.031
200	1.969	875	0.094	200	1.243	20000	0.063
400	1.984	1751	0.375	400	10.089	40000	0.113

$$\begin{matrix} & \mathbf{B} & & \mathbf{u} & & \mathbf{f} \\ \begin{bmatrix} x & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & x \end{bmatrix} & * & \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} & = & \begin{bmatrix} u_0 \\ 0 \\ 0 \\ u_5 \end{bmatrix} \end{matrix} \quad (19)$$

Matrix **B** is obtained by placing the diagonal elements of matrix **M** in the center column of matrix **B**. x is undefined and irrelevant to the calculation. Using the TDMA, the calculation is completed by the operation of matrix **B**, and the storage capacity is saved.

When the number of elements is small, there is little difference between the computation time of Eq. (18) using the GEM and that of Eq. (19) using the TDMA; however, as the number of elements increases, a decisive difference occurs in the computation time. Table III shows the computation time CT(s) using the author's PC. Although there are some shakes in these computation times, they are effective as reference values. The calculation results using the TMSD with the OAF (TMSD-OPT) are also shown regarding $N_x = 100, 200, \text{ and } 400$. The convergence time with $N_c(10)$, $C_t = cN_x^b$, and $b = 2$ holds. In the GEM calculation, the evaluation of the calculation time is $C_t = cN_x^b, b = 3$, which agrees with our knowledge. However, in the TDMA calculation, the calculation time is 0.5×10^{-3} seconds or less with $N_x < 1000$. The calculation results of $N_x = 10000, 20000, \text{ and } 40000$ using the TDMA are shown. The estimation of the calculation time is $C_t = cN_x^b, b = 1$ as expected. (Such an examination is unnecessary as far as it is limited to a 1D analysis, but it is not special in a 3D analysis that the number of elements exceeds 10000.) In the GEM calculations, we found that a large calculation time is required as the number of elements increases. In 2D and 3D problems, Eq. (18) becomes a diagonal dominant large sparse matrix. Then we have to consider an effective calculation method to replace the GEM. The conjugate gradient method (CGM) is one method of calculating this problem. The convergence speed using this method is almost equivalent to the TMSD [6]. However, since it is limited to the case where the coefficient matrix is symmetrical, it cannot be applied to higher-order differences. SOR is also commonly used. When the diagonal components are equal, let $\mathbf{M}'\mathbf{u} = \mathbf{f}'$ be the expression obtained by transforming $\mathbf{Mu} = \mathbf{f}$ such that the diagonal component is 1. To solve this via the iterative method, we transform $\mathbf{M}'\mathbf{u} = \mathbf{f}'$ to $\mathbf{u}^{n+1} = \mathbf{u}^n - \omega\mathbf{M}'\mathbf{u}^n + \omega\mathbf{f}'$. If this is iteratively calculated under successive displacement of the variable, it becomes an SOR calculation. In the calculation of $\mathbf{M}'\mathbf{u}^n$, if the non-zero components of $\mathbf{M}'$ are confirmed beforehand, it is possible to perform efficient calculations and the calculation speed is almost equivalent to that of the TMSD calculation.

Finally, we discuss the implicit method of the 1D problem. There are the Crank-Nicolson method and the full implicit method for the implicit solution. The β method can manage both [Ref. 2, pp. 66-68]. The β method is adopted here. The basic equation is expressed as follows.

$$u_i^{n+1} = u_i^n + \Delta t \left[\beta \frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{(\Delta x)^2} + (1-\beta) \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{(\Delta x)^2} \right] \quad (20)$$

Rearranging this, the following equation is obtained.

$$-\beta \frac{\Delta t}{(\Delta x)^2} u_{i-1}^{n+1} + (1+\beta \frac{2\Delta t}{(\Delta x)^2}) u_i^{n+1} - \beta \frac{\Delta t}{(\Delta x)^2} u_{i+1}^{n+1} = u_i^n + (1-\beta) \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{(\Delta x)^2} \quad (21)$$

The term for setting the boundary condition shifts to the right side. For each calculation step, simultaneous equations are solved using the TDMA. With reference to Eq. (19), the expression defining the whole of Eq. (20) is expressed as $\mathbf{B}_1 \mathbf{u}^{n+1} = \mathbf{f}^n$. $\mathbf{B}_1$ is produced at the beginning of the calculation and is used in common in each calculation step. When $\beta = 1$, it becomes the full implicit method, and for $\beta = 0.5$, the Crank-Nicolson method.

Figure 3: Time-dependent solution using the implicit scheme. (a) The Crank-Nicolson method, $\alpha_c = 1, \beta = 0.5$ and (b) the full implicit scheme, $\beta = 1$, $\alpha_c = 10$, $N_c(3)=7$, $N_c(10)=22$, cf. $\alpha_c = 10^{12}$, $N_c(3)=1$, $N_c(10)=1$.

Figure 3 (a) shows the calculation result using the Crank-Nicolson method. Let us express the acceleration factor as α_c ; here, $\alpha_c = 1$. The criteria time difference is the same as the

FTCS definition. The deviation between the numerical solution and the theoretical solution in Figure 1 has been resolved. The implicit method is also very useful when only the convergence value is a problem, then we use full implicit method, $\beta = 1$. In this method, no matter how much Δt is taken, the calculated sequence does not oscillate and uniformly converges. In other words, the Crank-Nicolson method is conditionally stable, whereas the full implicit method is unconditionally stable [Ref. 2, pp. 66-68]. Figure 3(b) shows the calculation result of the full implicit solution with $\alpha_c = 10$. There is a discrepancy between the numerical solution and the analytical solution, but this is not a problem because only the convergence value is concerned. If we use $\alpha_c = 10^{12}$, it becomes $N_c(3)=1$, $N_c(10)=1$. The full implicit scheme is expressed as follows with $\beta = 1$ in Eq. (21).

$$-\frac{\Delta t}{(\Delta x)^2} u_{i-1}^{n+1} + \left(1 + \frac{2\Delta t}{(\Delta x)^2} \right) u_i^{n+1} - \frac{\Delta t}{(\Delta y^*)^2} u_{i+1}^{n+1} = u_i^n \quad (22)$$

Dividing both sides of this expression by Δt and setting $\Delta t \rightarrow \infty$ results in the direct method as shown in Eq. (18). Therefore, the full implicit solution has significance as a semi-direct method in 1D problems.

