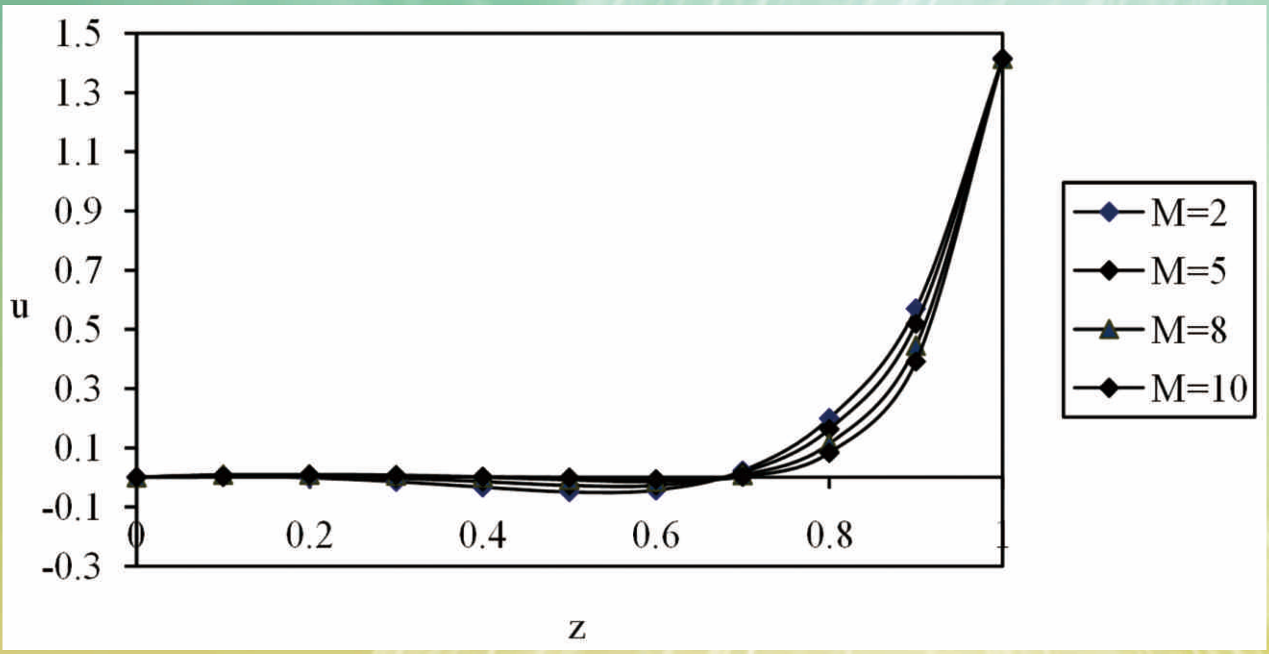
ISSN: 2161-1203 Vol. 4, No. 5, December 2014

Scientific Research

American Journal of Computational Mathematics



ISSN: 2161-1203

www.scirp.org/journal/ajcm

Journal Editorial Board

ISSN: 2161-1203 (Print) 2161-1211 (Online)
http://www.scirp.org/journal/ajcm

American Journal of Computational Mathematics, 2014, 4, 387-481
Published Online December 2014 in SciRes. http://www.scirp.org/journal/ajcm

Table of Contents

Volume 4 Number 5 December 2014

The figure on the front cover is from the article published in American Journal of Computational Mathematics, 2014, Vol. 4, No. 5, pp. 396-405 by P. Sulochana.

American Journal of Computational Mathematics (AJCM)

Journal Information

SUBSCRIPTIONS

The *American Journal of Computational Mathematics* (Online at Scientific Research Publishing, www.SciRP.org) is published quarterly by Scientific Research Publishing, Inc., USA.

Subscription rates:

Print: $79 per issue.
To subscribe, please contact Journals Subscriptions Department, E-mail: sub@scirp.org

SERVICES

Advertisements
Advertisement Sales Department, E-mail: service@scirp.org

Reprints (minimum quantity 100 copies)
Reprints Co-ordinator, Scientific Research Publishing, Inc., USA.
E-mail: sub@scirp.org

COPYRIGHT

PRODUCTION INFORMATION

For manuscripts that have been accepted for publication, please contact:
E-mail: ajcm@scirp.org

American Journal of Computational Mathematics, 2014, 4, 387-395
Published Online December 2014 in SciRes. http://www.scirp.org/journal/ajcm
http://dx.doi.org/10.4236/ajcm.2014.45033

Fast and Numerically Stable Approximate Solution of Trummer's Problem

Mohammad M. Tabanjeh

Department of Mathematics and Computer Science, Virginia State University, Petersburg, USA
Email: mtabanjeh@vsu.edu

Received 9 September 2014; revised 10 October 2014; accepted 20 October 2014

Abstract

Trummer's problem is the problem of multiplication of an $n \times n$ Cauchy matrix C by a vector. It serves as the basis for the solution of several problems in scientific computing and engineering [1]. The straightforward algorithm solves Trummer's problem in $O(n^2)$ flops. The fast algorithm solves the problem in $O(n\log^2 n)$ flops [2] but has poor numerical stability. The algorithm we discuss here in this paper is the celebrated multipoint algorithm [3] which has been studied by Pan *et al.* The algorithm approximates the solution in $O(n\log n)$ flops in terms of n but its cost estimate depends on the bound of the approximation error and also depends on the correlation between the entries of the pair of n-dimensional vectors defining the input matrix C.

Keywords

Cauchy Matrix, Mulipoint Algorithm, Structure Matrices, Displacement Operators

1. Introduction

Computations with dense structured matrices have many applications in sciences, communications and engineering. The structure enables dramatic acceleration of the computations and major decrease in memory space but sometimes leads to numerical stability problems. The best well-known classes of structured matrices are *Toeplitz*, *Hankel*, *Cauchy* and *Vandermonde* matrices.

The computations with such matrices are widely applied in the areas of algebraic coding, control, signal processing, solution of partial differential equations and algebraic computing. For example, Toeplitz matrices arise in some major signal processing computations and the problem of multiplying Vandermonde matrix by a vector is equivalent to polynomial evaluation, whereas solving a Vandermonde system is equivalent to polynomial interpolation. Moreover, Cauchy matrices appear in the study of integral equations and conformal mappings. The complexity of computations with $n \times n$ dense structured matrices dramatically decreases in comparison with the general $n \times n$ matrices, that is, from the order of n^2 words of storage space and n^{α} arithmetic operations (ops) with $2.37 < \alpha \le 3$ in the best algorithms, to $O(n)$ words of storage space and $O(n\log n)$ ops (see **Table 1** below for more details).

2. Some Basic Definitions

Throughout this paper, we use the following notations; $\mathbb{N}$ denotes the set of nonnegative integers, $\mathbb{R}^{+}$ is the set of positive real numbers, $\mathbf{Z}^{+}$ denotes the set of positive integers, and $\mathbf{R}$ denotes the set of real numbers.

Definition 2.1. A matrix $\boldsymbol{T} = \left[t_{i,j}\right]_{i,j=0}^{n-1}$ is a Toeplitz matrix if $t_{i,j} = t_{i+1,j+1}$ for every pair of its entries $t_{i,j}$ and $t_{i+1,j+1}$. A matrix $\boldsymbol{H} = \left[h_{i,j}\right]_{i,j=0}^{n-1}$ is a Hankel matrix if $h_{i,j} = h_{i-1,j+1}$ for every pair of its entries $h_{i,j}$ and $h_{i-1,j+1}$.

Definition 2.2. For a given vector $\boldsymbol{v} = (v_i)_{i=0}^{n-1}$, the matrix $\boldsymbol{V} = V(\boldsymbol{v})$ of the form $\boldsymbol{V} = \left[v_i^j\right]_{i,j=0}^{n-1}$ is called a Vandermonde matrix.

Definition 2.3. Given two vectors $\boldsymbol{s}$ and $\boldsymbol{t}$ such that $s_i \ne t_j$ for all i and j, the $n \times n$ matrix $\boldsymbol{C} = C(\boldsymbol{s},\boldsymbol{t})$ is a Cauchy (generalized Hilbert) where

$$C(\boldsymbol{s},\boldsymbol{t}) = \left[\frac{1}{s_i - t_j}\right]_{i,j=0}^{n-1}.$$

For more details regarding the four classes of structured matrices, see **Table 2** below.

Table 1. Parameter and flops count.

Matrices A size $n \times n$	Number of parameters for A	Number of flops required for Multiplication by a vector
General	n^2	$2n^2 - 2n$
Toeplitz	$2n-1$	$O(n\log n)$
Hankel	$2n-1$	$O(n\log n)$
Vandermonde	n	$O(n\log^2 n)$
Cauchy	$2n$	$O(n\log^2 n)$

Table 2. General definition of the four classes of structured matrices.

Toeplitz matrices, $\boldsymbol{T} = \left[t_{i-j}\right]_{i,j=0}^{n-1}$	Hankel matrices, $\boldsymbol{H} = \left[h_{i+j}\right]_{i,j=0}^{n-1}$
$\begin{bmatrix} t_0 & t_{-1} & \cdots & t_{1-n} \\ t_1 & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \cdots & t_1 & t_0 \end{bmatrix}$	$\begin{bmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_1 & h_2 & ⋰ & h_n \\ \vdots & ⋰ & ⋰ & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} \end{bmatrix}$
Vandermonde matrices, $\boldsymbol{V} = \left[v_i^j\right]_{i,j=0}^{n-1}$	Cauchy matrices, $\boldsymbol{C} = \left[\frac{1}{x_i - y_j}\right]_{i,j=0}^{n-1}$
$\begin{bmatrix} 1 & v_0 & \cdots & v_0^{n-1} \\ 1 & v_1 & \cdots & v_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & v_{n-1} & \cdots & v_{n-1}^{n-1} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{x_0 - y_0} & \cdots & \frac{1}{x_0 - y_{n-1}} \\ \frac{1}{x_1 - y_0} & \cdots & \frac{1}{x_1 - y_{n-1}} \\ \vdots & \ddots & \vdots \\ \frac{1}{x_{n-1} - y_0} & \cdots & \frac{1}{x_{n-1} - y_{n-1}} \end{bmatrix}$

Remark 2.1. It is quite easy to verify that $\boldsymbol{TJ}$ and $\boldsymbol{JT}$ are Hankel matrices if $\boldsymbol{T}$ is a Toeplitz matrix, and $\boldsymbol{HJ}$ and $\boldsymbol{JH}$ are Toeplitz matrices if $\boldsymbol{H}$ is a Hankel matrix where $\boldsymbol{J}$ is the following reflection matrix,

$$\boldsymbol{J} = \begin{pmatrix} 0 & & 1 \\ & \cdot^{\cdot^{\cdot}} & \\ 1 & & 0 \end{pmatrix}.$$

3. The Displacement Operators of Dense Structured Matrices

The concept of displacement operators and displacement rank which was introduced by T. Kailath, S. Y. Kung, and M. Morf in 1979 and studied by Pan, Bini, and other authors is one of the powerful tools for studying and dealing with matrices that have structure. The displacement rank approach, when it was initially introduced, was intended for more restricted use [4], namely, to measure how "close" to Toeplitz a given matrix is. Then the idea turned out to be even more powerful, thus it was developed, generalized and extended to other structured matrices. In this section, we consider the most general and modern interpretation of the displacement of a matrix.

The main idea is, for a given structured matrix $\boldsymbol{A}$, we need to find an operator L that transforms the matrix into a low rank matrix $L(\boldsymbol{A})$ such that one can easily recover $\boldsymbol{A}$ from its image $L(\boldsymbol{A})$ and operate with low rank matrices instead. Such operators that shift and scale the entries of the structured matrices turn out to be appropriate tools for introducing and defining the matrices of *Toeplitz-like*, *Hankel-like*, *Vandermonde-like*, and *Cauchy-like* types [5].

Definition 3.1. For any fixed field $\mathbb{F}$ such as the complex field $\mathbb{C}$ and a fixed pair $\{\boldsymbol{M},\boldsymbol{N}\}$ of operator matrices, we define the linear displacement operators $L:\mathbb{F}^{m\times n}\to\mathbb{F}^{m\times n}$ of Sylvester type,

$$L(\boldsymbol{A}) = \nabla_{\boldsymbol{M},\boldsymbol{N}}(\boldsymbol{A}) = \boldsymbol{MA}-\boldsymbol{AN}$$

and Stein type,

$$L(\boldsymbol{A}) = \nabla_{\boldsymbol{M},\boldsymbol{N}}(\boldsymbol{A}) = \boldsymbol{A}-\boldsymbol{MAN}\ .$$

The image $L(\boldsymbol{A})$ of the operator L is called the *displacement* of the matrix $\boldsymbol{A}$. The operators of Sylvester and Stein types can be transformed easily into one another if at least one of the two associated operator matrices is non-singular. The following theorem explains this fact.

Theorem 3.1. $\nabla_{\boldsymbol{M},\boldsymbol{N}} = \boldsymbol{M}\nabla_{\boldsymbol{M}^{-1},\boldsymbol{N}}$ if the operator matrix $\boldsymbol{M}$ is non-singular, and $\nabla_{\boldsymbol{M},\boldsymbol{N}} = -\nabla_{\boldsymbol{M},\boldsymbol{N}^{-1}}$ if the operator matrix $\boldsymbol{N}$ is non-singular.

Proof:

$$\nabla_{\boldsymbol{M},\boldsymbol{N}}(\boldsymbol{A}) = \boldsymbol{MA}-\boldsymbol{AN} = \boldsymbol{MA}-\boldsymbol{MM}^{-1}\boldsymbol{AN} = \boldsymbol{M}\left(\boldsymbol{A}-\boldsymbol{M}^{-1}\boldsymbol{AN}\right) = \boldsymbol{M}\Delta_{\boldsymbol{M}^{-1},\boldsymbol{N}}(\boldsymbol{A}),$$

$$\nabla_{\boldsymbol{M},\boldsymbol{N}}(\boldsymbol{A}) = \boldsymbol{MA}-\boldsymbol{AN} = \left(\boldsymbol{MAN}^{-1}\boldsymbol{N}-\boldsymbol{AN}\right) = \left(\boldsymbol{MAN}^{-1}-\boldsymbol{A}\right)\boldsymbol{N} = -\left(\boldsymbol{A}-\boldsymbol{MAN}^{-1}\right)N = -\Delta_{\boldsymbol{M},\boldsymbol{N}^{-1}}(\boldsymbol{A})\boldsymbol{N}.$$

The operator matrices that we will be using are the matrices $\boldsymbol{Z}_f$, $\boldsymbol{Z}_f^{\mathrm{T}}$, and $D(\boldsymbol{d})$ where $\boldsymbol{Z}_f$, is the unit f-circulant matrix,

$$\boldsymbol{Z}_f = \begin{bmatrix} 0 & \cdots & 0 & f \\ 1 & \ddots & \vdots & \vdots \\ \vdots & \cdots & & 0 \\ 0 & \cdots & 1 & 0 \end{bmatrix}$$

f is any scalar, $\boldsymbol{Z}_f^{\mathrm{T}}$ is the transpose of $\boldsymbol{Z}_f$, and $D(\boldsymbol{d})$ is a diagonal matrix with diagonal entries $d_0,\cdots,d_{n-1}$,

$$D(\boldsymbol{d}) = \operatorname{diag}(d_i)_{i=0}^{n-1} = \begin{pmatrix} d_0 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & d_{n-1} \end{pmatrix}.$$

We may use the operator matrices $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_0$ in the case of Toeplitz matrices, $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_1^{\mathrm{T}}$ in the case of Hankel matrices, $\boldsymbol{Z}_0$ and $D(\boldsymbol{v})$ in the case of Vandermonde matrices, and $D(\boldsymbol{x})$ and $D(\boldsymbol{y})$ in the case of Cauchy matrices. However, there are other choices of operator matrices that can transform these matrices to low rank.

4. The Correlation of Structured Matrices to Polynomials

The product

$$V(\boldsymbol{x})\boldsymbol{p}=\boldsymbol{v} \tag{1}$$

represents the vector $\boldsymbol{v}=(v_i)$ of the values of the polynomial $p(x)=\sum_j p_j x^j$ on a node set $\{x_i\}$.