We studied the case of the “regular” domain. The physical phenomenon is irrelevant to the coordinate system. When this is applied to a 1D analysis, the same result (with little difference) must be obtained using an arbitrary coordinate movement. Figure 4(a) shows the result obtained by shifting the coordinates

in Figure 1(a) to the left by $m_p \Delta x = 0.5 \Delta x$. It is defined as a regular element disposition [4]. The reason to investigate this problem is that it is necessary to consider regular element disposition for examining arbitrary domains in 2D and 3D problems. Although the calculation result is strange, it is the same as the phenomenon occurring in Figure 1(b), and the influence of a singular point in the initial condition setting appears. Here, only the convergence of the numerical calculation is a problem. The same problem is calculated using $\alpha_a = 0.5$ in Figure 4(b). In Figure 4(a), the numerical calculation usually

diverges using $\alpha_a = 1$. However, it does not diverge because we changed Eq. (7) as follows [5].

Figure 4: Numerical calculation over regular element disposition. (a) The FTCS scheme with $N_x = 5, \alpha_a = 1$, (b) the FTCS scheme with $N_x = 6, \alpha_a = 0.5$, (c) the FTCS scheme $N_x = 10, \alpha_a = 1$, and (d) the TMSD scheme with $\alpha_{bopt} = 1.26, N_c(3)=9, N_c(10)=24$.

$$u_i^{n+1} = u_i^n + \varphi_i \Delta t \left[P_{2,x}''(u_i^n) \right], \varphi_i = (l_{wi})^{P_w} \quad (23)$$

φ_i is the time interval adjustment factor and l_{wi} is the wall distance factor. Correction is necessary only for the boundary proximity part of $i = 1$ and $i = 4$. At $i = 1$, it is defined as $l_{w1} = l_1/\Delta x = 0.5$, and the same is true at $i = 4$. At this time, it is defined as $P_w = 1$, and otherwise it is $P_w = 0$. This technique

was used in IFDM-BPIM. Let us follow this with IFDM-APIM as well. In IFDM-APIM, the difference expression is generalized. The Shortley and Weller (SW) method is considered herein [14]. With reference to Figure 4(b), it is defined as follows.

$$P''_{2,x}(u_i^n) = \frac{2u_{i-1}^n}{l_1(l_1 + l_2)} - \frac{2u_i^n}{l_1 l_2} + \frac{2u_{i+1}^n}{l_2(l_1 + l_2)} \equiv A_{-1}u_{i-1} - A_0u_i + A_1u_{i+1} \quad (24)$$

As $l_1 = l_2 = \Delta x$, it results in a usual second-order accurate central-difference scheme and becomes $A_0 = 2/(\Delta x)^2$. The criteria time difference is $\Delta t_c = (\Delta x)^2/2 = 1/A_0$. As described previously, using Eq. (7), the calculation diverges when $\Delta t = (\Delta x)^2/2$ is adopted in Figure 4(a). At $i = 1$, $A_0 = 2/(0.5\Delta x \times \Delta x) = 4/(\Delta x)^2$. When adopting the reciprocal number for the criteria time difference, it becomes $\Delta t_c = (\Delta x)^2/4$ and $\alpha_a = 1$ gives a stable convergence. However, assuming that l_1 takes an arbitrary value in the range of $0 < l_1 < \Delta x$, a problem arises. If it is assumed to be $l_1 = 0.1\Delta x$, it becomes $\Delta t_c = (\Delta x)^2/20$, and we have to use a very small time interval. The first way to avoid this problem is to adopt $\Delta x \leq l_1, l_2 < 2\Delta x$. This process is adopted in the ghost fluid method^{9,10}. As a second method, it is possible to avoid the problem by adjusting the time difference in this part to $l_{w1}\Delta t (l_{w1} = l_1/\Delta x)$. Equation (23) means that the latter method was adopted. To eliminate the influence of the initial condition singularity in Figure 4(a), we adopt $N_x = 10, \alpha_a = 1$ as shown in Figure 4(c). The numerical solution and the theoretical solution are in good agreement. Using $\alpha_a = 0.8$ in Figure 4(c), in the movement $m_v\Delta x (0 < m_v < 1)$ of the coordinate, regardless of how m_v is specified, the numerical solution and the theoretical solution seem to be perfectly matched only by changing the calculation points. The definition of Eq. (23) seems to be inevitable that it is not simply a way of suppressing divergence but must be specified like this. As previously mentioned, if we choose the grid disposition with the regulation $\Delta x \leq l_1, l_2 < 2\Delta x$ or adopt the Crank-Nicolson method, this problem can be completely avoided. In the TMSD calculation, only the convergence value is a problem, so the above method is

completely effective. Figure 4(d) is a calculation example corresponding to Figure 2(b). The OAF, $\alpha_{bopt} = 1.26$, of $N_x = 5$ can be applied accordingly. (Calculating using a numerical stability analysis will result in different results, but not very different.) In this case, the calculation using Eq. (23) is equivalent to the calculation by SOR. This can be easily confirmed by numerical calculations.

Derivation of High-Order Finite Differences Based on Lagrange Interpolation Function

Conventionally, the difference scheme was derived via Taylor series expansion [15]. Let us assess the second-order accurate scheme and the SW scheme. The following two formulas are obtained using the Taylor series expansion.

$$f(x + \Delta x) = f(x) + (\Delta x) \frac{\partial f}{\partial x} + \frac{(\Delta x)^2}{2!} \frac{\partial^2 f}{\partial x^2} + \frac{(\Delta x)^3}{3!} \frac{\partial^3 f}{\partial x^3} + \frac{(\Delta x)^4}{4!} \frac{\partial^4 f}{\partial x^4} + \dots \quad (25)$$

$$f(x - \Delta x) = f(x) - (\Delta x) \frac{\partial f}{\partial x} + \frac{(\Delta x)^2}{2!} \frac{\partial^2 f}{\partial x^2} - \frac{(\Delta x)^3}{3!} \frac{\partial^3 f}{\partial x^3} + \frac{(\Delta x)^4}{4!} \frac{\partial^4 f}{\partial x^4} + \dots \quad (26)$$

The following expression is derived from both equations.

$$\frac{\partial^2 f}{\partial x^2} = \frac{f(x - \Delta x) - 2f(x) + f(x + \Delta x)}{(\Delta x)^2} - \frac{(\Delta x)^2}{12} \frac{\partial^4 f}{\partial x^4} + \dots \quad (27)$$

The second term on the right side of the above equation is the highest-order term of the difference error, which indicates that the accuracy is second order. Next, let us derive the SW scheme, Eq. (24). The following equations hold.

$$f(x + l_2) = f(x) + l_2 \frac{\partial f}{\partial x} + \frac{l_2^2}{2!} \frac{\partial^2 f}{\partial x^2} + \frac{l_2^3}{3!} \frac{\partial^3 f}{\partial x^3} + \frac{l_2^4}{4!} \frac{\partial^4 f}{\partial x^4} + \dots \quad (28)$$

$$f(x - l_1) = f(x) - l_1 \frac{\partial f}{\partial x} + \frac{l_1^2}{2!} \frac{\partial^2 f}{\partial x^2} - \frac{l_1^3}{3!} \frac{\partial^3 f}{\partial x^3} + \frac{l_1^4}{4!} \frac{\partial^4 f}{\partial x^4} + \dots \quad (29)$$

By multiplying Eq. (28) by l_1 and Eq. (29) by l_2 and rearranging them so that $\partial f / \partial x$ term is zero, the following equation is obtained.

$$\frac{\partial^2 f}{\partial x^2} = \frac{2f(x-l_1)}{l_1(l_1+l_2)} - \frac{2f(x)}{l_1 l_2} + \frac{2f(x+l_2)}{l_2(l_1+l_2)} + \frac{(l_1-l_2)}{3} \frac{\partial^3 f}{\partial x^3} - \frac{(l_1^2-l_1 l_2+l_2^2)}{12} \frac{\partial^4 f}{\partial x^4} + \dots \quad (30)$$

This equation reduces to Eq. (27) under the condition of $l_1 = l_2 = \Delta x$, but when $l_1 \neq l_2$, the fourth term on the right-hand side remains. If it is defined that the highest-order error term shows the accuracy of the difference, this is the first-order accuracy. Equation (27) has been conventionally referred to as a second-order accurate difference, which is a special form of Eq. (30). Both are inseparable from quadratic interpolation curves as described below. For this reason, we use the term "second-order difference" to include both. Although the above method has the advantage that it is possible to evaluate the accuracy of the difference, it becomes difficult as it gradually shifts to a higher-order difference and handling irregularly spaced grid points. At this time, from the perspective that it is sufficient to calculate only the difference scheme, more efficient derivation methods are necessary. Consider the following Lagrange interpolation function passing through three points.

$$P_2(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} u_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} u_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} u_3 \quad (31)$$

The notation of $P_2(x)$ indicates that this is a quadratic expression. It is obvious that this is equivalent to a quadratic equation $u = ax^2 + bx + c$ going through the three points $P_1(x_1, u_1)$, $P_2(x_2, u_2)$, and $P_3(x_3, u_3)$. The first term on the right side is u_1 at x_1 and zero at x_2 and x_3 . The second and third terms can be explained in the same way. Let us express each term as follows.

$$P_{21}(x) = (a_{11}x^2 + a_{12}x + a_{13})u_1 \quad (31-1)$$

$$P_{22}(x) = (a_{21}x^2 + a_{22}x + a_{23})u_2 \quad (31-2)$$

$$P_{23}(x) = (a_{31}x^2 + a_{32}x + a_{33})u_3 \quad (31-3)$$

Each coefficient $a_{i,j}$ is defined by solving the following equation.

$$\begin{matrix} & \mathbf{X} & & \mathbf{A} & & \mathbf{I} \\ \left[\begin{matrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{matrix} \right] & \left[\begin{matrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{matrix} \right] & = & \left[\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \right] \end{matrix} \quad (32)$$

$\mathbf{X}$ is the Vandermonde matrix and $\mathbf{A} = \mathbf{X}^{-1}$ [Ref. 6]. Let us define $\mathbf{A}$ as a difference coefficient matrix. The following relation holds with respect to differentiation.

$$\begin{aligned} \frac{\partial P_2(x)}{\partial x} (\equiv P_{2,x}'') &= \frac{\partial P_{21}(x)}{\partial x} + \frac{\partial P_{22}(x)}{\partial x} + \frac{\partial P_{23}(x)}{\partial x} \\ &= (2a_{11}x + a_{12})u_1 + (2a_{21}x + a_{22})u_2 \\ &\quad + (2a_{31}x + a_{32})u_3 \end{aligned} \quad (33)$$

$$\begin{aligned} \frac{\partial^2 P_2(x)}{\partial x^2} (\equiv P_{2,x}''') &= \frac{\partial^2 P_{21}(x)}{\partial x^2} + \frac{\partial^2 P_{22}(x)}{\partial x^2} + \frac{\partial^2 P_{23}(x)}{\partial x^2} \\ &= 2a_{11}u_1 + 2a_{21}u_2 + 2a_{31}u_3 \end{aligned} \quad (34)$$

The same result can be obtained by arbitrarily setting the coordinate zero point, but in this paper, it is defined as $x_1 = 0$. The first derivative changes depending on the position. The second derivative is constant regardless of the position. This is a characteristic of a quadratic curve. Equation (34) is equivalent to the SW scheme. Figure 5 shows the calculation result of the Lagrange interpolation function calculated using the difference coefficient matrix, defined as $u_1 = 1$, $u_2 = 2$, and $u_3 = 1.5$. The above method is expanded to the interpolation expression of an arbitrary order, and instantaneously the derivatives, of which the orders are equal to or less than the order of interpolation

equation, are derived at the designated position. This process can be determined instantaneously using a calculator. As can be seen from the derivation process, this method is applied irrespective of the calculation point disposition (equidistant or not). Using Lagrange's polynomial interpolation directly, Figure 5 can be drawn, but it is difficult to define the difference schemes. Numerically, the same results can be obtained from n-th order polynomials; however, it is impossible to define the difference schemes. The advantage of obtaining $\mathbf{A} = \mathbf{X}^{-1}$ is that the difference schemes are instantaneously determined, and the explicit/implicit methods become possible using high-order difference schemes. As previously mentioned, the method of deriving the difference scheme using the Taylor series expansion can evaluate the accuracy, but it lacks maneuverability. Although it is not possible to directly evaluate the accuracy of the difference scheme in the above method, it is possible to evaluate the accuracy by comparing the results of numerical calculations with some analytical solutions [6]. The above method was published as the author's original method; however, Klein and Berrut [16] previously suggested a method to derive finite difference formulas over an equidistant grid interval using Lagrange's polynomial interpolants. Although there seems to be a difference in the algorithms between the author's method and their method, it was confirmed that their results of finite difference formulas can be obtained from the author's method as well.

Figure 5: The Lagrange interpolation function.

Table IV shows the results of the central difference of $P''_{m,x}$, $m = 2, 4, \dots, 10$ at equidistant intervals ($\Delta x = 1$). In the central difference in Table IV, the center point number is expressed as zero.

Table IV: Coefficients of the central difference with equally spaced intervals $\Delta x = 1$.

Order	Position	-1	0	1	2	3	4	5
2		1	-2	1				
4		1.333333	-2.5	1.333333	-0.08333			
6		1.5	-2.72222	1.5	-0.15	0.011111		
8	Symmetry	1.6	-2.84722	1.6	-0.2	0.025397	-0.00179	
10		1.666667	-2.92722	1.666667	-0.2381	0.039683	-0.00496	0.000317

Table V: Coefficients of the even number order difference with equally spaced intervals $\Delta_x = 1$ (the values in this table are shown in finite decimal numbers, but they are instantaneously and exactly given in the process of numerical calculation analysis).