If $\boldsymbol{x}=\boldsymbol{w}=\left(w_n^i\right)_{i=0}^{n-1}$ is the vector of the n^{th} roots of unity,

$$w_n=\mathrm{e}^{\frac{2\pi}{n}i}=\cos\left(\frac{2\pi}{n}\right)+i\sin\left(\frac{2\pi}{n}\right),\ \text{where } i=\sqrt{-1},\ \text{and } w_n^n=1,$$

then the matrix $F=\frac{V(\boldsymbol{w})}{\sqrt{n}}$ and multipoint evaluation turns into discrete Fourier transform which takes only $O(n\log n)$ ops and allows numerically stable implementation according to [6]. If we express $V(\boldsymbol{x})$ in Equation (1) via Cauchy matrices we will get

$$V(x)=\frac{1}{\sqrt{n}}\mathrm{dia}\left(1-x_i^n\right)_{i=0}^{n-1}C(x,w)\,\mathrm{diag}\left(w_i\right)_{i=0}^{n-1}F. \tag{2}$$

Here, $D(\boldsymbol{h})=\mathrm{diag}(h_i)_{i=0}^{n-1}$ for $h=(h_i)_{i=0}^{n-1}$ denotes $n\times n$ diagonal matrix with diagonal entries $h_0,\cdots,h_{n-1}$.

Note that the numerical stability is very important in approximation algorithm for multipoint polynomial evaluation. It relies on expressing $V(\boldsymbol{x})$ in terms of Cauchy matrices as in Equation (2). Clearly in Equation (2), the product $V(\boldsymbol{x})$ by a vector has been reduced to ones with Cauchy, which brings us to Trummer's problem, that is, the problem of multiplication of Cauchy matrix by a vector. Its solution by multipoint algorithm ([6], pp. 261-262) leads to multipoint polynomial evaluation based on Equation (2) which is fast in terms of ops and numerically stable as it was proved by Pan.

We may vary the vector $\boldsymbol{x}$ by linearly mapping it to the vector $\boldsymbol{y}=a\boldsymbol{x}+b\boldsymbol{e}$ where we can take $\boldsymbol{e}=(1)_{i=0}^{n-1}$ and $0\neq a$ and b are any scalars.

5. New Transformation of Cauchy Matrices

As we mentioned earlier, Trummer's problem is the problem of multiplication of an $n\times n$ Cauchy matrix $\boldsymbol{C}$ by a vector which is the basis for the solution of many important problems of scientific computing and engineering. The straightforward algorithm solves Trummer's problem in $O(n^2)$ flops. The fast algorithm solves the problem in $O(n\log^2 n)$ flops but has poor numerical stability.

The algorithm we presenting in this paper approximates the solution in $O(n\log n)$ flops in terms of n. However, its cost estimate depends on the bound of the approximation error and on the correlation between the entries of the pair of n-dimensional vectors defining the input matrix $\boldsymbol{C}$. This algorithm is numerically stable as we will see throughout this section and the next section.

The main goal in this paper is to enrich the power of the multipoint algorithm by introducing and proving some new expressions for Cauchy matrix via other Cauchy matrices [7], which we may vary by changing one of their basis vectors. Under a certain choice of such a vector, the solution of Trummer's problem will be simplified; thus, the power of the multipoint algorithm can be developed as we will see in the next section.

Therefore, we will achieve our goal by using a simple transformation of the useful basic formula of [8], and the resulting expressions for $\boldsymbol{C}$ will give us further algorithmic opportunities.

Definition 5.1. For a pair of n-dimensional vectors $\boldsymbol{a}=(a_i)_{i=0}^{n-1}$, $\boldsymbol{b}=(b_j)_{j=0}^{n-1}$ let $C(\boldsymbol{a},\boldsymbol{b})=\left(1/(a_i-b_i)\right)_{i,j=0}^{n-1}$, $V(\boldsymbol{a})=\left(a_i^j\right)_{i,j=0}^{n-1}$, $H(\boldsymbol{a})=\left(h_{i,j}\right)_{i,j=0}^{n-1}$, $h_{i,j}=a_{i,j}$ for $i+j\leq n-1$, $h_{i,j}=0$ for $i+j\geq n-1$, denote the associated $n\times n$ Cauchy, Vandermonde, and triangular Hankel matrices, respectively. For a vector $\boldsymbol{a}=(a_i)_{i=0}^{n-1}$ with $a_i\neq a_j$ for $i\neq j$ a Cauchy degenerate matrix $C(\boldsymbol{a})$ has the diagonal entries zeros and the $(i,j)^{th}$ entry

$1/(a_i - a_j)$ for $i \neq j$. Furthermore, $p_{\boldsymbol{b}}(x)$ denote the polynomial $p_{\boldsymbol{b}}(x) = \prod_{j=0}^{n-1}(x - b_j)$ and its derivative $p'_{\boldsymbol{b}}(x) = \sum_{i=0}^{n-1}\prod_{j=0(j\neq i)}^{n-1}(x - b_j)$. Lastly,

$$D(\boldsymbol{a},\boldsymbol{b}) = \operatorname{diag}\left(p_{\boldsymbol{b}}(a_i)\right)_{i=0}^{n-1} = \operatorname{diag}\left(\prod_{j=0}^{n-1}(a_i - b_j)\right)_{i=0}^{n-1}$$

and

$$D'(\boldsymbol{b}) = \operatorname{diag}\left(p'_{\boldsymbol{b}}(b_i)\right)_{j=0}^{n-1} = \operatorname{diag}\left(\sum_{i=0}^{n-1}\prod_{j=0}^{n-1}(b_i - b_j)\right)$$

denote a pair of $n \times n$ diagonal matrices, defined by the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

Theorem 5.1. (See [8]) Let $c_i \neq d_j$, $i, j = 0,1,\cdots,n-1$. Then

$$C(\boldsymbol{c},\boldsymbol{d}) = D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) H(\boldsymbol{d}) V(\boldsymbol{d})^{\mathrm{T}} \tag{3}$$

$$C(\boldsymbol{c},\boldsymbol{d}) = D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) V(\boldsymbol{d})^{-1} D'(\boldsymbol{d}). \tag{4}$$

The main idea of the transformation of the basic vectors defining the problem is taken from [9], where this idea was used for multipoint polynomial evaluation and interpolation.

Definition 5.2. Trummer's problem is the problem of computing the vector $C(\boldsymbol{a},\boldsymbol{b})\boldsymbol{v}$ for three given vectors $\boldsymbol{a} = (a_i)_{i=0}^{n-1}$, $\boldsymbol{b} = (b_j)_{j=0}^{n-1}$ and $\boldsymbol{v} = (v_j)_{j=0}^{n-1}$ where $a_i \neq b_j$ for all pairs *i*, *j*. Trummer's degenerate problem is the problem of computing the vector $C(\boldsymbol{a})\boldsymbol{v}$ for two given vectors $\boldsymbol{a} = (a_i)_{i=0}^{n-1}$ and $\boldsymbol{v} = (v_j)_{j=0}^{n-1}$ where $a_i \neq a_j$ for $i \neq j$.

Definition 5.3. $w_k = \exp(2\pi i/k)$, where $i = \sqrt{-1}$, is a primitive k^{th} root of 1, $w_k^k = 1$, $w_k^l \neq 1$ for $l = 1,\cdots,k-1$.

Lemma 5.1. $\sum_{l=0}^{k-1}(w_k^i)^l = 0$ for $i = 1,\cdots,k-1$.

Approximate solution of Trummer's degenerate problem can be reduced to Trummer's problem due to the next simple result.

Lemma 5.2. $C(\boldsymbol{c}) = 1/h\sum_{i=0}^{h-1} C(\boldsymbol{c},\boldsymbol{c}+\delta w_h^i \boldsymbol{e}) + O(\delta^h)$ as $\delta \to 0$, where $\boldsymbol{e} = (1)_{j=0}^{n-1}$ is the vector filled with the values one and δ is a scalar parameter.

Proof: $\sum_{g=0}^{h-1} 1/(c_i - c_j - \delta w_h^g) = 1/(c_i - c_j)\sum_{l=0}^{\infty}\sum_{g=0}^{h-1}(\delta w_h^g/c_i - c_j)^l = h/(c_i - c_j)(1 + O(\delta^h))$ due to Lemma 5.1.

6. Transformations of Cauchy Matrices and Trummer's Problem

Theorem 6.1. For a triple of *n*-dimensional vector $\boldsymbol{b} = (b_i)_{i=0}^{n-1}$, $\boldsymbol{c} = (c_j)_{j=0}^{n-1}$, $\boldsymbol{d} = (d_k)_{k=0}^{n-1}$ where $b_i \neq c_j$, $c_j \neq d_k$, $d_k \neq b_i$ for $i, j, k = 0,\cdots,n-1$ we have the following matrix equations:

$$C(\boldsymbol{c},\boldsymbol{d}) = D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) V(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{d}) C(\boldsymbol{b},\boldsymbol{d}) \tag{5}$$

$$C(\boldsymbol{c},\boldsymbol{d}) = D(\boldsymbol{c},\boldsymbol{d})^{-1} D(\boldsymbol{c},\boldsymbol{b}) C(\boldsymbol{c},\boldsymbol{b}) D'(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{d}) C(\boldsymbol{b},\boldsymbol{d}) \tag{6}$$

$$C(\boldsymbol{c},\boldsymbol{d}) = C(\boldsymbol{c},\boldsymbol{b}) D(\boldsymbol{b},\boldsymbol{c}) V(\boldsymbol{b})^{-\mathrm{T}} V(\boldsymbol{d})^{\mathrm{T}} D(\boldsymbol{d},\boldsymbol{c})^{-1} \tag{7}$$

$$C(\boldsymbol{c},\boldsymbol{d}) = -C(\boldsymbol{c},\boldsymbol{b}) D(\boldsymbol{b},\boldsymbol{c}) D'(\boldsymbol{b})^{-1} C(\boldsymbol{b},\boldsymbol{d}) D(\boldsymbol{d},\boldsymbol{b}) D(\boldsymbol{d},\boldsymbol{c})^{-1}. \tag{8}$$

Proof Theorem 6.1:

1) Proof of Equation (5):

From Equation (3), we obtain $C(\boldsymbol{b},\boldsymbol{d})^{-1} = V(\boldsymbol{d})^{-\mathrm{T}} H(\boldsymbol{d})^{-1} V(\boldsymbol{b})^{-1} V(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{d})$. This is done by taking the inverse of Equation (3) and replacing the vectors ***c***, ***d*** by ***b***, ***d***. Then substitute the equation

$$C(\boldsymbol{b},\boldsymbol{d})^{-1} = V(\boldsymbol{d})^{-\mathrm{T}} H(\boldsymbol{d})^{-1} V(\boldsymbol{b})^{-1} V(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{d})$$

and Equation (3) for $C(\boldsymbol{c},\boldsymbol{d})$ into the following matrix identity

$$C(\boldsymbol{c},\boldsymbol{d}) = C(\boldsymbol{c},\boldsymbol{d}) C(\boldsymbol{b},\boldsymbol{d})^{-1} C(\boldsymbol{b},\boldsymbol{d}).$$

This gives:

$$\begin{aligned} C(\boldsymbol{c},\boldsymbol{d}) &= C(\boldsymbol{c},\boldsymbol{d})\left[V(\boldsymbol{d})^{-\mathrm{T}} H(\boldsymbol{d})^{-1} V(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{d})\right] C(\boldsymbol{b},\boldsymbol{d}) \\ &= D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) H(\boldsymbol{d}) V(\boldsymbol{d})^{\mathrm{T}} \left[V(\boldsymbol{d})^{-\mathrm{T}} H(\boldsymbol{d})^{-1} V(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{d})\right] C(\boldsymbol{b},\boldsymbol{d}) \\ &= D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) H(\boldsymbol{d}) \left[V(\boldsymbol{d})^{\mathrm{T}} V(\boldsymbol{d})^{-\mathrm{T}}\right] H(\boldsymbol{d})^{-1} V(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{d}) C(\boldsymbol{b},\boldsymbol{d}) \\ &= D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) \left[H(\boldsymbol{d}) H(\boldsymbol{d})^{-1}\right] V(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{d}) C(\boldsymbol{b},\boldsymbol{d}) \\ &= D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) V(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{d}) C(\boldsymbol{b},\boldsymbol{d}). \end{aligned}$$

Clearly, the last one is just Equation (5).

2) Proof of Equation (6):

From Equation (4); $C(\boldsymbol{c},\boldsymbol{d}) = D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) V(\boldsymbol{d})^{-1} D'(\boldsymbol{d})$ replace the vector ***d*** by ***b***, then we will get:

$$C(\boldsymbol{c},\boldsymbol{b}) = D(\boldsymbol{c},\boldsymbol{b})^{-1} V(\boldsymbol{c}) V(\boldsymbol{b})^{-1} D'(\boldsymbol{b}).$$

Then solve for $V(\boldsymbol{c}) V(\boldsymbol{b})^{-1}$;

$$V(\boldsymbol{c}) V(\boldsymbol{b})^{-1} = D(\boldsymbol{c},\boldsymbol{b}) C(\boldsymbol{c},\boldsymbol{b}) D'(\boldsymbol{b})^{-1}.$$

Now substitute the last expression into Equation (5) and obtain the following:

$$\begin{aligned} C(\boldsymbol{c},\boldsymbol{d}) &= D(\boldsymbol{c},\boldsymbol{d})^{-1} \left[D(\boldsymbol{c},\boldsymbol{b}) C(\boldsymbol{c},\boldsymbol{b}) D'(\boldsymbol{b})^{-1}\right] D(\boldsymbol{b},\boldsymbol{d}) C(\boldsymbol{b},\boldsymbol{d}) \\ &= D(\boldsymbol{c},\boldsymbol{d})^{-1} D(\boldsymbol{c},\boldsymbol{b}) C(\boldsymbol{c},\boldsymbol{b}) D'(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{d}) C(\boldsymbol{b},\boldsymbol{d}) \end{aligned}$$

which is obviously Equation (6).

3) Proof of Equation (7):

From Equation (4); $C(\boldsymbol{c},\boldsymbol{d}) = D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) V(\boldsymbol{d})^{-1} D'(\boldsymbol{d})$ first solve for $V(\boldsymbol{d})^{-1} D'(\boldsymbol{d})$ to get:

$$V(\boldsymbol{d})^{-1} D'(\boldsymbol{d}) = V(\boldsymbol{c})^{-1} D(\boldsymbol{c},\boldsymbol{d}) C(\boldsymbol{c},\boldsymbol{d}),$$

replace $\boldsymbol{c} \leftrightarrow \boldsymbol{d}$ to get:

$$V(\boldsymbol{c})^{-1} D'(\boldsymbol{c}) = V(\boldsymbol{d})^{-1} D(\boldsymbol{d},\boldsymbol{c}) C(\boldsymbol{d},\boldsymbol{c}).$$

Now, replace the vector ***d*** by ***b*** and obtain

$$V(\boldsymbol{c})^{-1} D'(\boldsymbol{c}) = V(\boldsymbol{b})^{-1} D(\boldsymbol{b},\boldsymbol{c}) C(\boldsymbol{b},\boldsymbol{c}). \tag{9}$$

Since $C(\boldsymbol{c},\boldsymbol{d}) = -C(\boldsymbol{d},\boldsymbol{c})^{\mathrm{T}}$ we have also $-C(\boldsymbol{c},\boldsymbol{d}) = C(\boldsymbol{d},\boldsymbol{c})^{\mathrm{T}}$.

Start with Equation (4) which is $C(\boldsymbol{c},\boldsymbol{d}) = D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) V(\boldsymbol{d})^{-1} D'(\boldsymbol{d})$ and replace the vector $\boldsymbol{c} \leftrightarrow \boldsymbol{d}$ to obtain $C(\boldsymbol{d},\boldsymbol{c}) = D(\boldsymbol{d},\boldsymbol{c})^{-1} V(\boldsymbol{d}) V(\boldsymbol{c}) D'(\boldsymbol{c})$, then use the equation $-C(\boldsymbol{c},\boldsymbol{d}) = C(\boldsymbol{d},\boldsymbol{c})^{\mathrm{T}}$ to get:

$$\begin{aligned} -C(\boldsymbol{c},\boldsymbol{d}) = C(\boldsymbol{d},\boldsymbol{c})^{\mathrm{T}} &= \left[D(\boldsymbol{d},\boldsymbol{c})^{-1} V(\boldsymbol{d}) \left[V(\boldsymbol{c}) D'(\boldsymbol{c})\right]\right]^{\mathrm{T}} \\ &= \left[D(\boldsymbol{d},\boldsymbol{c})^{-1} V(\boldsymbol{d}) \left[V^{-1}(\boldsymbol{b}) D(\boldsymbol{b},\boldsymbol{c}) C(\boldsymbol{b},\boldsymbol{c})\right]\right]^{\mathrm{T}} \text{ (from Equation (9))} \\ &= \left[D(\boldsymbol{d},\boldsymbol{c})^{-1} V(\boldsymbol{d}) V^{-1}(\boldsymbol{b}) D(\boldsymbol{b},\boldsymbol{c}) C(\boldsymbol{b},\boldsymbol{c})\right]^{\mathrm{T}} \\ &= C(\boldsymbol{b},\boldsymbol{c})^{\mathrm{T}} D(\boldsymbol{b},\boldsymbol{c}) V(\boldsymbol{b})^{-\mathrm{T}} V^{\mathrm{T}}(\boldsymbol{d}) D(\boldsymbol{d},\boldsymbol{c})^{-1}. \end{aligned}$$

Now replace $C(\boldsymbol{b},\boldsymbol{c})^{\mathrm{T}}$ in the last equation by $-C(\boldsymbol{c},\boldsymbol{b})$ (that is, use the identity $-C(\boldsymbol{c},\boldsymbol{b}) = C(\boldsymbol{b},\boldsymbol{c})^{\mathrm{T}}$)

$$-C(\boldsymbol{c},\boldsymbol{d}) = -C(\boldsymbol{b},\boldsymbol{c}) D(\boldsymbol{b},\boldsymbol{c}) V(\boldsymbol{b})^{-\mathrm{T}} V(\boldsymbol{d})^{\mathrm{T}} D(\boldsymbol{d},\boldsymbol{c})^{-1}.$$

This implies that $C(\boldsymbol{c},\boldsymbol{d}) = C(\boldsymbol{c},\boldsymbol{b}) D(\boldsymbol{b},\boldsymbol{c}) V(\boldsymbol{b})^{-\mathrm{T}} V(\boldsymbol{d})^{\mathrm{T}} D(\boldsymbol{d},\boldsymbol{c})^{-1}$ which is Equation (7).