P2	0	1	2								
1	1.0000	-2.0000	1.0000								
P4	0	1	2	3	4						
1	0.9167	-1.6667	0.5000	0.3333	-0.0833						
2	-0.0833	1.3333	-2.5000	1.3333	-0.0833						
3	-0.0833	0.3333	0.5000	-1.6667	0.9167						
P6	0	1	2	3	4	5	6				
1	0.7611	-0.8167	-1.4167	2.6111	-1.5833	0.5167	-0.0722				
2	-0.0722	1.2667	-2.3333	1.1111	0.0833	-0.0667	0.0111				
3	0.0111	-0.1500	1.5000	-2.7222	1.5000	-0.1500	0.0111				
4	0.0111	-0.0667	0.0833	1.1111	-2.3333	1.2667	-0.0722				
5	-0.0722	0.5167	-1.5833	2.6111	-1.4167	-0.8167	0.7611				
P8	0	1	2	3	4	5	6	7	8		
1	0.6482	0.0254	-4.1500	7.6500	-7.3472	4.7000	-1.9500	0.4754	-0.0518		
2	-0.0518	1.1143	-1.8389	0.2000	1.1250	-0.8222	0.3500	-0.0857	0.0093		
3	0.0093	-0.1357	1.4500	-2.6222	1.3750	-0.0500	-0.0389	0.0143	-0.0018		
4	-0.0018	0.0254	-0.2000	1.6000	-2.8472	1.6000	-0.2000	0.0254	-0.0018		
5	-0.0018	0.0143	-0.0389	-0.0500	1.3750	-2.6222	1.4500	-0.1357	0.0093		
6	0.0093	-0.0857	0.3500	-0.8222	1.1250	0.2000	-1.8389	1.1143	-0.0518		
7	-0.0518	0.4754	-1.9500	4.7000	-7.3472	7.6500	-4.1500	0.0254	0.6482		
P10	0	1	2	3	4	5	6	7	8	9	10
1	0.5658	0.8054	-7.4607	15.9476	-20.939	19.8950	-13.683	6.6492	-2.1679	0.4260	-0.0382
2	-0.0382	0.9857	-1.2942	-1.1619	3.3500	-3.3022	2.2583	-1.0857	0.3504	-0.0683	0.0061
3	0.0061	-0.1050	1.3196	-2.2960	0.8417	0.5450	-0.4972	0.2548	-0.0839	0.0165	-0.0015
4	-0.0015	0.0222	-0.1857	1.5619	-2.7806	1.5200	-0.1333	-0.0127	0.0125	-0.0032	0.0003
5	0.0003	-0.0050	0.0397	-0.2381	1.6667	-2.9272	1.6667	-0.2381	0.0397	-0.0050	0.0003
6	0.0003	-0.0032	0.0125	-0.0127	-0.1333	1.5200	-2.7806	1.5619	-0.1857	0.0222	-0.0015
7	-0.0015	0.0165	-0.0839	0.2548	-0.4972	0.5450	0.8417	-2.2960	1.3196	-0.1050	0.0061
8	0.0061	-0.0683	0.3504	-1.0857	2.2583	-3.3022	3.3500	-1.1619	-1.2942	0.9857	-0.0382
9	-0.0382	0.4260	-2.1679	6.6492	-13.683	19.8950	-20.939	15.9476	-7.4607	0.8054	0.5658

As previously mentioned, in this paper, a local coordinate system with $x_0 = 0$ at the left end point is used. For this reason, the second-order (accurate) difference is expressed as shown in Table V-P2. For second-order differences, the interpolation function is quadratic and the evaluation of $P_{2,x}''$ is only the center point, but in fourth-order differences, the evaluation of $P_{4,x}''$ there are three points including the central difference, except for two end points, and it changes depending on the position. They are expressed as:

$$P_4''(x_1) = A_{1,0}u_0 + A_{1,1}u_1 + A_{1,2}u_2 + A_{1,3}u_3 + A_{1,4}u_4 \quad (35-1)$$

$$P_4''(x_2) = A_{2,0}u_0 + A_{2,1}u_1 + A_{2,2}u_2 + A_{2,3}u_3 + A_{2,4}u_4 \quad (35-2)$$

$$P_4''(x_3) = A_{3,0}u_0 + A_{3,1}u_1 + A_{3,2}u_2 + A_{3,3}u_3 + A_{3,4}u_4 \quad (35-3)$$

Including the other orders, the general expression is as follows.

$$P_m''(x_i) = \sum_{k=0}^m A_{i,k} u_k, i = 1, 2, \dots, m-1 \quad (36)$$

Now it is assumed that $u_k = \text{constant}$ in Eq. (36). Then $P_m''(x_i) = 0$, meaning $\sum_k A_{i,k} = 0$. Table V shows the calculation results. In the central difference, in Eq. (36), $i = m/2 (\equiv m_c)$ is the central term. Table V demonstrates that the maximum value of the absolute value of each coefficient $A_{i,k}$ is $|A_{m_c, m_c}|$. In second-order differences, the criteria time difference is set to $\Delta t_c = (\Delta x)^2/2$ in Eqs. (4) and (13). There may be instances that commonly use this standard. However, in this paper, it is used as follows. It is equivalent to define $\Delta t_c = (\Delta x)^2/2$, and in $P_2''(x_1) = A_{1,0}u_0 + A_{1,1}u_1 + A_{1,2}u_2$ to define $\Delta t_c = 1/|A_{1,1}|$. Therefore, the criteria time difference is generally defined as $\Delta t_c = 1/|A_{m_c, m_c}|$. The m-th order difference scheme is not limited to the central difference. The expression of Eq. (36) holds even if $m = 3, 5, \dots, 9$. Table VI shows the calculation results. For odd-order differences, in the case of equidistant grid points, there are two occurrences of the maximum value of the coefficient (usually one place in case of unequal spacing). In odd-order differences, the criteria time difference is determined in the same manner as in even-order differences. Table VI-P3 demonstrates the third-order difference retracted to the second-order difference. That is, at x_1 , $P_3''(x_1)$ is determined irrespective of u_3 .

$$\begin{aligned} P_3''(x_1) &= A_{1,0}u_0 + A_{1,1}u_1 + A_{1,2}u_2 + A_{1,3}u_3 \\ \rightarrow P_2''(x_1) &= A_{1,0}u_0 + A_{1,1}u_1 + A_{1,2}u_2 \end{aligned} \quad (37)$$

This is the case of equidistant grid points, not at unequally spaced grid points. The validity of this result can be easily confirmed by numerical calculations and is also derived from Lagrange interpolation function. Tables VI-P5, P7, and P9 demonstrate that the retraction occurred in the pair of central portions.