4) Proof of Equation (8):

From Equation (4): $C(\boldsymbol{c},\boldsymbol{d}) = D(\boldsymbol{c},\boldsymbol{d})^{-1} V(\boldsymbol{c}) V(\boldsymbol{d})^{-1} D'(\boldsymbol{d})$ solve for $V(\boldsymbol{c}) V(\boldsymbol{d})^{-1}$ and obtain

$$D(\boldsymbol{c},\boldsymbol{d})C(\boldsymbol{c},\boldsymbol{d})D'(\boldsymbol{d})^{-1}=V(\boldsymbol{c})V(\boldsymbol{d})^{-1}.$$

Expand Equation (4) $C(\boldsymbol{c},\boldsymbol{d})D(\boldsymbol{c},\boldsymbol{d})D'(\boldsymbol{d})^{-1}=V(\boldsymbol{c})V(\boldsymbol{d})^{-1}$, and change $\boldsymbol{c}\leftrightarrow\boldsymbol{d}$ and $\boldsymbol{d}\leftrightarrow\boldsymbol{b}$ to get the equation: $D(\boldsymbol{d},\boldsymbol{b})C(\boldsymbol{d},\boldsymbol{b})D'(\boldsymbol{b})^{-1}=V(\boldsymbol{d})V(\boldsymbol{b})^{-1}$ and take the transpose of both side of the last equation to get:

$$V(\boldsymbol{b})^{-\mathrm{T}}V(\boldsymbol{d})^{\mathrm{T}}=D'(\boldsymbol{b})^{-1}C(\boldsymbol{d},\boldsymbol{b})^{\mathrm{T}}D(\boldsymbol{d},\boldsymbol{b}). \quad (10)$$

Substitute Equation (10) and the matrix equation $C(\boldsymbol{d},\boldsymbol{b})^{\mathrm{T}}=-C(\boldsymbol{b},\boldsymbol{d})$ into Equation (7):

$$C(\boldsymbol{c},\boldsymbol{d})=C(\boldsymbol{c},\boldsymbol{b})D(\boldsymbol{b},\boldsymbol{c})\left[V(\boldsymbol{b})^{-\mathrm{T}}V(\boldsymbol{d})^{\mathrm{T}}\right]D(\boldsymbol{d},\boldsymbol{c})^{-1}$$

$$C(\boldsymbol{c},\boldsymbol{d})=C(\boldsymbol{c},\boldsymbol{b})D(\boldsymbol{b},\boldsymbol{c})\left[D'(\boldsymbol{b})^{-1}C(\boldsymbol{d},\boldsymbol{b})^{\mathrm{T}}D(\boldsymbol{d},\boldsymbol{b})\right]D(\boldsymbol{d},\boldsymbol{c})^{-1} \quad \text{(this is from Equation (10))}$$

$$=-C(\boldsymbol{c},\boldsymbol{b})D(\boldsymbol{b},\boldsymbol{c})D'(\boldsymbol{b})^{-1}C(\boldsymbol{b},\boldsymbol{d})D(\boldsymbol{d},\boldsymbol{b})D(\boldsymbol{d},\boldsymbol{c})^{-1} \quad \text{which is Equation (8).}$$

7. Approximate Stable Solutions of Trummer's Problems

The algorithm we are studying and presenting in this section depends on the multipoint algorithm which approximates the solution of Trummer's problem in $O(n)$ ops in terms of *n*, and works efficiently for a large classes of input vectors but sometimes has problems with some of the input vectors, especially if the ratio of the input vectors is close to 1.

Recall, the power series expansion: $\sum_{i=0}^{\infty}x^i=1+x+x^2+x^3+\cdots$ this series converges whenever $|x|<1$ and has a sum $\frac{1}{1-x}$.

The basis for the algorithm is the following expressions:

$$\frac{1}{s_i-t_j}=\frac{1}{s_i\left(1-\left(\frac{t_j}{s_i}\right)\right)}=\frac{1}{s_i}\sum_{k=0}^{\infty}\left(\frac{t_j}{s_i}\right)^k \quad (11)$$

where $\boldsymbol{s}=(s_i)_{i=0}^{n-1}$ and $\boldsymbol{t}=(t_i)_{i=0}^{n-1}$. Clearly this series converges whenever $\left|\frac{t_j}{s_i}\right|<1$. Now for large *M* the expression $\frac{1}{s_i}\sum_{k=0}^{M}\left(\frac{t_i}{s_i}\right)^k$ approximates $\frac{1}{s_i-t_j}$. On the other hand $\frac{1}{s_i-t_j}$ can be also written as

$$\frac{1}{s_i-t_j}=\frac{1}{-t_j\left(1-\left(\frac{s_i}{t_j}\right)\right)}=-\frac{1}{t_j}\sum_{k=0}^{\infty}\left(\frac{s_i}{t_j}\right)^k. \quad (12)$$

Once again for large *M* the expression $-\frac{1}{t_j}\sum_{k=0}^{M}\left(\frac{s_i}{t_j}\right)$ approximate $\frac{1}{s_i-t_j}$.

The product of Cauchy matrix by a vector is $C(\boldsymbol{s},\boldsymbol{t})\boldsymbol{v}=\sum_{j=0}^{n-1}\left(\frac{v_j}{s_i-t_j}\right)_{i=0}^{n-1}$. This is just Trummer's problem. If we simplify this expression, we get the following approximations:

$$\frac{v_j}{s_i-t_j}\simeq\sum_{j=0}^{n-1}-\frac{1}{t_j}\sum_{k=0}^{M}\left(\frac{s_i}{t_j}\right)^k v_j=-\sum_{j=0}^{n-1}\left(\frac{v_j}{t_j}\right)\sum_{k=0}^{M}\left(\frac{s_i}{t_j}\right)^k=-\sum_{j=0}^{n-1}\left(\frac{v_j}{t_j}\right)\sum_{k=0}^{M}\frac{s_i^k}{t_j^k}=-\sum_{j=0}^{n-1}\frac{v_j}{t_j^{k+1}}\sum_{k=0}^{M}s_i^k=\sum_{k=0}^{M}z_k s_i^k$$

where $z_k = -\sum_{j=0}^{n-1} \frac{v_j}{t_j^{k+1}}$.

For any $n \times n$ Cauchy matrix $C(\boldsymbol{s},\boldsymbol{t})$, the approximation requires $O(nM)$ ops for all *i* and it is numerically stable.

If either of the ratios $\left|\frac{s_i}{t_j}\right|$ or $\left|\frac{t_j}{s_i}\right|$ is small, then the approximate error will be small for large *M*. However, there will be a problem whenever one of the ratios is close to 1, in this case, the error will be large.

8. Discussions and Conclusions

Recall the following two formulas from Section 5:

$$C(\boldsymbol{s},\boldsymbol{t}) = D(\boldsymbol{s},\boldsymbol{t})^{-1} D(\boldsymbol{s},\boldsymbol{q}) C(\boldsymbol{s},\boldsymbol{q}) D'(\boldsymbol{q})^{-1} D(\boldsymbol{q},\boldsymbol{t}) C(\boldsymbol{q},\boldsymbol{t}), \tag{13}$$

$$C(\boldsymbol{s},\boldsymbol{t}) = -C(\boldsymbol{s},\boldsymbol{q}) D(\boldsymbol{q},\boldsymbol{s}) D'(\boldsymbol{q})^{-1} C(\boldsymbol{q},\boldsymbol{t}) D(\boldsymbol{t},\boldsymbol{q}) D(\boldsymbol{t},\boldsymbol{s})^{-1}. \tag{14}$$

Equations (13) and (14) are Vandermonde-free and Hankel-free, but they enable us to transform the basis vectors ***s*** and ***t*** for $C(\boldsymbol{s},\boldsymbol{t})$ into the two pairs of basis vectors ***s***, ***q*** and ***q***, ***t*** for any choice of the vector $\boldsymbol{q} = (q_j)$, $q_j . \neq s_i$, $q_j \neq t_k$, $i, j, k = 0,\cdots,n-1$. Then Trummer's problem is reduced to the evaluation of the diagonal matrices $D'(\boldsymbol{q})^{-1}$, $D(\boldsymbol{u},\boldsymbol{v})$ and/or $D(\boldsymbol{u},\boldsymbol{v})^{-1}$ for $(\boldsymbol{s},\boldsymbol{t})$, $(\boldsymbol{q},\boldsymbol{t})$, $(\boldsymbol{s},\boldsymbol{q})$, $(\boldsymbol{q},\boldsymbol{s})$, $(\boldsymbol{t},\boldsymbol{q})$ and/or $(\boldsymbol{t},\boldsymbol{s})$ and also reduced to recursive multiplication of the above matrices and $C(\boldsymbol{q},\boldsymbol{t})$ and $C(\boldsymbol{s},\boldsymbol{q})$ by vectors.

To compute the matrices $D'(\boldsymbol{v})$, $D(\boldsymbol{u},\boldsymbol{v})$ and $D(\boldsymbol{u},\boldsymbol{v})^{-1}$ for given $(\boldsymbol{u},\boldsymbol{v})$ in general, we first compute the coefficients of the polynomial $p_u(x) = \prod_{j=0}^{n-1}(x - v_j)$ and then $p_v(u_i)$.

And $p'_u(v_i)$, $i = 0,\cdots,n-1$. We compute the coefficients by simply pairwise multiply the linear factors $x - v_j$ first and then, recursively, the computed products. The computation is numerically stable and uses $O(n\log^2 n)$ ops. Multipoint polynomial evaluation can be computed in $O(n\log^2 n)$ arithmetic operations (ops), but it is not numerically stable; therefore the fast and numerically stable approximation techniques of [10] can be used instead. If we choose any vector $\boldsymbol{q} = (q_i)_{i=0}^{n-1}$, we will simplify the evaluation of the matrices

$$D(\boldsymbol{u},\boldsymbol{v}),\ D(\boldsymbol{u},\boldsymbol{v})^{-1} \text{ and } D'(\boldsymbol{q}) \text{ where } \boldsymbol{u} = \boldsymbol{q} \text{ or } \boldsymbol{v} = \boldsymbol{q}.$$

For example, if

$$q_i = a w_n^i, \quad i = 0,1,\cdots,n-1 \tag{15}$$

is the scaled n^{th} roots of unity for a scalar *a* and $w_n = \exp(2\pi i/n)$, where $i = \sqrt{-1}$. Then

$$p_{\boldsymbol{q}}(x) = \prod_{i=0}^{n-1}(x - a w_n^i) = x^n - a^n, \quad p'_{\boldsymbol{q}}(x) = n x^{n-1}$$

and the matrices $D(\boldsymbol{u},\boldsymbol{q})$ and $D(\boldsymbol{q})$ can be immediately evaluated in $O(n\log n)$ flops. In addition, any polynomial $p(x)$ of degree *n* can be evaluated at the scaled n^{th} roots of 1 in $O(n\log n)$ ops by means of Fast Fourier Transform (FFT). Trummer's problem is the multiplication of $C(\boldsymbol{q},\boldsymbol{t})$ by a vector or $C(\boldsymbol{s},\boldsymbol{q})$ by a vector. Its solution can be simplified under appropriate choice of the vector ***q***. One way to do it is to restrict ***q*** to the above choice in Equation (15). Even with this particular choice, yet the scalar *a* allows faster convergence of the power series of the Multipole Algorithm presented in Section 7. This can be extended to the Equations (5) and (7). On the other hand, one can linearly map the vector ***q*** into $\boldsymbol{y} = a\boldsymbol{q} + b\boldsymbol{e}$, where $\boldsymbol{e} = (1)_{j=0}^{n-1}$, and $0 \neq a$ and *b* are any scalars. In addition, the computations of the diagonal matrices will be simplified if our choice of the vector ***q*** is the scaled nth root of unity.

Remark 8.1. Trummer's problem frequently arises for Cauchy degenerate matrices that are defined as follows: $C(\boldsymbol{s}) = (c_{i,j})$, $c_{i,i} = 0$, $c_{i,j} = \frac{1}{s_i - s_j}$ for all pairs of distinct *i* and *j*.

We have

$$C(\boldsymbol{s}) = \frac{1}{h}\sum_{g=0}^{h-1} C(\boldsymbol{s}, \boldsymbol{s} + \delta w_h^g \boldsymbol{e}) + O(\delta^h) \text{ as } \delta \to 0$$

where $\boldsymbol{e} = (1)_{j=0}^{n-1}$, $\boldsymbol{s} = (s_i)$, δ is a scalar parameter. Hence,

$$\sum_{g=0}^{h-1} \frac{1}{s_i - s_j - \delta w_h^g} = \frac{1}{s_i - s_j} \sum_{l=0}^{\infty} \sum_{g=0}^{h-1} \left(\frac{\delta w_h^g}{s_i - s_j} \right)^l = \frac{h}{s_i - s_j} \left(1 + O\left(\delta^h\right)\right)$$

because $\sum_{l=0}^{n-1} w_n^{gl} = 0$ for $g = 1, \cdots, n-1$.

Acknowledgements

We thank the Editor and referees for their valuable comments.

References

[1] Rokhlin, V. (1985) Rapid Solution of Integral Equations of Classical Potential Theory. *Journal of Computational Physics*, **60**, 187-207. http://dx.doi.org/10.1016/0021-9991(85)90002-6

[2] Gerasoulis, A. (1987) A Fast Algorithm for the Multiplication of Generalized Hilbert Matrices with Vectors. *Mathematics of Computation*, **50**, 179-188. http://dx.doi.org/10.1090/S0025-5718-1988-0917825-9

[3] Greengard, L. and Rokhlin, V. (1987) A Fast Algorithm for Practice Simulation. *Journal of Computational Physics*, **73**, 325-348. http://dx.doi.org/10.1016/0021-9991(87)90140-9

[4] Pan, V.Y., Zheng, A., Huany, X. and Yu, Y. (1995) Fast Multipoint Polynomial Evaluation and Interpolation via Computation with Structured Matrices. *Annals of Numerical Mathematics*, **4**, 483-510.

[5] Pan, V.Y. (2001) Structured Matrices and Polynomials, Unified Superfast Algorithms. Birkhäuser, Boston. http://dx.doi.org/10.1007/978-1-4612-0129-8

[6] Bini, D. and Pan, V.Y. (1994) Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms. Birkhäuser, Boston. http://dx.doi.org/10.1007/978-1-4612-0265-3

[7] Pan, V.Y., Tabanjeh, M., Chen, Z., Landowne, E. and Sadikou, A. (1998) New Transformations of Cauchy Matrices and Trummer's Problem. *Computers & Mathematics with Applications*, **35**, 1-5. http://dx.doi.org/10.1016/S0898-1221(98)00091-1

[8] Fink, T., Heinig, G. and Rost, K. (1993) An Inversion Formula and Fast Algorithms for Cauchy-Vandermonde Matrices. *Linear Algebra & Its Applications*, **183**, 179-191. http://dx.doi.org/10.1016/0024-3795(93)90431-M

[9] Pan, V.Y., Landowne, E., Sadikou, A. and Tiga, O. (1993) A New Approach to Fast Polynomial Interpolation and Multipoint Evaluation. *Computers & Mathematics with Applications*, **25**, 25-30. http://dx.doi.org/10.1016/0898-1221(93)90129-J

[10] Pan, V.Y. (1995) An Algebraic Approach to Approximate Evaluation of a Polynomial on a Set of Real Points. *Advances in Computational Mathematics*, **3**, 41-58. http://dx.doi.org/10.1007/BF02431995

American Journal of Computational Mathematics, 2014, 4, 396-405
Published Online December 2014 in SciRes. http://www.scirp.org/journal/ajcm
http://dx.doi.org/10.4236/ajcm.2014.45034

Hall Effects on Unsteady MHD Three Dimensional Flow through a Porous Medium in a Rotating Parallel Plate Channel with Effect of Inclined Magnetic Field

P. Sulochana

Intell Engineering College, Anantapuramu, Andhra Pradesh, India
Email: arigela.sulochana@gmail.com

Received 16 July 2014; revised 18 August 2014; accepted 3 September 2014

Abstract

In this paper, we make an initial value investigation of the unsteady flow of incompressible viscous fluid between two rigid non-conducting rotating parallel plates bounded by a porous medium under the influence of a uniform magnetic field of strength H_0 inclined at an angle of inclination α with normal to the boundaries taking hall current into account. The perturbations are created by a constant pressure gradient along the plates in addition to the non-torsional oscillations of the upper plate while the lower plate is at rest. The flow in the porous medium is governed by the Brinkman's equations. The exact solution of the velocity in the porous medium consists of steady state and transient state. The time required for the transient state to decay is evaluated in detail and the ultimate quasi-steady state solution has been derived analytically. Its behaviour is computationally discussed with reference to the various governing parameters. The shear stresses on the boundaries are also obtained analytically and their behaviour is computationally discussed.