Table VI: Coefficients of the odd number order difference with equally spaced intervals $\Delta_x=1$.

P3	0	1	2	3						
1	1.0000	-2.0000	1.0000	0.0000						
2	0.0000	1.0000	-2.0000	1.0000						
P5	0	1	2	3	4	5				
1	0.8333	-1.2500	-0.3333	1.1667	-0.5000	0.0833				
2	-0.0833	1.3333	-2.5000	1.3333	-0.0833	0.0000				
3	0.0000	-0.0833	1.3333	-2.5000	1.3333	-0.0833				
4	0.0833	-0.5000	1.1667	-0.3333	-1.2500	0.8333				
P7	0	1	2	3	4	5	6	7		
1	0.7000	-0.3889	-2.7000	4.7500	-3.7222	1.8000	-0.5000	0.0611		
2	-0.0611	1.1889	-2.1000	0.7222	0.4722	-0.3000	0.0889	-0.0111		
3	0.0111	-0.1500	1.5000	-2.7222	1.5000	-0.1500	0.0111	0.0000		
4	0.0000	0.0111	-0.1500	1.5000	-2.7222	1.5000	-0.1500	0.0111		
5	-0.0111	0.0889	-0.3000	0.4722	0.7222	-2.1000	1.1889	-0.0611		
6	0.0611	-0.5000	1.8000	-3.7222	4.7500	-2.7000	-0.3889	0.7000		
P9	0	1	2	3	4	5	6	7	8	9
1	0.6040	0.4236	-5.7429	11.3667	-12.922	10.2750	-5.6667	2.0683	-0.4500	0.0442
2	-0.0442	1.0464	-1.5675	-0.4333	2.0750	-1.7722	0.9833	-0.3571	0.0772	-0.0075
3	0.0075	-0.1196	1.3857	-2.4722	1.1500	0.1750	-0.1889	0.0786	-0.0179	0.0018
4	-0.0018	0.0254	-0.2000	1.6000	-2.8472	1.6000	-0.2000	0.0254	-0.0018	0.0000
5	0.0000	-0.0018	0.0254	-0.2000	1.6000	-2.8472	1.6000	-0.2000	0.0254	-0.0018
6	0.0018	-0.0179	0.0786	-0.1889	0.1750	1.1500	-2.4722	1.3857	-0.1196	0.0075
7	-0.0075	0.0772	-0.3571	0.9833	-1.7722	2.0750	-0.4333	-1.5675	1.0464	-0.0442
8	0.0442	-0.4500	2.0683	-5.6667	10.2750	-12.922	11.3667	-5.7429	0.4236	0.6040

Figure 6(a) shows the result calculated by the forward time explicit method using the fifth-order difference scheme, $P_5''(x_i), i = 1, 2, 3, 4$, called FTEX(5). From Table VI-P5, it becomes $\Delta t_c = (\Delta x)^2/2.5$, and $\alpha_a = 1$ is used. The calculation accuracy with respect to time progression is equivalent to the case in Figure 1. Calculating this using the Crank-Nicolson implicit method improves the calculation accuracy even if the acceleration factor $\alpha_c = 1$. Figure 6(b) shows the result calculated with $\beta = 0.8$, which is highly accurate, but the rationale for specifying $\beta = 0.8$ is unclear. When adopting the acceleration factor $\alpha_c = 10^{12}$ in the full implicit method ($\beta = 1$), the calculation becomes equivalent to the direct method and converges in one set calculation. Similar results are obtained for the seventh-order difference and the ninth-order difference. The TMSD calculation can also be applied and convergence accelerates. An important fact arises here. For the second-order accurate difference, the TMSD calculation and the SOR calculation is completely equivalent. In this case, however, they deviate. Figure 6(c) shows the calculation result via the TMSD. The convergence indexes are $\alpha'_{bopt} = 1.59, N_C(10) = 21$. Figure 6(d) shows the calculation result by the SOR. The convergence indexes are $\omega'_{opt} = 1.11, N_C(10) = 42$. The TMSD

calculation converges faster than the SOR calculation. In the actual calculation, we do not use an odd-order difference alone (described in detail later). However, in the multidimensional problem and the higher-order difference numerical analysis, it is possible to improve the calculation accuracy by partially using the odd-order difference at the corner portion of the irregular domain.

Figure 6: Numerical calculations using one algebraic polynomial equation, the fifth order, and $N_x = 5$. (a) FTEX(5), $\alpha_a = 1$, (b) the Crank-Nicolson implicit method, $\alpha_c = 0.8$, (c) the TMSD calculation, $\alpha'_{opt} = 1.59$, $N_c(10) = 21$, and (d) the SOR calculation, $\omega'_{opt} = 1.11$, $N_c(10) = 42$.

Figures 7(a) and (b) show the results of calculating the case of Figure 1(c) using the fourth-order difference. Figure 7(a) adopts three fourth-order polynomials P_0, P_3 , and P_6 . This is a sparse setting of polynomials [6]. For example, the interpolation of P_0 is used for the calculation at each point of x_1, x_2 , and x_3 . Equations (35-1), (35-2), and (35-3) are used at each calculation point, respectively. In this setting, if the number of divisions is even, the polynomial setting is symmetrical, but if it is an odd number, symmetry cannot be always maintained. A total of 7 polynomials $P_0, P_1, \dots, P_6$ are used in Figure 7(b). This is a dense

setting of polynomials [6]. In this setting, the symmetry is maintained irrespective of the even number and odd number of divisions. This is called the standard fourth-order API scheme and defined as SAPI(4). The setting methods of polynomials at $N_x = 10$ are innumerable. It is also possible to set even-order and odd-order mixed polynomials and to set only odd-order polynomials. The corresponding numerical solution is obtained if the polynomials cross one another. However, if SAPI(m), $m = 2, 4, \dots, 10$, the symmetrical polynomial setting is uniquely determined only due to specifying the value of m . The highest-order polynomial considered in the paper is up to the tenth order. In Figure 7(b), in SAPI(m), $m = 2, 4, 6$, the calculation results seem to visually coincide strictly with analytical solutions. A slight deviation occurs at $m = 8$, and at $m = 10$, it is reduced to $N_x = m$ as described previously. Its time progressing solution is not very accurate. Figure 7(c) shows the result of the TMSD calculation with SAPI(4). The convergence indexes are $\alpha'_{bopt} = 1.63$ and $N_C(10)=44$. Also, in this case, the TMSD and SOR calculations are not equivalent. Figure 7(d) shows the result of the SOR calculation with SAPI(4); its convergence indexes are $\omega'_{bopt} = 1.61$ and $N_C(10)=47$. Figures 7(e) and (f) show SAPI(6). The TMSD calculation is superior to the “SUR” calculation; in particular, in the SUR calculation in Figure 7(f), divergence occurs unless it is set to under-relaxation ($\omega < 1$), and the convergence speed decelerates. When using the direct method, the symmetry of the matrix confirmed by the second-order difference is not retained. The calculation equation for $N_x = 6$, SAPI(4) is shown below.