Keywords

Hall Effects, Unsteady Rotating Flows, Three-Dimensional Flows, Parallel Plate Channels, Incompressible Viscous Fluids, Brinkman's Model

1. Introduction

The rotating flow between parallel plates is a classical problem that has important applications in magneto hydro dynamic (MHD) power generators and pumps, accelerators, aerodynamic heating, electrostatic precipitation polymer technology, petroleum industry, purification of crude oil and fluid droplets, sprays, designing cooling systems with liquid metal, centrifugal separation of matter from fluid and flow meters. The flows of fluids through porous medium are very important particularly in the fields of agricultural engineering for irrigation processes; in petroleum technology to study petroleum transport; in chemical engineering for filtration and purification processes. A series of investigations have been made by (Raptis *et al*., 1981 [1]; Raptis *et al*., 1981 [2], Raptis *et al*., 1982 [3]) into the steady of two-dimensional flow past a vertical wall for constant permeability of the porous medium. (Singh and Verma, 1995 [4]) analyzed an oscillatory three-dimensional flow through a porous medium when the permeability varied in space periodically. (Singh *et al*., 2000 [5]) investigated further a three-dimensional fluctuating flow and heat transfer through a porous medium when the permeability varied both in time and space. Further the flow of electrically conducting fluids in channels and pipes under the effect of transverse magnetic field occur in magnetohydrodynamic (MHD) generators, accelerators, pumps and flow meters. In view of these and many other important applications of these flows a number of scholars have shown their interest. Notable amongst them are (Shercliff, 1965 [6]; Ferraro and Plumpton, 1966 [7]; Crammer and Pai 1973, [8]). (Yen and Chang, 1964 [9]) studied the effects of wall electrical conductance on the MHD Couette flow. A magnetohydrodynamic (MHD) flow in a duct has also been studied by (Chang and Lundgren, 1961 [10]). (Attia and Kotb, 1996 [11]) investigated the two dimensional MHD flow between two porous, parallel, infinite, insulated, horizontal plates and the heat transfer through it when the lower plate was kept stationary and the upper plate was moving with uniform velocity. Very recently (Singh and Mathew, 2008 [12]) studied the injection/suction effect on a hydromagnetic oscillatory flow in a horizontal porous channel in a rotating system. The Hall current effect on the velocity and temperature fields of an unsteady Hartmann number has also been studied by (Attia, 2006 [13]). (Singh and Sharma, 2001 [14]) studied a three-dimensional Couette flow with transpiration cooling in the presence of stationary magnetic field applied perpendicular to the planes of the insulated plates. Another aspect of the above three-dimensional Couette flow when the magnetic field is fixed with the moving plate has also been investigated by (Singh, 2004 [15]). There are various other industrial applications of flows of electrically conducting fluids in the fields of geothermal systems, nuclear reactors, filtration, etc. where the conducting fluid flows through a porous medium which also rotates about an axis. In view of the importance of rotating flows a number of studies have appeared in the literature. (Mazumder, 1991 [16]) studied an oscillatory Ekman boundary layer flow bounded by two horizontal plates one of which is oscillating and the other is at rest. (Ganapathy, 1994 [17]) presented an alternative solution to the above problem. (Mazumder *et al*., 1976 [18]) analyzed the Hall effects on combined free and forced convection hydromagnetic flow through a channel. (Singh, 2000 [19]) studied the effects of transversely applied uniform magnetic field on oscillatory flow between two parallel flat plates when the entire system rotates about an axis normal to the planes of the plates. Hartman and Lazarus (1937 [20]) studied the influence of a transverse uniform magnetic field on the flow of a viscous incompressible electrically conducting fluid between two infinite parallel stationary and insulating plates. Then the problem was extended in numerous ways. The Hall current is important and it has a marked effect on the magnitude and direction of the current density and consequently on the magnetic force. The unsteady hydro magnetic viscous flow through a nonporous or porous medium has drawn attention in the recent years for possible applications in geophysical and cosmical fluid dynamics. Debnath *et al.* (1979 [21]) have studied the effects of Hall current on unsteady hydro magnetic flow past a porous plate in a rotating fluid system and the structure of the steady and unsteady flow fields is investigated. Rao and Krishna (1981 [22]) studied Hall effects on the non-torsionally generated unsteady hydro magnetic flow in semi-infinite expansion of an electrically conducting viscous rotating fluid. Krishna and Rao (1982 [23]) discussed the Stokes and Eckmann problems in magneto hydro dynamics taking Hall effects into account. M. VeeraKrishna and S. V. Suneetha (2009 [24]) discussed Hall effects on unsteady flow of incompressible viscous fluid between two rigid non-conducting rotating plates through porous medium under the influence of a uniform transverse magnetic field. S. V. Suneetha *et al.* (2010 [25]) discussed Hall effects on unsteady rotating magneto hydro dynamic flow of an incompressible homogeneous second grade fluid through a porous half space. Recently Hall effects on an unsteady MHD flow of a viscous incompressible electrically conducting fluid in a horizontal porous channel with variable pressure gradient in a rotating system have been studied by Sanatan Das and Rabindranath Jana (2013 [26]). In this paper, we make an initial value investigation of the unsteady flow of incompressible viscous fluid between two rigid non-conducting rotating parallel plates bounded by a porous medium taking hall current into account.

2. Formulation and Solution of the Problem

We consider the unsteady flow of an incompressible electrically conducting viscous fluid bounded by porous medium with two non-conducting rotating parallel plates. A uniform transverse magnetic field is applied to *z*-axis. In the presence of strong magnetic field a current is inclined in a direction normal to the both electric and magnetic field viz. Magnetic field of strength H_0 inclined at angle of inclination α to the normal to the boundaries in the transverse *xz*-plane. The inclined magnetic field gives rise to a secondary flow transverse to the channel. The hydro magnetic flow is generated in a fluid system by non-torsional oscillations of the upper plate. The lower plate is at rest. The origin is taken on the lower plate and the *x*-axis parallel to the direction of the upper plate. Since the plates are infinite in extent, all the physical quantities except the pressure depend on *z* and *t* only. In the equation of motion along *x*-direction, the *x*-component current density—$\mu_e J_z H_o \sin\alpha$ and the *z*-component current density $\mu_e J_x H_o \sin\alpha$. We choose a Cartesian system 0 (*x*, *y*, *z*) such that the boundary walls are at *z* = 0 and *z* = l. The flow through porous medium governed by the Brinkman equations. The unsteady hydro magnetic equations governing flow through porous medium under the influence of a transverse magnetic field with reference to a rotating frame are

$$\frac{\partial u}{\partial t}+2\Omega w=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\frac{\mathrm{d}^2u}{\mathrm{d}z^2}-\frac{\mu_e J_z H_o\sin\alpha}{\rho}-\frac{\nu}{k}u \tag{1}$$

$$\frac{\partial w}{\partial t}-2\Omega u=\nu\frac{\mathrm{d}^2w}{\mathrm{d}z^2}+\frac{\mu_e J_x H_o\sin\alpha}{\rho}-\frac{\nu}{k}w \tag{2}$$

where, (*u*, *w*) is the velocity components along *O* (*x*, *z*) directions respectively. ρ is the density of the fluid, μ_e is the magnetic permeability, ν is the coefficient of kinematic viscosity, *k* is the permeability of the medium, H_o is the applied magnetic field. When the strength of the magnetic field is very large, the generalized Ohm's law is modified to include the Hall current, so that

$$J+\frac{\omega_e\tau_e}{H_0}J\times H=\sigma\left(E+\mu_e q\times H\right) \tag{3}$$

where, *q* is the velocity vector, *H* is the magnetic field intensity vector, *E* is the electric field, *J* is the current density vector, ω_e is the cyclotron frequency, τ_e is the electron collision time, σ is the fluid conductivity and, μ_e is the magnetic permeability. In Equation (3) the electron pressure gradient, the ion-slip and thermoelectric effects are neglected. We also assume that the electric field *E* = 0 under assumptions reduces to

$$J_x-mJ_z\sin\alpha=-\sigma\mu_e H_0\sin\alpha w \tag{4}$$

$$J_z+mJ_x\sin\alpha=\sigma\mu_e H_0\sin\alpha u \tag{5}$$

where $m=\omega_e\tau_e$ is the Hall parameter.

On solving Equations (4) and (5) we obtain

$$J_x=\frac{\sigma\mu_e H_0\sin\alpha}{1+m^2\sin^2\alpha}\left(mu\sin\alpha-w\right) \tag{6}$$

$$J_z=\frac{\sigma\mu_e H_0\sin\alpha}{1+m^2\sin^2\alpha}\left(u+mw\sin\alpha\right). \tag{7}$$

Using the Equations (6) and (7), the equations of the motion with reference to rotating frame are given by

$$\frac{\partial u}{\partial t}+2\Omega w=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\frac{\mathrm{d}^2u}{\mathrm{d}z^2}+\frac{\sigma\mu_e^2H_0^2\sin^2\alpha}{\rho\left(1+m^2\sin^2\alpha\right)}\left(u+mw\sin\alpha\right)-\frac{\nu}{k}u \tag{8}$$

$$\frac{\partial w}{\partial t}-2\Omega u=\nu\frac{\mathrm{d}^2w}{\mathrm{d}z^2}-\frac{\sigma\mu_e^2H_0^2\sin^2\alpha}{\rho\left(1+m^2\sin^2\alpha\right)}\left(mu\sin\alpha-w\right)-\frac{\nu}{k}w. \tag{9}$$

By combining the Equations (8) and (9), we get.

Let $q=u+iw$,

$$\frac{\partial q}{\partial t}-2iK^2q=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\frac{\partial^2 q}{\partial z^2}-\frac{\sigma\mu_e^2H_o^2\sin^2\alpha}{\rho(1+im\sin\alpha)}q-\frac{\nu}{k}q. \tag{10}$$

The boundary and initial conditions are

$$q=0,\ t\le 0,\ z=0 \tag{11}$$

$$q=a\mathrm{e}^{i\omega t}+b\mathrm{e}^{-i\omega t},\ t>0,\ z=l. \tag{12}$$

We introduce the following non dimensional variables are

$$z^*=\frac{z}{l},\ q^*=\frac{ql}{\nu},\ t^*=\frac{t\nu}{l^2},\ \omega^*=\frac{\omega l^2}{\nu},\ \xi^*=\frac{\xi}{l},\ p^*=\frac{pl^2}{\rho\nu^2}.$$

Using non-dimensional variables, the governing equations are (dropping asterisks)

$$\frac{\partial q}{\partial t}=-\frac{\partial p}{\partial x}+\frac{\partial^2 q}{\partial z^2}-\frac{M^2\sin^2\alpha}{(1+im\sin\alpha)}q-D^{-1}q \tag{13}$$

where,

$M^2=\dfrac{\sigma\mu_e^2H_0^2l^2}{\rho\nu}$ is the Hartmann number;

$K^2=\dfrac{\Omega^2l^2}{\nu}$ is the rotation parameter;

$D^{-1}=\dfrac{l^2}{k}$ is inverse Darcy parameter and;

$m=\omega_e\tau_e$ is the Hall parameter.

We choose $\dfrac{\partial p}{\partial x}=P_0+P_1\mathrm{e}^{i\omega_1 t}$ is the prescribed of pressure gradient, then the Equation (13) reduces to

$$\frac{\partial q}{\partial t}=-\left(P_0+P_1\mathrm{e}^{i\omega_1 t}\right)+\frac{\partial^2 q}{\partial z^2}-\left(\frac{M^2\sin^2\alpha}{(1+im\sin\alpha)}+D^{-1}+2iK^2\right)q. \tag{14}$$

Corresponding initial and boundary conditions are

$$q=0,\ t\le 0,\ z=0 \tag{15}$$

$$q=a\mathrm{e}^{i\omega t}+b\mathrm{e}^{-i\omega t},\ t>0,\ z=1. \tag{16}$$

Taking Laplace transform of Equation (14) using initial condition (15) the governing equations in terms of the transformed variable reduces to

$$\frac{\mathrm{d}^2\bar{q}}{\mathrm{d}z^2}-\left(\frac{M^2\sin^2\alpha}{(1+im\sin\alpha)}+D^{-1}+2iK^2+s\right)\bar{q}=-\frac{P_0}{s}-\frac{P_1}{s-i\omega_1}. \tag{17}$$

The relevant transformed boundary conditions are

$$\bar{q}=0,\ z=1, \tag{18}$$

$$\bar{q}=\frac{a}{s-i\omega}+\frac{b}{s+i\omega},\ z=0. \tag{19}$$

Solving the Equation (17) and making use of the boundary conditions (18) and (19), we obtain

$$\bar{q}=A\cosh\lambda_1 z+B\sinh\lambda_1 z+\frac{P_0}{\lambda_1^2 s}+\frac{P_1}{\lambda_1^2(s-i\omega_1)} \tag{20}$$

where

$$A=-\frac{P_0}{\lambda_1^2 s}-\frac{P_1}{\lambda_1^2\left(s-i\omega_1\right)},$$

$$B=-\frac{1}{\sinh\lambda_1}\left[\frac{a}{s-i\omega}+\frac{b}{s+i\omega}-\frac{P_0}{\lambda_1^2 s}-\frac{P_1}{\lambda_1^2\left(s-i\omega_1\right)}+\left(\frac{P_0}{\lambda_1^2 s}+\frac{P_1}{\lambda_1^2\left(s-i\omega_1\right)}\right)\cosh\lambda_1\right]$$

$$\lambda_1=\sqrt{s+\left(\frac{M^2\sin^2\alpha}{\left(1+im\sin\alpha\right)}+D^{-1}+2iK^2\right)}.$$

Taking inverse Laplace transform to the Equation (20), we obtain

$$\begin{aligned}q=&\frac{P_0}{a_1}-\frac{P_0\sinh\sqrt{a_1}z}{a_1\sinh\sqrt{a_1}}+\frac{P_0\cosh\sqrt{a_1}\sinh\sqrt{a_1}z}{a_1\sinh\sqrt{a_1}}+a\frac{\sinh\sqrt{a_2}z}{\sinh\sqrt{a_2}}\mathrm{e}^{i\omega t}+b\frac{\sinh\sqrt{a_3}z}{\sinh\sqrt{a_3}}\mathrm{e}^{-i\omega t}\\&+\left(-\frac{P_0\cosh\sqrt{a_1}z}{a_1}-\frac{P_1}{a_1+i\omega_1}\frac{\sinh\sqrt{a_1}z}{\sinh\sqrt{a_1}}+\frac{P_1}{a_1+i\omega_1}\frac{\cosh\sqrt{a_1}\sinh\sqrt{a_1}z}{\sinh\sqrt{a_1}}\right)\mathrm{e}^{-a_1t}\\&+\left(-\frac{P_0\cosh\sqrt{a_4}z}{a_4}+\frac{P_1}{a_4}\frac{\sinh\sqrt{a_4}z}{\sinh\sqrt{a_4}}-\frac{P_1}{a_4}\frac{\cosh\sqrt{a_4}\sinh\sqrt{a_4}z}{\sinh\sqrt{a_4}}+\frac{P_1}{a_4}\right)\mathrm{e}^{i\omega_1t}\\&+\sum_{n=1}^{\infty}\left[-\frac{az}{n^2\pi^2+a_2}+\frac{bz}{n^2\pi^2+a_3}+\frac{P_0z\left(\cos n\pi-1\right)}{n^2\pi^2\left(n^2\pi^2+a_1\right)}+\frac{P_1z\left(\cos n\pi-1\right)}{n^2\pi^2\left(n^2\pi^2+a_1+i\omega_1\right)}\right]\mathrm{e}^{-\left(a_1+n^2\pi^2\right)t}\end{aligned}\tag{21}$$

$$a_1=\frac{M^2\sin^2\alpha}{\left(1+im\sin\alpha\right)}+D^{-1}+2iK^2,\ a_2=\frac{M^2\sin^2\alpha}{\left(1+im\sin\alpha\right)}+D^{-1}+2iK^2+i\omega,$$

$$a_3=\frac{M^2\sin^2\alpha}{\left(1+im\sin\alpha\right)}+D^{-1}+2iK^2-i\omega\ \text{ and }\ a_4=\frac{M^2\sin^2\alpha}{\left(1+im\sin\alpha\right)}+D^{-1}+2iK^2+i\omega_1.$$

The shear stresses on the upper plate and the lower plate are given by

$$\tau_U=\left(\frac{\mathrm{d}q}{\mathrm{d}z}\right)_{z=1}\ \text{ and }\ \tau_L=\left(\frac{\mathrm{d}q}{\mathrm{d}z}\right)_{z=0}.\tag{22}$$

3. Results and Discussion

The flow is governed by the non-dimensional parameters M the Hartman number, D^{-1} the inverse Darcy parameter, K is the rotation parameter and m is the Hall parameter. The velocity field in the porous region is evaluated analytically its behaviour with reference to variations in the governing parameters has been computationally analyzed. The profiles for u and w have been plotted in the entire flow field in the porous medium. The solution for the velocity consists of three kinds of terms 1) steady state, 2) the quasi-steady state terms associated with non-torsional oscillations in the boundary, 3) the transient term involving exponentially varying time dependence. From the expression (21), it follows that the transient component in the velocity in the fluid region decays in dimensionless time $t>\max\left\{\frac{1}{|a_1|},\frac{1}{|a_1+n^2\pi^2|}\right\}$. When the transient terms decay the steady oscillatory solution in the fluid region is given by

$$(q)_{\text{steady}}=\frac{P}{a_1}-\frac{P\sinh\sqrt{a_1}z}{a_1\sinh\sqrt{a_1}}+\frac{P\cosh\sqrt{a_1}\sinh\sqrt{a_1}z}{a_1\sinh\sqrt{a_1}}\tag{23}$$

$$(q)_{\text{oscillatory}}=a\frac{\sinh\sqrt{a_2}z}{\sinh\sqrt{a_2}}\mathrm{e}^{i\omega t}+b\frac{\sinh\sqrt{a_3}z}{\sinh\sqrt{a_3}}\mathrm{e}^{-i\omega t}.\tag{24}$$

We now discuss the quasi steady solution for the velocity for different sets of governing parameters namely viz. M the Hartman number and D^{-1} the inverse Darcy parameter, K the rotation parameter, m is the Hall parameter, P_0 & P_1 the non dimensional pressure gradients, the frequency oscillations ω, a and b the constants related to non torsional oscillations of the boundary, for computational analysis purpose we are fixing the axial pressure gradient as well as a and b, and $P_0 = P_1 = 10$, $\omega = \pi/4$, $\omega_1 = \pi/4$, $\alpha = \pi/3$. **Figures 1-8** corresponding to the velocity components u and w along the prescribed pressure gradient for different sets of governing parameters when the upper boundary plate executes non-torsional oscillations. The magnitude of the velocity u and w increases for the sets of values $0.1 \le z \le 0.3$ as well as which reduces for all values of z with increase in the intensity of the magnetic field (**Figure 1** and **Figure 5**). The resultant velocity q decreases with increasing the Hartmann number M. The magnitude of the velocity u decreases in the upper part of the fluid region $0.1 \le z \le 0.2$ while it experiences enhancement lower part $0.3 \le z \le 0.9$ with increasing the inverse Darcy parameter D^{-1} (**Figure 2**). The magnitude of the velocity w increases in the upper part of the fluid region $0.1 \le z \le 0.3$, while it reduces in lower part $0.4 \le z \le 0.9$ with increasing the inverse Darcy parameter D^{-1} (**Figure 6**). The resultant velocity q reduces with increasing the inverse Darcy parameter D^{-1}. The magnitude of velocity u decreases in the upper part of the fluid region while it experiences enhancement lower part $0.3 \le z \le 0.9$ and also the magnitude of velocity w increases throughout the fluid region (**Figure 3** and **Figure 7**). However the resultant velocity q enhances with increasing the Hall parameter m. Finally we notice that, from (**Figure 4** and **Figure 8**) the magnitude of the velocity component enhances for $0.1 \le z \le 0.3$ and $z = 0.7$, and reduces within the region $0.4 \le z \le 0.6$ and $0.8 \le z \le 0.9$ with increase in rotation parameter K. while the velocity component w enhances for $0.3 \le z \le 0.4$ and $z = 0.9$, and reduces for $0.1 \le z \le 0.2$, with increase in rotation parameter K.

The shear stresses τ_x and τ_y on the upper plate have been calculated for the different variations in the governing parameters and are tabulated in the **Table 1**, **Table 2**. On the upper plate we notice that the magnitudes of τ_x enhances the inverse Darcy parameter D^{-1}, the hall parameter m, rotation parameter K decreases with increase in the Hartmann number M (**Table 1**). The magnitude of τ_y decreases with increase in the Hartmann number M, the inverse Darcy parameter D^{-1} rotation parameter K and the Hall parameter m fixing the other parameters (**Table 2**). The similar behaviour is observed on the lower plate (**Table 3**, **Table 4**). We also notice that the magnitude of the shear stresses on the lower plate is very small compare to its values of the upper plate.

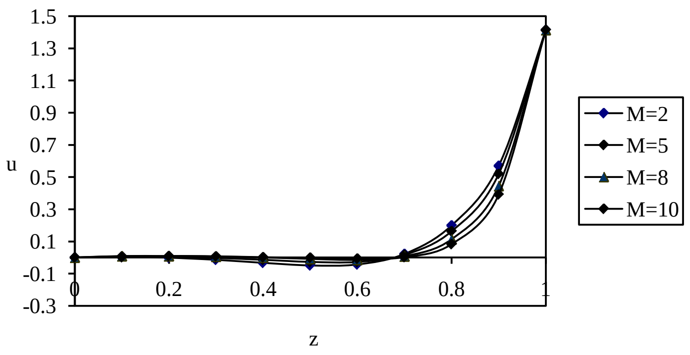


Figure 1. The velocity profile for u with M.

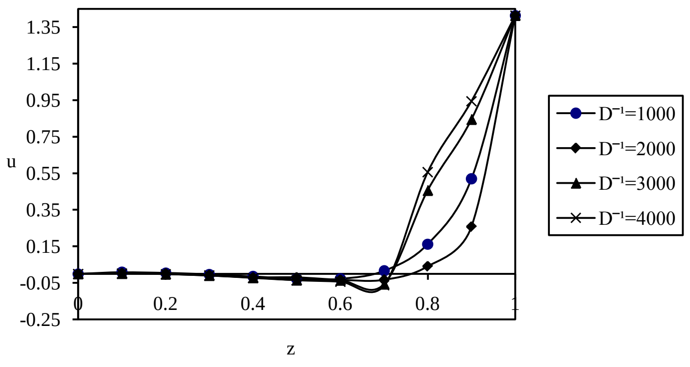


Figure 2. The velocity profile for u with D^{-1}.

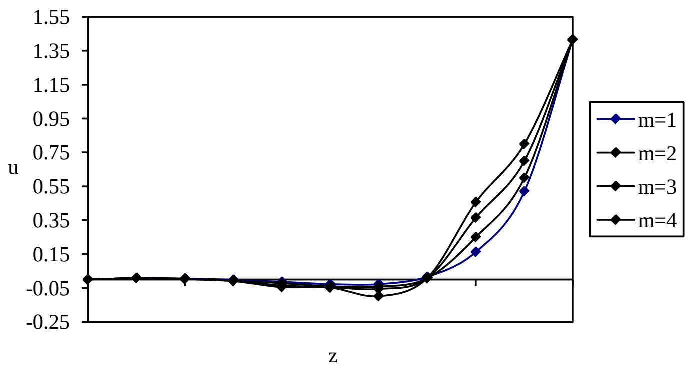


Figure 3. The velocity profile for *u* with *m*.

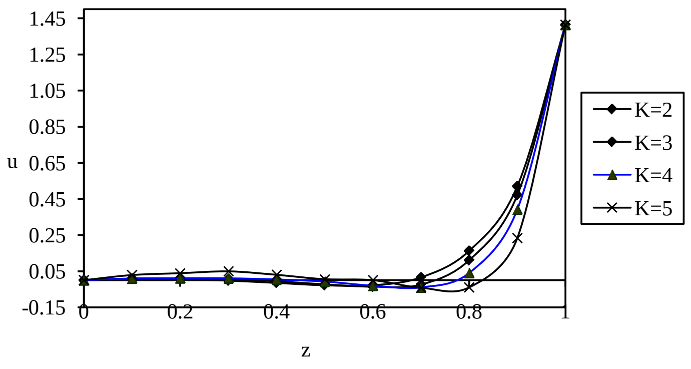


Figure 4. The velocity profile for *u* with *K*.

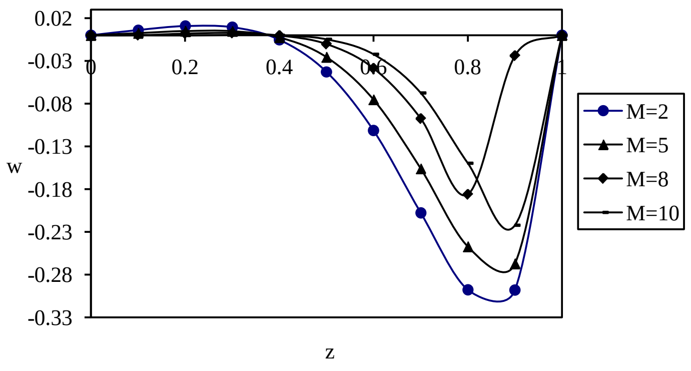


Figure 5. The velocity profile for *w* with *M*.

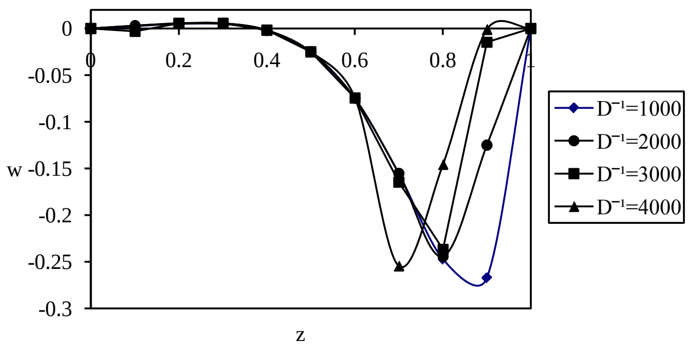


Figure 6. The velocity profile for *w* with D^{-1}.

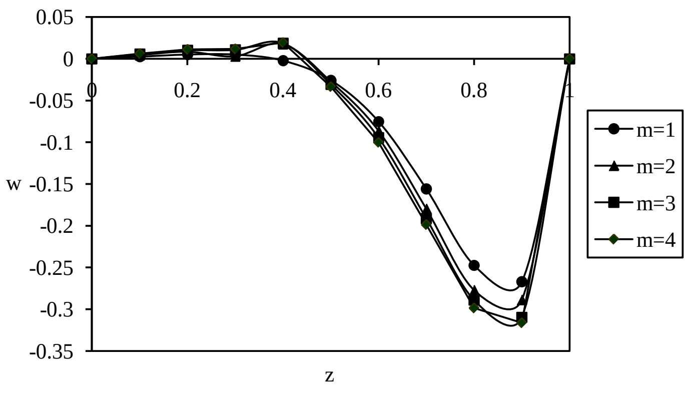


Figure 7. The velocity profile for w with m.

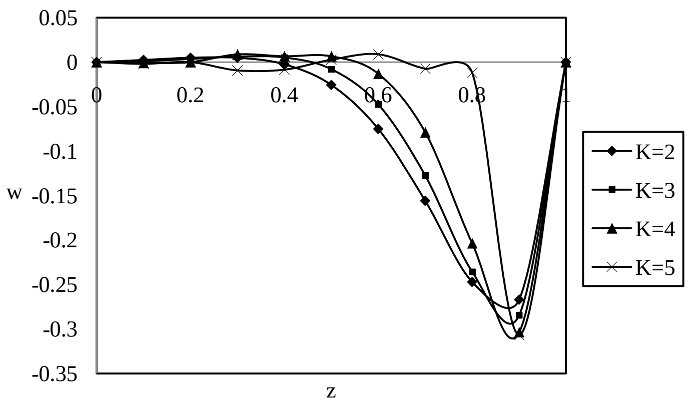


Figure 8. The velocity profile for w with K.

Table 1. The shear stress (τ_x) on the upper plate.

M	I	II	III	IV	V	VI	VII
2	0.045274	0.052798	0.668876	0.052787	0.065525	0.084474	0.144589
5	0.032905	0.043535	0.050487	0.043465	0.051896	0.052248	0.125547
D^{-1}	1000	2000	3000	1000	1000	1000	1000
m	1	1	1	2	3	1	1
K	2	2	2	2	2	3	4

Table 2. The shear stress (τ_y) on the upper plate.

M	I	II	III	IV	V	VI	VII
2	−0.05356	−0.040556	−0.03558	−0.04955	−0.32511	−0.041125	−0.0044585
5	−0.04555	−0.034255	−0.02622	−0.03512	−0.02222	−0.024451	−0.0001254
D^{-1}	1000	2000	3000	1000	1000	1000	1000
m	1	1	1	2	3	1	1
K	2	2	2	2	2	3	4

Table 3. The shear stress (τ_x) on the lower plate.

M	I	II	III	IV	V	VI	VII
2	0.008554	0.005542	0.002554	0.006658	0.003325	0.000144	−0.104595
5	0.007885	0.004102	0.001001	0.005114	0.002114	0.000025	−0.002852
D^{-1}	1000	2000	3000	1000	1000	1000	1000
m	1	1	1	2	3	1	1
K	2	2	2	2	2	3	4

Table 4. The shear stress (τ_y) on the lower plate.

M	I	II	III	IV	V	VI	VII
2	−0.000255	−0.000149	−0.000025	−0.000228	−0.000187	−0.0000145	−0.0000054
5	−0.000246	−0.000124	−0.000012	−0.000193	−0.000078	−0.0000102	−0.0000029
D^{-1}	1000	2000	3000	1000	1000	1000	1000
m	1	1	1	2	3	1	1
K	2	2	2	2	2	3	4

4. Conclusions

1) The resultant velocity q enhances with increasing hall parameter m and rotation parameter K, and decreases with increasing inverse Darcy parameter D^{-1} as well as the Hartmann number M.

2) On the upper plate the magnitude of τ_x enhances when increasing the hall parameter m; rotation parameter K and the inverse Darcy parameter D^{-1} decrease with increase in the Hartmann number M.

3) On the upper plate the magnitude of shear stress enhances when increasing the hall parameter M; rotation parameter K and the inverse Darcy parameter D^{-1} decrease with increase in the Hartmann number M.

4) The similar behaviour is observed on the lower plate.

5) The magnitude of the shear stresses on the lower plate is very small than the values of the upper plate.

References

[1] Raptis, A., Perdikis, C. and Tzivanidis, G. (1981) Free Convection Flow through a Porous Medium Bounded by a Vertical Surface. *Journal of Physics D*: *Applied Physics*, **14**, 99-102. http://dx.doi.org/10.1088/0022-3727/14/7/001

[2] Raptis, A., Tzivanidis, G. and Kafousias, N. (1981) Free Convection and Mass Transfer Flow through a Porous Medium Bounded by an Infinite Vertical Limiting Surface with Constant Suction. *Letters Heat Mass Transfer*, **8**, 417-424. http://dx.doi.org/10.1016/0094-4548(81)90029-1

[3] Raptis, A., Kafousias, N. and Massalas, C. (1982) Free Convection and Mass Transfer Flow through a Porous Medium Bounded by an Infinite Vertical Porous Plate with Constant Heat Flux. *Zeitschrift für Angewandte Mathematik und Mechanik*, **62**, 489-491. http://dx.doi.org/10.1002/zamm.19820620911

[4] Singh, K.D. and Verma, G.N. (1995) Three-Dimensional Oscillatory Flow through a Porous Medium with Periodic Permeability. *Zeitschrift für Angewandte Mathematik und Mechanik*, **75**, 599-604. http://dx.doi.org/10.1002/zamm.19950750811

[5] Singh, K.D., Sharma, R. and Chand, K. (2000) Three-Dimensional Fluctuating Flow and Heat Transfer through a Porous Medium with Variable Permeability. *Zeitschrift für Angewandte Mathematik und Mechanik*, **80**, 473-480. http://dx.doi.org/10.1002/1521-4001(200007)80:7<473::AID-ZAMM473>3.0.CO;2-1

[6] Shercliff, T.A. (1965) A Text-Book of Magneto Hydro Dynamics. Pergamon Press, London.