Figure 7: Numerical calculation using the higher-order difference scheme, $N_x = 10$. (a) FTEX(4) sparse setting, (b) FTEX(4); densely setting, (c) TMSD(4); $\alpha'_{bopt} = 1.63$, $N_C(10)=44$, (d) SOR(4); SOR $\omega'_{bopt} = 1.61$, $N_C(10)=47$, (e) TMSD(6); $\alpha'_{opt} = 1.41$, $N_C(10)=137$, and (f) SOR(6) $\omega'_{opt} = 0.85$, $N_C(10)=408$.

$$\begin{bmatrix} 60 & -18 & -12 & 3 & 0 \\ -48 & 90 & -48 & 3 & 0 \\ 3 & -48 & 90 & -48 & 3 \\ 0 & 3 & -48 & 90 & -48 \\ 0 & 3 & -12 & -18 & 60 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} = \begin{bmatrix} 33 \\ -3 \\ 0 \\ -3 \\ 33 \end{bmatrix} \quad (38)$$

This equation is correct with reference to Table V-P4. When this is represented by matrix **B**, it is expressed as follows.

$$\begin{matrix} & \mathbf{B} & & \mathbf{u} & & \mathbf{f} \\ \begin{bmatrix} x & x & x & 60 & -18 & -12 & 3 \\ x & x & -48 & 90 & -48 & 3 & 0 \\ x & 3 & -48 & 90 & -48 & 3 & x \\ 0 & 3 & -48 & 90 & -48 & x & x \\ 3 & -12 & -18 & 60 & x & x & x \end{bmatrix} & * & \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} & = & \begin{bmatrix} 33 \\ -3 \\ 0 \\ -3 \\ 33 \end{bmatrix} \end{matrix} \quad (39)$$

M is 5×5 , **B** is 5×7 , and the latter is a larger matrix. When the number of calculation points is small, both calculations are instantaneously completed. However, if the number of calculation points is 100, **M** is 100×100 and **B** is 100×7 , and in the latter case, as we use the band diagonal matrix algorithm (BDMA) [Ref. 13, pp. 50-54], which includes the TDMA as a special case, the computation time is terminated at 7/3 of the calculation in the case of the second-order difference. When there are many calculation points, a decisive difference occurs in the computation time for both. Also, in the calculation of the implicit method, a 100×7 band matrix must be adopted. The bandwidth of SAPI(m) is generally $2m-1$ and the number of the centre position is m when m is an even number. The above shows the handling of higher-order differences in regular domains, but it can manage exactly the same in irregular domains. In the calculation of FTEx and the TMSD and in the calculation of both calculation end points over an irregular domain, time difference width adjustment is performed as shown in Eq. (23). In the implicit solution, such an adjustment is unnecessary.

Finally, we consider the problems of unequally spaced grid points that will be needed in the future. Although the discussion thus far can be applied to any unequally spaced grid point, singularities must be considered in the FDM. Singularities in the initial condition were examined in this paper, but singularities in the boundary shape may be a problem. The need to set the difference interval smaller for a specific area occurs in multidimensional problems. In preparation for such a situation, we will confirm the problem of unequally spaced grid points in

1D problems. Figure 8(a) is an example calculated using FTEX with $N_x = 10(\Delta x = 0.1)$ and $\Delta x = 0.1/5 = 0.02$ in the range $0.4 \leq x \leq 0.6$ using SAPI(6). The criteria time difference is defined from $\Delta x = 0.02$ and the central difference in SAPI(6), becoming $\Delta t_c = 1.47 \times 10^{-4}$, and as $\alpha_a = 0.8$, it becomes $\Delta t = 1.18 \times 10^{-4}$. Changing the subdivision range to $0.3 \leq x \leq 0.6$ does not cause any problems. The result is shown as $n = 160k$, $k = 1, 2, 3 \dots$. The upper limit of α_a where the calculation does not diverge is $\alpha_{amax} = 0.95$. The calculation can be done using all SAPI(m), $m = 2, 4, \dots, 10$, but α_{amax} changes individually. In this case, using the implicit Crank-Nicolson method (CNM), the range of the subdivision point is set to $0.3 \leq x \leq 0.6$, and the result is calculated by $\alpha_c = 16(\Delta t = 2.35 \times 10^{-3})$ as shown in Figure 8(b). Compared with Figure 8(a), the number of calculations is 1/20. Using $\alpha_c = 10^{12}$ in the full implicit method, the convergence value can be obtained by one set of calculations. Figure 8(c) shows the TMSD calculation result using SAPI(6); the convergence index is $\alpha'_{bopt} = 1.98$, NC(3)=1910. If we use $\alpha_b = 1.99$, the calculation diverges; therefore, it is $\alpha'_{bopt} \cong \alpha_{bmax}$. This is characteristic of the TMSD calculation using high-order differences. Figure 8(d) demonstrates the SUR calculation result using SAPI(6); convergence index is $\omega'_{opt} = 0.58$, $N_C(3)=2559$. In this case, if we use $\omega \geq 0.59$, the calculation diverges. The TMSD calculation is superior to the SUR calculation.

Figure 8: Numerical calculations over unequal grid points, SAPI(6). (a) FTEX; $\alpha_a = 0.8$, $N_c(3)=5771$, (b) FIM(6); $\alpha_c = 16$, $N_c(3)=283$, (c) TMSD(6); $\alpha'_{opt} = 1.98$, $N_c(3)=1910$, and (d) SUR(6) $\omega'_{opt} = 0.58$, $N_c(3)=2559$.

Figure 9: Numerical calculation with the Poisson equation using the BDMA. (I) $\nabla^2_{(1)}u = 0$, (II) $\nabla^2_{(1)}u = 6x$, (III) $\nabla^2_{(1)}u = 16\exp(4x)$, and (IV) $\nabla^2_{(1)}u = -64\pi^2 \sin(8\pi x)$.

Numerical Calculation Accuracy Regarding 1D Poisson Equations

The convergence of the 1D problem has been examined thus far, but here we assess the accuracy of the convergence value. Since the convergence process is not a problem, the numerical calculation is performed via the direct method with the BDMA. The Laplace equation $\nabla_{(1)}^2 u = 0$ has only a few variations, therefore, the Poisson equation $\nabla_{(1)}^2 u = f$ is used. All of the calculation methods studied in the Laplace equation are also effective in the Poisson equation. We will examine SAPI(m), but we cannot visually distinguish the difference in the results using $m = 2, 4, \dots, 10$. However, there is a large difference in the calculation accuracy.