[7] Ferraro, V.C.A. and Plumpton, C. (1966) An Introduction to Magneto Fluid Mechanics. Clarandon Press, Oxford.

[8] Crammer, K.P. and Pai, S.I. (1973) Magneto Fluid Dynamics for Engineers and Applied Physicist. McGraw Hill Book

Co., New York.

[9] Yan, J.T. and Chang, C.C. (1964) Magneto Hydrodynamic Couette Flow as Affected by Wall Electrical Conductance. *Zeitschrift für Angewandte Mathematik und Physik*, **15**, 400-407. http://dx.doi.org/10.1007/BF01601291

[10] Chang, C.C. and Lundgren, T.S. (1961) Duct Flow in Magneto Hydrodynamics. *Zeitschrift für Angewandte Mathematik und Physik*, **12**, 100-114. http://dx.doi.org/10.1007/BF01601011

[11] Attia, H.A. and Kotb, N.A. (1996) MHD Flow between Two Parallel Plates with Heat Transfer. *Acta Mechanica*, **117**, 215-220. http://dx.doi.org/10.1007/BF01181049

[12] Singh, K.D. and Alphonsa, M. (2008) Injection/Suction Effects on an Oscillatory Hydromagnetic Flow in a Rotating Horizontal Porous Channel. *Indian Journal of Physics*, **82**, 435-445.

[13] Attia, H.A. (2006) Time Varying Hydro Magnetic Couette Flow with Heat Transfer of a Dusty Fluid in the Presence of Uniform Suction and Injection Considering the Hall Effect. *Turkish Journal of Engineering and Environmental Sciences*, **30**, 285-297.

[14] Singh, K.D. and Rakesh, S. (2001) MHD Three-Dimensional Couette Flow with Transpiration Cooling. *Zeitschrift für Angewandte Mathematik und Mechanik*, **81**, 715-720. http://dx.doi.org/10.1002/1521-4001(200110)81:10<715::AID-ZAMM715>3.0.CO;2-A

[15] Singh, K.D. (2004) Influence of Moving Magnetic Field on Three-Dimensional Couette Flow. *Zeitschrift für Angewandte Mathematik und Physik*, **55**, 894-902. http://dx.doi.org/10.1007/s00033-004-2065-8

[16] Mazumder, B.S. (1991) An Exact Solution of Oscillatory Couette Flow in a Rotating System. *Journal of Applied Mechanics*, **58**, 1104-1107. http://dx.doi.org/10.1115/1.2897694

[17] Ganapathy, R.A. (1994) A Note on Oscillatory Couette Flow in a Rotating System. *Journal of Applied Mechanics*, **61**, 208-209. http://dx.doi.org/10.1115/1.2901403

[18] Mazumder, B.S., Gupta, A.S. and Datta, N. (1976) Hall Effects on Combined Free and Forced Convective Hydromagnetic Flow through a Channel. *International Journal of Engineering Science*, **14**, 285-292. http://dx.doi.org/10.1016/0020-7225(76)90045-8

[19] Singh, K.D. (2000) An Oscillatory Hydromagnetic Couette Flow in a Rotating System. *Zeitschrift für Angewandte Mathematik und Mechanik*, **80**, 429-432. http://dx.doi.org/10.1002/1521-4001(200006)80:6<429::AID-ZAMM429>3.0.CO;2-1

[20] Hartmann, J. and Lazarus, F. (1937) Kongelige danske videnskabernes selskab. *Matematisk-Fysiske Meddelelser*, **15**, 6-7.

[21] Debnath, L., Ray, S.C. and Chatterjee, A.K. (1979) Effects of Hall Current on Unsteady Hydromagnetic Flow past a Porous Plate in a Rotating Fluid System. *Zeitschrift für Angewandte Mathematik und Mechanik*, **59**, 469-471. http://dx.doi.org/10.1002/zamm.19790590910

[22] Prasada Rao, D.R.V. and Krishna, D.V. (1981) Hall Effect on Unsteady Hydromagnetic Flow. *Indian Journal of Pure and Applied Mathematics*, **12**, 270-276.

[23] Prasada Rao, D.R.V., Krishna, D.V. and Debnath. L. (1982) Hall Effects on Free and Forced Convective Flow in a Rotating Channel. *Acta Mechanica*, **43**, 49-59. http://dx.doi.org/10.1007/BF01175815

[24] Veera Krishna, M. and Suneetha, S.V. (2009) Hall Effects on Unsteady MHD Rotating Flow of an Incompressible Viscous Fluid through a Porous Medium. *Journal of Pure and Applied Physics*, **21**, 143-156.

[25] Suneetha, S.V., Veera Krishna, M. and Siva Pradad, R. (2010) Hall Effect on Unsteady Rotating Hydro Dynamic Flow of an Incompressible Second Grade Fluid in a Porous Half Space. *Journal of Pure and Applied Physics*, **22**, 143-156.

[26] Das, S., Mandal, H.K. and Jana, R.N. (2013) Hall Effect on Unsteady MHD Flow through a Porous Channel in a Rotating System with Variable Pressure Gradient. Ph.D. Thesis, Vidya Sagar University, Midnapur, 1-23.

American Journal of Computational Mathematics, 2014, 4, 406-413
Published Online December 2014 in SciRes. http://www.scirp.org/journal/ajcm
http://dx.doi.org/10.4236/ajcm.2014.45035

Statistical Modelling of Soybean Crop Yield in Regions of Central India through Mathematical and Computational Approach

Sarvraj Singh[1], Dilpreet Tuteja[1], Param Tripathi[1], Chirag Basavaraj[2]

[1]Jaypee University of Engineering & Technology, Guna, India
[2]R.V. College of Engineering, Bangalore, India
Email: sarvraj.5@hotmail.com

Received 7 August 2014; revised 15 September 2014; accepted 1 October 2014

Abstract

In this paper, we have discussed a number of fitting methods to predict crop yield of soybean depending on the nature of environment and a comparison is done between them on the basis of available data set. Later we have suggested a suitable method for the prediction of the crop yield on the basis of residual (error) terms. Statistical analysis is also used for getting the relationships between different components (variables) of available data set. At last, we have discussed about Chaos that can distort the whole mathematical analysis and a computational approach.

Keywords

Climate Change, Prediction, Chaos, Uncertainty

1. Introduction

Climate describes the ensemble sum of typical conditions of temperature, relative humidity, cloudiness, precipitation, wind speed and direction and innumerable other meteorological factors that prevail regionally for extended periods [1]. Weather of a demographic region is defined by the hourly description of the climatic conditions experienced by the inhabitants of that region. Here we discuss the soybean yield as a function of these environmental parameters.

Many different approaches are used for constraining climate based crop yield predictions based on observations of past empirical change in the yield [2]. Here we setup distinct models based on the environmental model parameters; significant correlations are calculated based on the inferred outputs. Meteorologists say that if only they could design an accurate mathematical model of the atmosphere with all its complexities, they could forecast the weather with real precision. But this is an idle boast, immune to any evaluation, for any inadequate weather forecast would obviously be blamed on imperfections in the model. Catering to the often glitches in the models prepared the fidelity of the dynamics governing the respective models can be doubted. With the introduction of computer simulations the weather predictions can be done in just a few minutes. We make use of such a technique to generalize the crop yield, and make prediction on the basis of the environmental factors like wind speed, wind direction, temperature and humidity. These factors are trivial when considering crop yield however, makes a difference as suggested by the models ahead.

Since the sensors of the parameters mentioned above are respect to one region in Central India, so we consider the crop that this region has lavishly produced, soybean. Soybean is one of the important crops of the world [3]. In India the production of soybean is currently restricted to mainly Madhya Pradesh, Uttar Pradesh, Maharashtra and Gujarat. Himachal Pradesh, Punjab and Delhi are other states with some marginal produce. According to 2010 estimates of soybean production India produces 4.4% of the total production; central India is the largest contributor of soybean yield. This brings us to concentrate more over this region for our fitting models.

Soybean is a crop that grows in warm and moist climate. An optimum yield requires a temperature ranging between 26.5˚C to 30˚C. For rapid germination and vigorous seedling growth soil temperatures of 15.5˚C or above are most suitable. A lower temperature delays flowering. Although, moisture enhances the yield of the crop but excess of moisture can make it prone to foliar diseases like frogeye leafs spot and septoria brown spot. Therefore, an optimum amount of humidity is required for the crop.

Wind direction and velocity also have a significant influence on crop growth [4]. While it has a few benefits, gusty winds blowing in one direction can harm the crop. Beneficial impacts include increasing the supply of carbon dioxide by increasing turbulence in the atmosphere. It also alters the balance of hormones. Strong winds in a region may uproot the crop or be an inevitable carrier of dispersive seeds that may hamper the yield. **Table 1** elucidates the conditions prevalent in Central India, state of Madhya Pradesh that monitor the soybean growth.

As far as the prediction of the yield on a larger perspective is considered, the simulations carried out by supercomputers are based on curve fitting methods. Curve fitting is the process of constructing a curve that has the best fit to a series of data points, possibly subject to constraints. Curve fitting involves interpolation [5], where an exact fit to the data is required in which a “smooth” function is constructed that approximately fits the data. A related topic is regression analysis, which focuses more on questions of statistical inference which includes the uncertainty present due to the random errors in the observed data. Fitted curves can be used as approximate data visualization for a model to which it is applied and to summarize the relationships among two or more variables.

Table 1. Varying environmental parameters dependent for yield of soybean in Central India.

Month	Temperature (X)	Humidity (Y)	Wind speed (Z)	Wind direction (W)
January	142.188	14.992	50.708	4.510
February	159.590	19.230	34.500	4.610
March	191.820	25.080	13.190	4.486
April	202.441	30.287	17.602	5.180
May	252.712	33.287	17.541	5.503
June	255.880	32.440	56.000	9.268
July	238.640	28.380	64.650	6.770
August	245.000	24.100	81.910	4.708
September	203.300	25.210	71.035	4.000
October	143.916	25.330	32.250	5.267
November	148.460	20.600	31.234	3.065
December	159.660	17.970	40.830	2.972

Extrapolation refers to the use of a fitted curve beyond the range of the observed data, and is subject to a greater degree of uncertainty since it may reflect the method used to construct the curve as much as it reflects the observed data. In order to fit a polynomial up to three degree which exactly fits four constraints, each constraint can be a point, angle, or curvature (which is the reciprocal of the radius of an osculating circle). Angle and curvature constraints are most often added to the ends of a curve, and in such cases are called end conditions. Identical end conditions are frequently used to ensure a smooth transition between polynomial curves contained within a single spline. If we have more than $n + 1$ constraints (n is the degree of the polynomial), we can still run the polynomial curve through those constraints. An exact fit to all constraints is not certain (but it might happen, for example, in the case of a first degree polynomial exactly fitting three collinear points). In general, however, some method is then needed to evaluate each approximation. The least squares method is one way to compare the deviations.

Low-order polynomials tend to be smooth and high order polynomial curves tend to be lumpy. To define this more precisely, the maximum number of inflection points possible in a polynomial curve is $n-2$, where n is the order of the polynomial equation. An inflection point is a location on the curve where it switches from a positive radius to negative. It is only possible that high order polynomials will be lumpy; they could also be smooth, but there is no guarantee of this, unlike with low order polynomial curves. A fifteenth degree polynomial could have, at most, thirteen inflection points, but could also have twelve, eleven, or any number down to zero.

2. Fitting a Polynomial Function

When a given set of data does not appear to satisfy a linear equation, we can try a suitable polynomial as a regression curve to fit data. The least squares technique can be readily used to fit the data to a polynomial.

Consider a polynomial of degree $m-1$

$$y = a_1 + a_2 x + a_3 x^2 + \cdots + a_m x^{m-1} = f(x). \tag{1}$$

If the data contains n sets of x and y values, then the sum of squares of the errors is given by

$$Q = \sum_{i=1}^{n} \left[y_i - f(x_i) \right]^2. \tag{2}$$

Since $f(x)$ is a polynomial and contains coefficients a_1, a_2, a_3 etc. we have to estimate all m coefficients. As before, we have the following m equations that can be solved for these coefficients.

$$\frac{\partial Q}{\partial a_1} = 0,$$
$$\frac{\partial Q}{\partial a_2} = 0,$$
$$\cdots$$
$$\cdots$$
$$\frac{\partial Q_m}{\partial a_m} = 0.$$

Consider a general term,

$$\frac{\partial Q}{\partial a_j} = -2\sum_{i=1}^{n}\left[y_i - f(x_i)\right]\frac{\partial f(x_i)}{\partial a_j} = 0,$$
$$\frac{\partial f(x_i)}{\partial a_j} = x_i^{j-1}.$$

Thus we have

$$\left.\begin{aligned} &\sum_{i=1}^{n}\left[y_i - f(x_i)\right] x_i^{j-1} = 0 \\ &\sum \left[y_i x_i^{j-1} - x_i^{j-1} f(x_i)\right] = 0 \end{aligned}\right. \qquad j = 1, 2, \cdots, m.$$

Substituting for $f(x_i)$

$$\sum_{i=1}^{n} x_i^{j-1}\left(a_1 + a_2 x^i + a_3 x_i^2 + \cdots + a_m x_i^{m-1}\right) = \sum_{i=1}^{n} y_i x_i^{j-1}.$$

These are *m* equations $(j = 1, 2\cdots, m)$ and each summation is for $i = 1$ to *n*.

$$\begin{aligned}
&a_1 n + a_2 \sum x_i + a_3 \sum x_i^2 + \cdots + a_m \sum x_i^{m-1} = \sum y_i,\\
&a_1 \sum x_i + a_2 \sum x_i^2 + a_3 \sum x_i^3 \cdots + a_m \sum x_i^m = \sum y_i x_i,\\
&\vdots\\
&a_1 \sum x_i^{m-1} + a_2 \sum x_i^m + a_3 \sum x_i^{m-1} \cdots + a_m \sum x_i^{2m-2} = \sum y_i x_i^{m-1}.
\end{aligned}$$

The set of *m* equations can be represented in a matrix notation as follows:

$$CA = B$$

where

$$C = \begin{bmatrix} n & \sum x_i & \sum x_i^2 & \cdots & \sum x_i^{m-1} \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \cdots & \sum x_i^m \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sum x_i^{m-1} & \sum x_i^m & \cdots & \cdots & \sum x_i^{2m-2} \end{bmatrix}, \quad A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_m \end{bmatrix}, \quad B = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \\ \sum y_i x_2 \\ \vdots \\ \sum y_i x_i^{m-1} \end{bmatrix}.$$

The element of matrix *C* is

$$C(j,k) = \sum_{i=1}^{n} x_i^{j+k-2}, \; j = 1, 2, \cdots, m, \text{ and } k = 1, 2, \cdots, m$$

$$B(j) = \sum_{i=1}^{n} y_i x_i^{j-1}, \; j = 1, 2, \cdots, m.$$

The first model which we fit the yearly soybean yield is the linear model described by

$$L = a_0 + a_1 x + a_2 y + a_3 z + a_4 w \tag{3}$$

where a_0 being a constant term, *w* is the wind direction in degree, *x* being temperature parameter in degree Celsius, "*y*" the percentage humidity, "*z*" is the speed of wind in km/hr.

The error in the generalisation

$$\left[L - \left(a_0 + a_1 x + a_2 y + a_3 z + a_4 w\right)\right]. \tag{4}$$

And squaring the error term for Minimum Squared Error

$$E = \left[\left[L - \left(a_0 + a_1 x + a_2 y + a_3 z + a_4 w\right)\right]^2\right]. \tag{5}$$

Differentiating with respect to various factors, similar to equation for the weighted coefficients for the parameters that determine the yield, given by

$$\begin{aligned}
&24\sum L = 284a_0 + 9892.52a_1 + 1233.43a_2 + 46602.534a_3 + 15964.916a_4,\\
&899.32\sum L = 9892.52a_0 + 404388.23a_1 + 50420.375a_2 + 1905026.85a_3 + 652616.73a_4,\\
&112.13\sum L = 1233.43a_0 + 50420.37a_1 + 6286.56a_2 + 237524.64a_3 + 81370.27a_4,\\
&4236.594\sum L = 46602.53a_0 + 1905026.85a_1 + 237524.64a_2 + 8974364.36a_3 + 3074369.168a_4,\\
&1451.356\sum L = 15964.916a_0 + 725184.539a_1 + 81370.274a_2 + 3074403.016a_3 + 1053217.119a_4.
\end{aligned}$$

The yield that is $\sum L$ as per statistics available from the first estimate of soybean crop from Soybean Processor Association of India [6] (SoPA 2012) is 1150 kg/hectare.

Solving the equations to get the values of the weighted coefficients

$$\begin{aligned} a_0 &= 104.545, \\ a_1 &= 2.145, \\ a_2 &= -0.000001139, \\ a_3 &= -8.368, \\ a_4 &= 0.0000013472. \end{aligned}$$

Generalizing the model the yield can be predicted by

$$L = 104.545 + 0.0000013472w + 2.145x - 0.000001139y - 8.368z$$

where the *w*, *x*, *y*, *z* are the parameters discussed above.

The second model which we fit the yearly soybean yield is the linear model described by

$$L = a_0 + a_1x^2 + a_2y^2 + a_3z^2 + a_4w^2 \tag{6}$$

where a_0 being a constant term, *w* is the wind direction in degrees, *x* being temperature parameter in degree Celsius, "*y*" the percentage humidity, "*z*" is the speed of wind in km/hr.

The error in the generalisation

$$\left[L - \left(a_0 + a_1x^2 + a_2y^2 + a_3z^2 + a_4w^2\right)\right]. \tag{7}$$

And squaring the error term for Minimum Squared Error

$$E = \left[\left[L - \left(a_0 + a_1x^2 + a_2y^2 + a_3z^2 + a_4w^2\right)\right]^2\right]. \tag{8}$$

Differentiating with respect to various factors, similar to equation for the weighted coefficients for the parameters that determine the yield, given by

$$\begin{aligned} &24\sum L = 284a_0 + 7751.249a_1 + 25624.804a_2 + 317.6214a_3 + 479550.034a_4, \\ &15502.498\sum L = 170527.478a_0 + 120163722.1a_1 + 397248472.8a_2 + 4923925.118a_3 + 7434223443a_4, \\ &51249.608\sum L = 563745.688a_0 + 397248472.8a_1 + 1313261160a_2 + 16277972.24a_3 + 24576751260a_4, \\ &635.2428\sum L = 6987.6708a_0 + 4923965.424a_1 + 16277972.24a_2 + 201766.7074a_3 + 304630706.4a_4, \\ &479550.034\sum L = 5275050.374a_0 + 3717111721a_1 + 12288375630a_2 + 152315353.2a_3 + 22996823510a_4. \end{aligned}$$

The yield that is $\sum L$ as per statistics available from the first estimate of soybean crop from Soybean Processor Association of India (SoPA 2012) is 1150 kg/hectare.

Solving the equations to get the values of the weighted coefficients

$$\begin{aligned} a_0 &= -30.304, \\ a_1 &= 2.247, \\ a_2 &= 0.06088, \\ a_3 &= -0.242, \\ a_4 &= -1.545. \end{aligned}$$

Generalizing the model the yield can be predicted by

$$L = -30.304 + 2.247x^2 + 0.06088y^2 - 0.242z^2 - 1.545a_4w^2$$

where the *w*, *x*, *y*, *z* are the parameters discussed above.

The third model which we fit the yearly soybean yield is the linear model described by

$$L = a_0 + a_1x^3 + a_2y^2 + a_3z^4 + a_4w \tag{9}$$

where a_0 being a constant term, *w* is the wind direction in degrees, *x* being temperature parameter in degree Celsius, "*y*" the percentage humidity, "*z*" is the speed of wind in km/hr.

The error in the generalisation

$$\left[L-\left(a_0+a_1x^3+a_2y^2+a_3z^4+a_4w\right)\right]. \tag{10}$$

And squaring the error term for Minimum Squared Error

$$E=\left[\left[L-\left(a_0+a_1x^3+a_2y^2+a_3z^4+a_4w\right)\right]^2\right]. \tag{11}$$

Differentiating with respect to various factors, similar to equation for the weighted coefficients for the parameters that determine the yield, given by

$$24\sum L=284a_0+128601055.4a_1+600745.167a_2+309524.226a_3+46697.354a_4,$$
$$111691004.5\sum L=12860105.41a_0+68339793020000a_1+319241581600a_2$$
$$+164484050800a_3+24815407980a_4,$$
$$54613.197\sum L=600745.167a_0+319241565800a_1+1491300643a_2$$
$$+768368524.2a_3+115922354.2a_4,$$
$$28138.566\sum L=309522.466a_0+164484050800a_1+768368524.2a_2$$
$$+3958889448a_3+59727117.16a_4,$$
$$4245.214\sum L=46697.354a_0+24815407980a_1+115922354.2a_2$$
$$+59727117.16a_3+9010920.94a_4.$$

The yield that is $\sum L$ as per statistics available from the first estimate of soybean crop from Soybean Processor Association of India (SoPA 2012) is 1150 kg/hectare.

Solving the equations to get the values of the weighted coefficients

$$a_0=-0.00004890265298,$$
$$a_1=0.0006236823179,$$
$$a_2=-0.21479748791427158,$$
$$a_3=0.2395042332638745,$$
$$a_4=-5.32059174909421e-8.$$

Generalizing the model the yield can be predicted by

$$L=-30.304+2.247x^2+0.06088y^2-0.242z^2-1.545a_4w^2$$

where the *w*, *x*, *y*, *z* are the parameters discussed above.

3. Chaos

Chaos is associated with complex and unpredictable behavior of phenomena over time [7]. Such behavior can arise in deterministic dynamical systems. These processes are intriguing in that the realizations corresponding to different, although extremely close, initial conditions typically diverge. The practical implication of this phenomenon is that, despite the underlying determinism, we cannot predict, with any reasonable precision, the values of the process for large time values; even the slightest error in specifying the initial condition eventually ruins our attempt. The chaos in terms of correlation coefficient within various environmental factors (say *n*) is given by

$$C=\left(r_{12}r_{23}r_{34}\cdots\right)^{1/n}.$$

4. Conclusion

Table 2 describes the possible correlation permutation and **Table 3** elucidates the variability of the yield amongst the different models under scrutiny. The results suggest, about the dependence of the yield on the environmental factors more under the variable weighted powers rather than being in linearly or quadratic fashion. **Figure 1** shows the proper harvesting time of the season for maximising the yield of soybean. The data are indeed direct acceptance of the model variable power model as the data match with the conventional values of

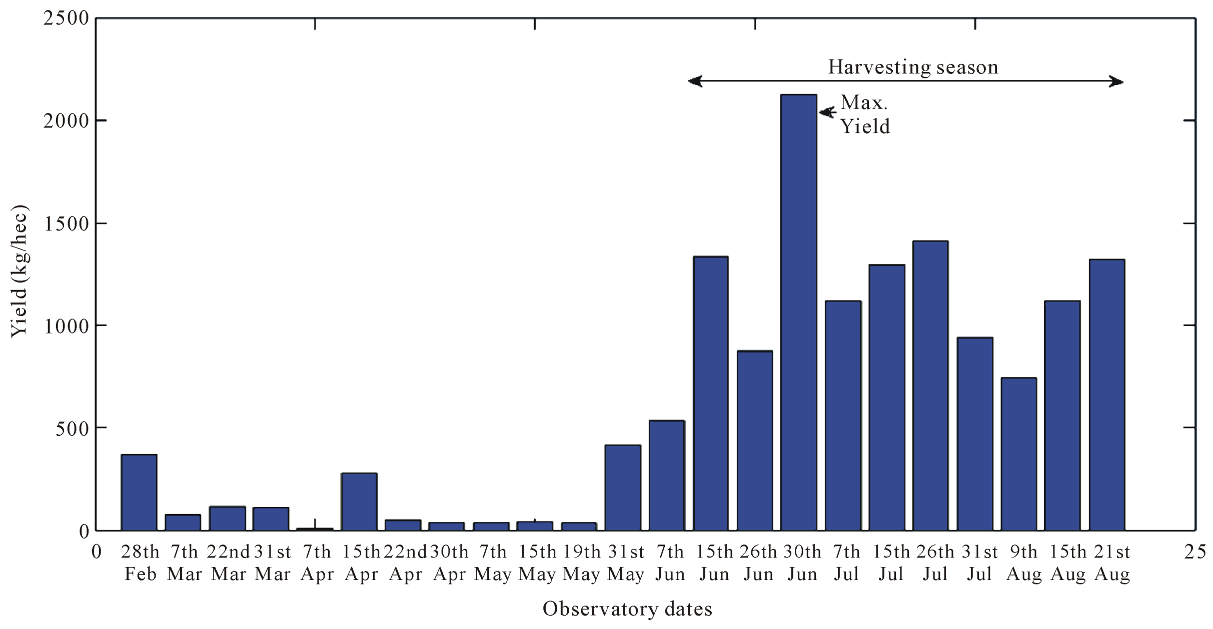


Figure 1. Statistical yield suggested by the third model for the yield of soybean in Central India.

Table 2. Correlation permutations amongst various environmental factors.

	Wind direction & temperature	Wind direction & humidity	Wind direction & wind speed	Temperature & humidity	Temperature & wind speed	Humidity & wind speed
Linear model	0.7919	0.2105	0.7554	0.6390	−0.2465	0.0168
Quadratic model	0.7930	0.2086	0.7420	−0.2465	0.6390	0.0168
Variable power model	0.7831	0.4155	0.5424	−0.1147	0.6038	0.1638

Table 3. Correct prediction percentage amongst the three models under consideration.

	Chaos
Linear model	0.2622
Quadratic model	0.2634
Variable power model	0.3554

the 2012 estimate, thereby proving the legitimacy of the accuracy of the computational calculation of yield using hidden environmental parameters.

References

[1] Schwartz, M.D. (1995) Detecting Structural Climate Change: An Air Mass-Based Approach in the North Central United States, 1958-1992. *Annals of the Association of American Geographers*, **85**, 553-568. http://dx.doi.org/10.1111/j.1467-8306.1995.tb01812.x

[2] Fahd, T. (1996) Botany and Agriculture. In: Morelon, R. and Rashed, R., Eds., *Encyclopedia of the History of Arabic Science*, Routledge, London, 815.

[3] Saxena, M.C. and Pandeny, R.K. (1972) Characteristics and Performance of Some Promising Varieties of Soybean. *Indian Journal of Agricultural Sciences*, **41**.

[4] Bansil, P.C. (1984) A Strategy for Self Sufficiency in Vegetable Oils. *Quarterly Economic and Agricultural Report*, **27**.

[5] Ahn, S.-J. (2008) Geometric Fitting of Parametric Curves and Surfaces. *Journal of Information Processing Systems*, **4**.

[6] (2013) All India State Wise Production and Yield of Soybean, First Estimate, Soybean Production Association of India.

[7] Devaney, R.L. (2003) An Introduction to Chaotic Dynamical Systems. 2nd Edition, Westview Press, Boulder.

American Journal of Computational Mathematics, 2014, 5, 414-445
Published Online December 2014 in SciRes. http://www.scirp.org/journal/ajcm
http://dx.doi.org/10.4236/ajcm.2014.45036

Extension of Smoothed Particle Hydrodynamics (SPH), Mathematical Background of Vortex Blob Method (VBM) and Moving Particle Semi-Implicit (MPS)

Hiroshi Isshiki

Institute of Mathematical Analysis (IMA), Osaka, Japan
Email: isshiki@dab.hi-ho.ne.jp

Received 12 October 2014; revised 13 November 2014; accepted 24 November 2014

Abstract

SPH has a reasonable mathematical background. Although VBM and MPS are similar to SPH, their mathematical backgrounds seem fragile. VBM has some problems in treating the viscous diffusion of vortices but is known as a practical method for calculating viscous flows. The mathematical background of MPS is also not sufficient. Not with standing, the numerical results seem reasonable in many cases. The problem common in both VBM and MPS is that the space derivatives necessary for calculating viscous diffusion are not estimated reasonably, although the treatment of advection is mathematically correct. This paper discusses a method to estimate the above mentioned problem of how to treat the space derivatives. The numerical results show the comparison among FDM (Finite Difference Method), SPH and MPS in detail. In some cases, there are big differences among them. An extension of SPH is also given.

Keywords

SPH, MPS, VBM, Concentration of Mass or Vorticity, Estimation of Space Derivatives

1. Introduction

The author has once shown how to obtain the space derivative in an irregular mesh using the moving least square method [1]. A similar problem is discussed from a different viewpoint in SPH. Gingold & Monaghan [2] and Lucy [3] have developed Smooth Particle Hydrodynamics method (SPH). In SPH, the continuous mass distribution is approximated by the finite number of particles. Namely, the continuous quantities are represented by the finite number of the discrete quantities. At the same time, the space derivatives of the distributed quantities are expressed algebraically by the discrete quantities. Then, they transform the continuous system into a discrete system convenient for the numerical solution of the initial and boundary value problem.

Vortex Blob Method (VBM) is one of the practical numerical tools for high Reynolds number flows [4]. The abduction is approximated reasonably. However, the diffusion of vortices can't be approximated precisely. We have shown that this problem can be solved, if we apply the ideas developed by SPH.

Moving Particle Semi-implicit method (MPS) uses a similar discretization of the initial and boundary value problem as SPH and VBM. However, the mathematical background is not sufficient. Not with standing, the numerical results seem reasonable in many cases [5]. This suggests us that we may find the mathematical background [1] [6]. We have shown an interesting relationship between SPH and MPS as far as the gradient operator is concerned.

The numerical results show the comparison among FDM, SPH and MPS in detail. In some cases, there are big differences among them.

A classification of numerical solutions is shown in **Figure 1**. The weak solutions: FEM, FVM, IRM and GIRM are Finite Element Method, Finite Volume Method, Integral Representation Method and Generalized Integral Representation Method, respectively. The strong solutions: FDM, SPH, MPS, LBM and ColM are Finite Difference Method, Smooth Particle Hydrodynamics, Moving Particle Semi-Implicit Method, Voltex Blob Method, Lattice Boltzmann Method and Collocation Method, respectively. The general characteristics of the various numerical solutions are shown in **Table 1**. Since SPH, VBM and MPS belong to the strong solutions and are mesh-less methods, the low computational cost becomes very important in some cases.

2. Discretization Used in SPH and the Extension

Let $\boldsymbol{x} = x^{\alpha}\boldsymbol{e}_{\alpha} = x^1\boldsymbol{e}_1 + x^2\boldsymbol{e}_2 + x^3\boldsymbol{e}_3 = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$ be a position vector, where the suffix in Greek letter is used to refer to the component of the coordinates, and the summation convention is used for the Greek suffixes. We define a function $\eta(\boldsymbol{x})$ as an integral of $w(\boldsymbol{x}-\boldsymbol{x}')\xi(\boldsymbol{x}')\rho(\boldsymbol{x}')$ with respect to a volume *V*, where $w(\boldsymbol{x})$, $\xi(\boldsymbol{x})$ and $\rho(\boldsymbol{x})$ are a weight function, an auxiliary function and the density of a fluid, respectively:

$$\eta(\boldsymbol{x}) = \int_V w(\boldsymbol{x}-\boldsymbol{x}')\xi(\boldsymbol{x}')\rho(\boldsymbol{x}')\mathrm{d}V'. \tag{1}$$

The function $\eta(\boldsymbol{x})$ and $\xi(\boldsymbol{x})$ can be scalars, vectors and tensors. If we introduce an discretization of the volume *V* and the density ρ as

$$V = \sum_j V_j, \quad j = 0,1,\cdots,N-1, \tag{2}$$

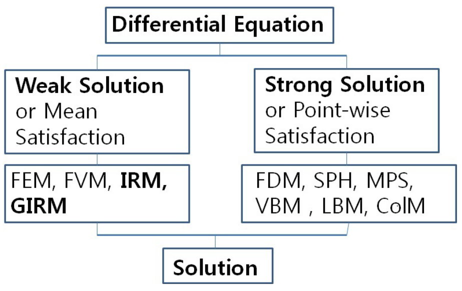

Differential Equation
Weak Solution or Mean Satisfaction
Strong Solution or Point-wise Satisfaction
FEM, FVM, IRM, GIRM
FDM, SPH, MPS, VBM , LBM, ColM
Solution

Figure 1. Classification of numerical solutions.

Table 1. General characteristics of weak and strong solutions (◎: very good, ○: good, △: not good).