Figure 9(I) demonstrates a case of $f = 0$ that results in the previously discussed problem. When there is almost no digit-loss error, $m = 2$, the average of the absolute value of the error (the absolute error) is the smallest ($1.1\text{E} - 15 \equiv 1.1 \times 10^{-15}$). The results including other cases are shown in Table VII(I). In Table VII, the area surrounded by the thick frame is the part of the intrinsic-error-zero range. Theoretically, all are the intrinsic-error-zero calculations; however, we can clearly confirm the digit loss error in the higher-order difference calculations. As a whole, the lower-order difference and the smaller number of divisions ensure the calculation accuracy. Fukuchi [6] confirmed that high-order differences do not necessarily guarantee high accuracy. First, the problem of Runge's phenomenon [16,17] occurring using polynomial interpolants can be considered a reason. This phenomenon will inevitably cause an error in the evaluation of the derivative coefficient. Second, when using higher-order differences, the digit-loss error becomes noticeable. The numerical calculation is performed in double precision, but to improve digit-loss error of the higher-order difference, it is necessary to calculate in quadruple precision.

Table VII: Calculation accuracy of SAPI(m) depending on the division number regarding the Poisson equation $\nabla_{(1)}^2 u = f$.

(I)	2	4	6	8	10
10	1.1E-15	3.0E-14	3.9E-12	2.0E-10	1.3E-07
100	1.9E-15	3.8E-12	5.1E-11	2.0E-07	1.0E-04
1000	1.3E-13	8.7E-11	3.3E-08	1.0E-05	1.3E-02
(II)k=3	2	4	6	8	10
10	2.9E-16	9.2E-15	3.1E-13	4.0E-11	4.5E-08
100	9.6E-17	7.9E-13	1.1E-11	3.7E-08	1.8E-05
1000	3.0E-14	1.8E-11	6.5E-09	2.0E-06	2.5E-03
(II)k=4	2	4	6	8	10
10	1.8E-03	6.9E-15	1.9E-13	2.0E-11	2.5E-08
100	1.7E-05	5.7E-13	8.0E-12	2.6E-08	1.2E-05
1000	1.7E-07	1.3E-11	4.7E-09	1.5E-06	1.8E-03
(II)k=5	2	4	6	8	10
10	4.6E-03	5.3E-05	1.3E-13	1.2E-11	1.4E-08
100	4.2E-05	6.2E-10	6.0E-12	1.9E-08	8.9E-06
1000	4.2E-07	9.8E-12	3.5E-09	1.1E-06	1.4E-03
(II)k=2.5	2	4	6	8	10
10	4.1E-04	9.3E-05	6.1E-05	4.7E-05	3.6E-05
100	4.8E-06	2.8E-07	1.9E-07	1.9E-07	2.2E-05
1000	5.1E-08	9.1E-10	8.5E-09	2.5E-06	3.1E-03
(III)k=4	2	4	6	8	10
10	3.8E-03	2.0E-04	2.0E-05	1.8E-06	2.1E-07
100	3.6E-05	3.9E-09	1.8E-11	3.7E-08	1.8E-05
1000	3.5E-07	1.8E-11	6.5E-09	2.0E-06	2.5E-03
(IV)k=8	2	4	6	8	10
80	5.3E-03	1.4E-04	6.9E-06	4.3E-07	5.0E-07
160	1.3E-03	5.5E-06	7.7E-08	2.7E-09	1.3E-06
800	5.2E-05	6.6E-09	4.1E-11	1.6E-07	8.0E-05

Figure 9(II) shows a case where the analytical solution is $u = x^k$. Let us first consider the case where k is a positive integer. The calculation example is the case of $k = 3$. In SAPI(m), the minimum value of m is 2. If $k = 1, 2$, it becomes the intrinsic-error-zero calculation for all SAPI(m). However, if $k = 3$ and $m = 2$, it is the intrinsic-error-zero calculation. This is because in the equally spaced central difference equation, Eq. (27), there is no third-order difference term in the error terms. The calculation results are shown in Table VII(II) for $k=3$. There is an error arising from digit-loss error of higher-order differences, so the overall trend is similar to Table VII(I). In Table VII (II) for $k = 4$, $m \geq 4$ gives the intrinsic-error-zero calculation. In the calculation of SAPI(2), it can be confirmed that it is the second-order accuracy (as the difference width is $1/10$, and the error becomes $1/100$), whereas, in the range $m \geq 4$, the digit-loss error becomes dominant. In Table VII(II) for $k = 5$, the range of

intrinsic-error-zero is $m \geq 6$, because we have to use differences other than the central difference. If the analytical solution is an n -th algebraic polynomial, the Taylor series expansion has finite terms. In this case, if it is at least $m \geq n$ in SAPI(m), it becomes the intrinsic-error-zero calculation. However, in all other cases, intrinsic errors always occur. The case where $k = 2.5$ is shown in Table VII(II) $k = 2.5$. Figure 9(III) and Table VII(III) for $k = 4$ show a case where the analytical solution is $u = \exp(kx)$. The error criterion in this case is $\exp(k)$. Figure 9(IV) and Table VII(IV) for $k = 8$ demonstrate a case where the analytical solution is $u = \cos(k\pi x)$. The calculation error of SAPI(m) exhibits a complicated appearance based on the combination of N_x , f , and digit loss errors.

Below we describe the basic features of error under double precision calculation.

- (i) As $N_x \geq 10$ and $m = 2$, for second-order differences, the error is less than 1%. In this case, the digit-loss error is minimized, and the error shrinks in proportion to $1/N_x^2$.
- (ii) Although SAPI(8) and SAPI(10) are effective when the division number N_x is small, as the number of divisions increases, the precision tends to decrease due to digit-loss errors. It should not be used with $N_x > 100$.
- (iii) In conclusion, SAPI(4) or SAPI(6) can be used for higher-order differences, and by setting $N_x \geq 1000$, the calculation accuracy $a10^{-9}$ ($1 \leq a < 10$) or less can be expected. This means that the calculation error will be within 10^{-8} . In this case, the error does not result in shrinking in proportion to $1/N_x^m$. The influence of the digit-loss error and Runge's phenomenon is involved in a complicated way.

Conclusion and Discussion

Starting from the usual second-order accurate difference, there are two methods to realize high-precision calculation with FDM. One is called compact FDM, and there are many reports on its usage [18,19]. The other is the method described in this paper.

Both are important, but the latter has not yet been systematically studied. The reason is that there is no simple algorithm for deriving higher-order differences and managing the boundary is difficult. This paper shows the possibility to fully solve these problems. The contents of this paper are summarized as follows.

(1) The solution to calculating higher-order differences is the difference coefficient matrix $\mathbf{A}$ defined in this paper, and it is obtained from Vandermonde matrix, $\mathbf{X}$, as $\mathbf{A} = \mathbf{X}^{-1}$. From this, difference schemes of each class can be obtained instantaneously. The results calculated using this method are completely equivalent to the conventional results in equally spaced central differences. It is also applied at unequally spaced grid points and has meaning as a generalized scheme of the SW scheme.

(2) The solution to elliptic PDEs is a convergent solution when the boundary is fixed in parabolic PDEs. From this perspective, there is no need to distinguish between their numerical calculation methods.

(3) In the second-order difference, the simplest scheme of parabolic PDEs in explicit methods is the FTCS scheme, assuming the simultaneous displacement of the variables. When these are successively displaced, it becomes a calculation system including all of the conventional iterative calculation methods of elliptic PDEs. This is called the TMSD scheme. This method evolves into a numerical analysis method of 2D and 3D problems.

(4) In second-order accurate FDM calculations, calculations using the TMSD and SOR are completely equivalent. However, they are not equivalent in calculations using a higher-order difference. Thus, it was confirmed that the TMSD calculation is faster in convergence.

(5) Although there is a direct method for numerically solving elliptic PDEs, the calculation using the GEM requires much time. When using the TDMA, the computation time is surprisingly shortened.

(6) There are two implicit methods: the Crank-Nicolson method and the full implicit method for solving parabolic PDEs. By adopting the β method, it becomes a single system including both. The calculation is conducted using the TDMA. At this time, if the acceleration factor is increased with the full implicit method ($\beta=1$), it will eventually converge in one set calculation. The reason is that as the acceleration factor $\alpha_c \rightarrow \infty$, the full implicit method becomes equivalent to the direct method.

(7) With the instantaneous derivation of higher-order differences, there are numerous combinations of varying difference schemes for numerically analyzing differential equations. A numerical analysis was performed by defining a unique combination of high-order difference schemes, called SAPI(m).

(8) The calculation accuracy was confirmed by a numerical analysis of the Poisson equation with an analytical solution expressed by several basic functions. As a result, it was clear that the problem of digit-loss error or Runge's phenomenon occurs in higher-order differences, and the higher-order differences do not necessarily mean greater accuracy. In the double precision calculation, it is effective up to the sixth-order difference.

(9) The calculation accuracy changes depend on $f(x)$ of the Poisson equation $\nabla_{(1)}^2 u = f(x)$ and the division number N_x . Generally, by increasing the number of divisions to $N_x = 1000$, SAPI(4) and SAPI(6) can reduce the error to within 10^{-8} . To further improve the accuracy, quadruple precision calculation is required (this should be studied in the future).

This paper described the calculation system of the IFDM-APIM in a 1D problem. It is easy to develop this into a 2D calculation system, with the following exception: the band diagonal matrix algorithm (BDMA) in the implicit method is not formulated simply as a development of the 1D problem. Fortunately, a powerful approach called the alternating direction implicit (ADI) method has been established in the field of FDM [Ref. 2, pp. 87-89]. Although this method generally assumes second-order accurate differences, it is easily expanded to higher-order differences, and this system can be created using the β method.

Then, the numerical calculation of elliptic PDEs can be performed using the full implicit method. By increasing the time difference (acceleration factor), it always converges with several iterations (usually less than 10 times), and high-accurate and high-speed calculations over arbitrary domains become possible as in the 1D problem. The details will be reported separately in the near future.

References

1. Ferziger JH, Peric M. Computational Methods for fluid Dynamics, 3rd ed. Springer. 2003; 35–37.
2. Hoffmann KA, Chiang ST. Computational fluid dynamics for engineer Volume I, Engineering Education System. Wichita, Kansas. 1993; 159-167.
3. Osher S, Fedkiw RP. Applied Mathematical Science 153 Level Set Methods and Dynamic Implicit Surfaces. New York: Springer Verlag Inc. 2003.
4. Fukuchi T. Numerical calculation of fully-developed laminar flows in arbitrary cross-sections using finite-difference method. AIP Advances. 2011; 1: 042109.
5. Fukuchi T. Numerical stability analysis and rapid algorithm for calculations of fully developed laminar flow through ducts using time-marching method. AIP Advances. 2013; 3: 032101.
6. Fukuchi T. Finite difference method and algebraic polynomial interpolation for numerically solving Poisson's equation over arbitrary domains. AIP Advances. 2014; 4: 060701.
7. Fukuchi T. Numerical analyses of steady-state seepage problems using the interpolation finite difference method, Soils and Foundations. The Japanese Geotechnical Society. 2016; 54: 608-626.
8. Fukuchi T. New high-precision empirical methods for predicting the seepage discharges and free surface locations of earth dams validated by numerical analyses using the IFDM, Soils and Foundations. The Japanese Geotechnical Society. 2018; 58: 427-445.
9. Gibou F, Fedkiw RP, Cheng LT, Kang M. A Second-Order-Accurate Symmetric Discretization of the Poisson Equation

- on Irregular Domains. *Journal of Computational Physics*. 2002; 176: 205–227.
10. Gibou F, Fedkiw RP. A Fourth Order Accurate Discretization for the Laplace and Heat Equations on Arbitrary Domains, with Applications to the Stefan Problem. *J. Comput. Phys*. 2005; 202: 577–601.
11. Ng YT, Chen H, Min C, Gibou F. Guidelines for Poisson solvers on irregular domains with Dirichlet boundary conditions using the ghost fluid method. *J. Sci. Comput*. 2009; 41: 300–320.
12. Hancock MJ. The 1-D Heat Equation: 18.303 Linear Partial Differential Equations. 2006. Available Online at: <https://ocw.mit.edu/courses/mathematics/18-303-linear-partial-differential-equations-fall-2006/lecture-notes/heateqni.pdf>
13. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in C, The Art of Scientific Computing*. Second Edition. Cambridge: Cambridge University Press. 1992.
14. Shortley GH, Weller R. The numerical solution of Laplace's equation. *J. Appl. Phys*. 1938; 24: 334–348
15. Forsythe GE, Wasow WR. *Finite-Difference Methods for Partial Differential Equations*, Dover Phoenix Editions. United States: Dover Publications, Inc. 2004.
16. Klein G, Berrut JP. Linear rational finite differences from derivatives of barycentric interpolants. *siam j. Numer. Anal*. 2012; 50: 643–656.
17. Web-Runge's phenomenon, 2019/02/15. Available Online at: https://en.wikipedia.org/wiki/Runge%27s_phenomenon
18. Lynch RE, Rice JR. A high-order difference method for differential equations. *Math. Comp*. 1980; 34: 333–372.
19. Li M, Tang T. A compact fourth-order finite difference scheme for the steady incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*. 1995; 20: 1137–1151.