	Week method	Strong method
Stability	○	×
Precision	○	△
Computational cost (memory, time)	×	◎

$$\rho(\boldsymbol{x})\mathrm{d}V = M_j, \quad j = 0,1,\cdots,N-1, \tag{3}$$

where the suffix in Roman letter is used to discriminate the point, and M_j is the mass of the volume element $\mathrm{d}V_j$. We have an approximation of Equation (1):

$$\eta(\boldsymbol{x}) = \sum_j w(\boldsymbol{x}-\boldsymbol{x}_j)\xi(\boldsymbol{x}_j)M_j \,. \tag{4}$$

Furthermore, if we assume for the weight w

$$\int_V w(\boldsymbol{x})\mathrm{d}V = 1 \tag{5a}$$

and

$$w = 0 \ \text{ for } |\boldsymbol{x}| > \sigma. \tag{5b}$$

If σ tends to 0, $w(\boldsymbol{x})$ tends to $\delta(\boldsymbol{x})$, where $\delta(\boldsymbol{x})$ is Dirac's delta function, namely:

$$w(\boldsymbol{x}) \to \delta(\boldsymbol{x}), \ \text{ when } \sigma \to 0. \tag{6}$$

Using Equation (6), we have a following approximation:

$$\eta(\boldsymbol{x}) = \int_V w(\boldsymbol{x}-\boldsymbol{x}')\xi(\boldsymbol{x}')\rho(\boldsymbol{r}')\mathrm{d}V' = \xi(\boldsymbol{x})\rho(\boldsymbol{x}). \tag{7}$$

From Equations (4) and (7), we obtain

$$\xi(\boldsymbol{x})\rho(\boldsymbol{x}) = \sum_j w(\boldsymbol{x}-\boldsymbol{x}_j)\xi(\boldsymbol{x}_j)M_j \,. \tag{8}$$

If we assume $\xi(\boldsymbol{x}) = 1$, then, we have from Equation (8)

$$\rho(\boldsymbol{x}) = \int_V w(\boldsymbol{x}-\boldsymbol{x}')\rho(\boldsymbol{x}')\mathrm{d}V' = \sum_j w(\boldsymbol{x}-\boldsymbol{x}_j)M_j \,. \tag{9}$$

Substituting $\boldsymbol{x} = \boldsymbol{x}_i$, we obtain

$$\rho(\boldsymbol{x}_i) = \sum_j w(\boldsymbol{x}_i-\boldsymbol{x}_j)M_j \,. \tag{10}$$

Hence, we have

$$\rho(\boldsymbol{x}_i) \approx w(0)M_i = w(0)\mathrm{d}V_i\rho(\boldsymbol{x}_i) \approx \rho(\boldsymbol{x}_i). \tag{11}$$

This means that Equation (9) is a plausible approximation.

Substituting Equation (9) into Equation (8), we obtain

$$\xi(\boldsymbol{x}) = \frac{\sum_j w(\boldsymbol{x}-\boldsymbol{x}_j)\xi(\boldsymbol{x}_j)M_j}{\sum_j w(\boldsymbol{x}-\boldsymbol{x}_j)M_j} = \frac{1}{\rho(\boldsymbol{x})}\sum_j w(\boldsymbol{x}-\boldsymbol{x}_j)\xi(\boldsymbol{x}_j)M_j \,. \tag{12}$$

Setting $\boldsymbol{x}$ to $\boldsymbol{x}_i$ in Equation (12), we have using Equation (10)

$$\xi(\boldsymbol{x}_i) = \frac{1}{\rho(\boldsymbol{x}_i)}\sum_j w(\boldsymbol{x}_i-\boldsymbol{x}_j)\xi(\boldsymbol{x}_j)M_j \,. \tag{13}$$

Operating $\partial/\partial x_\alpha$ on both sides of Equation (8), we obtain

$$\frac{\partial\xi(\boldsymbol{x})}{\partial x^\alpha}\rho(\boldsymbol{x}) + \xi(\boldsymbol{x})\frac{\partial\rho(\boldsymbol{x})}{\partial x^\alpha} = \sum_j \frac{\partial w(\boldsymbol{x}-\boldsymbol{x}_j)}{\partial x^\alpha}\xi(\boldsymbol{x}_j)M_j \,. \tag{14}$$

If we assume

$$w\left(\boldsymbol{x}-\boldsymbol{x}_j\right)=w\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right). \tag{15}$$

We derive

$$\frac{\partial w\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)}{\partial x^\alpha}=\frac{\mathrm{d}w\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)}{\mathrm{d}\left|\boldsymbol{x}-\boldsymbol{x}_j\right|}\frac{\partial\left|\boldsymbol{x}-\boldsymbol{x}_j\right|}{\partial x^\alpha}=w'\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)\frac{x^\alpha-x_j^\alpha}{\left|\boldsymbol{x}-\boldsymbol{x}_j\right|}, \tag{16}$$

$$\frac{\partial\rho\left(\boldsymbol{x}\right)}{\partial x^\alpha}=\sum_j\frac{\partial w\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)}{\partial x^\alpha}M_j=\sum_j w'\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)\frac{x^\alpha-x_j^\alpha}{\left|\boldsymbol{x}-\boldsymbol{x}_j\right|}M_j. \tag{17}$$

Substituting Equations (9), (16) and (17) into Equation (14), we obtain

$$\frac{\partial\xi\left(\boldsymbol{x}\right)}{\partial x^\alpha}=-\frac{1}{\rho\left(\boldsymbol{x}\right)}\sum_j w'\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)\frac{x^\alpha-x_j^\alpha}{\left|\boldsymbol{x}-\boldsymbol{x}_j\right|}\left(\xi\left(\boldsymbol{x}\right)-\xi\left(\boldsymbol{x}_j\right)\right)M_j. \tag{18}$$

Setting $\boldsymbol{x}$ to $\boldsymbol{x}_i$ in Equation (18), we have

$$\left[\frac{\partial\xi\left(\boldsymbol{x}\right)}{\partial x^\alpha}\right]_i=-\frac{1}{\rho\left(\boldsymbol{x}_i\right)}\sum_{j\neq i} w'\left(\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|\right)\frac{x_i^\alpha-x_j^\alpha}{\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|}\left(\xi\left(\boldsymbol{x}_i\right)-\xi\left(\boldsymbol{x}_j\right)\right)M_j. \tag{19}$$

$$\left[\frac{\partial\rho\left(\boldsymbol{x}\right)}{\partial x^\alpha}\right]_i=\sum_{j\neq i} w'\left(\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|\right)\frac{x_i^\alpha-x_j^\alpha}{\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|}M_j, \tag{20}$$

where we assume $w'\left(0\right)=0$.

Now, we derive formulas for second derivatives. Firstly, from Equation (14), we have

$$\begin{aligned}&\frac{\partial^2\xi\left(\boldsymbol{x}\right)}{\partial x^\alpha\partial x^\beta}\rho\left(\boldsymbol{x}\right)+\frac{\partial\xi\left(\boldsymbol{x}\right)}{\partial x^\beta}\frac{\partial\rho\left(\boldsymbol{x}\right)}{\partial x^\alpha}+\frac{\partial\xi\left(\boldsymbol{x}\right)}{\partial x^\alpha}\frac{\partial\rho\left(\boldsymbol{x}\right)}{\partial x^\beta}+\xi\left(\boldsymbol{x}\right)\frac{\partial^2\rho\left(\boldsymbol{x}\right)}{\partial x^\alpha\partial x^\beta}\\&=\sum_j\frac{\partial^2 w\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)}{\partial x^\alpha\partial x^\beta}\xi\left(\boldsymbol{x}_j\right)M_j.\end{aligned} \tag{21}$$

From Equations (16) and (17), we obtain

$$\begin{aligned}\frac{\partial^2 w\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)}{\partial x^\alpha\partial x^\beta}&=\frac{\partial}{\partial x^\alpha}\left[w'\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)\frac{x^\beta-x_j^\beta}{\left|\boldsymbol{x}-\boldsymbol{x}_j\right|}\right]\\&=w''\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)\frac{\left(x^\alpha-x_j^\alpha\right)\left(x^\beta-x_j^\beta\right)}{\left|\boldsymbol{x}-\boldsymbol{x}_j\right|^2}+w'\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)\left(\frac{\delta^{\alpha\beta}}{\left|\boldsymbol{x}-\boldsymbol{x}_j\right|}-\frac{\left(x^\alpha-x_j^\alpha\right)\left(x^\beta-x_j^\beta\right)}{\left|\boldsymbol{x}-\boldsymbol{x}_j\right|^3}\right),\end{aligned} \tag{22}$$

$$\begin{aligned}\frac{\partial^2\rho\left(\boldsymbol{x}\right)}{\partial x^\alpha\partial x^\beta}&=\sum_j\frac{\partial^2 w\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)}{\partial x^\alpha\partial x^\beta}M_j\\&=\sum_j\left[w''\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)\frac{\left(x^\alpha-x_j^\alpha\right)\left(x^\beta-x_j^\beta\right)}{\left|\boldsymbol{x}-\boldsymbol{x}_j\right|^2}+w'\left(\left|\boldsymbol{x}-\boldsymbol{x}_j\right|\right)\left(\frac{\delta^{\alpha\beta}}{\left|\boldsymbol{x}-\boldsymbol{x}_j\right|}-\frac{\left(x^\alpha-x_j^\alpha\right)\left(x^\beta-x_j^\beta\right)}{\left|\boldsymbol{x}-\boldsymbol{x}_j\right|^3}\right)\right]M_j,\end{aligned} \tag{23}$$

where $\delta^{\alpha\beta}$ is Chronecker's delta: $\delta^{\alpha\beta}=1$ and $\delta^{\alpha\beta}=0$ when $\alpha=\beta$ and $\alpha\neq\beta$, respectively. Substituting Equations (9), (17), (22) and (23) into Equation (21), we derive

$$
\begin{aligned}
&\frac{\partial^2\xi(\boldsymbol{x})}{\partial x^\alpha\partial x^\beta}\rho(\boldsymbol{x})=\sum_j\frac{\partial^2 w(|\boldsymbol{x}-\boldsymbol{x}_j|)}{\partial x^\alpha\partial x^\beta}\xi(\boldsymbol{x}_j)M_j-\frac{\partial\xi(\boldsymbol{x})}{\partial x^\beta}\frac{\partial\rho(\boldsymbol{x})}{\partial x^\alpha}-\frac{\partial\xi(\boldsymbol{x})}{\partial x^\alpha}\frac{\partial\rho(\boldsymbol{x})}{\partial x^\beta}-\xi(\boldsymbol{x})\frac{\partial^2\rho(\boldsymbol{x})}{\partial x^\alpha\partial x^\beta}\\
&=\sum_j\left[w''(|\boldsymbol{x}-\boldsymbol{x}_j|)\frac{(x^\alpha-x_j^\alpha)(x^\beta-x_j^\beta)}{|\boldsymbol{x}-\boldsymbol{x}_j|^2}+w'(|\boldsymbol{x}-\boldsymbol{x}_j|)\left(\frac{\delta^{\alpha\beta}}{|\boldsymbol{x}-\boldsymbol{x}_j|}-\frac{(x^\alpha-x_j^\alpha)(x^\beta-x_j^\beta)}{|\boldsymbol{x}-\boldsymbol{x}_j|^3}\right)\right]\xi(\boldsymbol{x}_j)M_j\\
&\quad+\frac{1}{\rho(\boldsymbol{x})}\sum_j w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{x_i^\beta-x_j^\beta}{|\boldsymbol{x}-\boldsymbol{x}_j|}(\xi(\boldsymbol{x}_i)-\xi(\boldsymbol{x}_j))M_j\sum_j w'(|\boldsymbol{x}-\boldsymbol{x}_j|)\frac{x^\alpha-x_j^\alpha}{\|\boldsymbol{x}-\boldsymbol{x}_j\|}M_j\\
&\quad+\frac{1}{\rho(\boldsymbol{x})}\sum_j w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{x_i^\alpha-x_j^\alpha}{|\boldsymbol{x}-\boldsymbol{x}_j|}(\xi(\boldsymbol{x}_i)-\xi(\boldsymbol{x}_j))M_j\sum_j w'(|\boldsymbol{x}-\boldsymbol{x}_j|)\frac{x^\beta-x_j^\beta}{|\boldsymbol{x}-\boldsymbol{x}_j|}M_j\\
&\quad-\xi(\boldsymbol{x})\sum_j\left[w''(|\boldsymbol{x}-\boldsymbol{x}_j|)\frac{(x^\alpha-x_j^\alpha)(x^\beta-x_j^\beta)}{|\boldsymbol{x}-\boldsymbol{x}_j|^2}+w'(|\boldsymbol{x}-\boldsymbol{x}_j|)\left(\frac{\delta^{\alpha\beta}}{|\boldsymbol{x}-\boldsymbol{x}_j|}-\frac{(x^\alpha-x_j^\alpha)(x^\beta-x_j^\beta)}{|\boldsymbol{x}-\boldsymbol{x}_j|^3}\right)\right]M_j\\
&=-\sum_j\left[w''(|\boldsymbol{x}-\boldsymbol{x}_j|)\frac{(x^\alpha-x_j^\alpha)(x^\beta-x_j^\beta)}{|\boldsymbol{x}-\boldsymbol{x}_j|^2}+w'(|\boldsymbol{x}-\boldsymbol{x}_j|)\left(\frac{\delta^{\alpha\beta}}{|\boldsymbol{x}-\boldsymbol{x}_j|}-\frac{(x^\alpha-x_j^\alpha)(x^\beta-x_j^\beta)}{|\boldsymbol{x}-\boldsymbol{x}_j|^3}\right)\right]\cdot(\xi(\boldsymbol{x})-\xi(\boldsymbol{x}_j))M_j\\
&\quad+\frac{1}{\rho(\boldsymbol{x})}\sum_j\sum_k w'(|\boldsymbol{x}-\boldsymbol{x}_k|)w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{\left[(x_i^\beta-x_j^\beta)(x^\alpha-x_k^\alpha)+(x_i^\alpha-x_j^\alpha)(x^\beta-x_k^\beta)\right]}{|\boldsymbol{x}-\boldsymbol{x}_j|\cdot|\boldsymbol{x}-\boldsymbol{x}_k|}(\xi(\boldsymbol{x}_i)-\xi(\boldsymbol{x}_j))M_jM_k.
\end{aligned}
\tag{24}
$$

Setting $\boldsymbol{x}$ to $\boldsymbol{x}_i$ in Equations (22), (23) and (24), we have

$$
\left[\frac{\partial^2 w(|\boldsymbol{x}-\boldsymbol{x}_j|)}{\partial x^\alpha\partial x^\beta}\right]_i=w''(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{(x_i^\alpha-x_j^\alpha)(x_i^\beta-x_j^\beta)}{|\boldsymbol{x}_i-\boldsymbol{x}_j|^2}+w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\left(\frac{\delta^{\alpha\beta}}{|\boldsymbol{x}_i-\boldsymbol{x}_j|}-\frac{(x_i^\alpha-x_j^\alpha)(x_i^\beta-x_j^\beta)}{|\boldsymbol{x}_i-\boldsymbol{x}_j|^3}\right),\tag{25}
$$

$$
\left[\frac{\partial^2\rho(\boldsymbol{x})}{\partial x^\alpha\partial x^\beta}\right]_i=\sum_j\left[w''(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{(x_i^\alpha-x_j^\alpha)(x_i^\beta-x_j^\beta)}{|\boldsymbol{x}_i-\boldsymbol{x}_j|^2}+w'(|\boldsymbol{x}-\boldsymbol{x}_j|)\left(\frac{\delta^{\alpha\beta}}{|\boldsymbol{x}_i-\boldsymbol{x}_j|}-\frac{(x_i^\alpha-x_j^\alpha)(x_i^\beta-x_j^\beta)}{|\boldsymbol{x}_i-\boldsymbol{x}_j|^3}\right)\right]M_j,\tag{26}
$$

$$
\begin{aligned}
&\left[\frac{\partial^2\xi(\boldsymbol{x})}{\partial x^\alpha\partial x^\beta}\right]_i\rho(\boldsymbol{x}_i)\\
&=-\sum_{j\neq i}\left[w''(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{(x_i^\alpha-x_j^\alpha)(x_i^\beta-x_j^\beta)}{|\boldsymbol{x}_i-\boldsymbol{x}_j|^2}+w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\left(\frac{\delta^{\alpha\beta}}{|\boldsymbol{x}_i-\boldsymbol{x}_j|}-\frac{(x_i^\alpha-x_j^\alpha)(x_i^\beta-x_j^\beta)}{|\boldsymbol{x}_i-\boldsymbol{x}_j|^3}\right)\right](\xi(\boldsymbol{x})-\xi(\boldsymbol{x}_j))M_j\\
&\quad+\frac{1}{\rho(\boldsymbol{x}_i)}\sum_{j\neq i}\sum_{k\neq i}w'(|\boldsymbol{x}_i-\boldsymbol{x}_k|)w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{\left[(x_i^\beta-x_j^\beta)(x_i^\alpha-x_k^\alpha)+(x_i^\alpha-x_j^\alpha)(x_i^\beta-x_k^\beta)\right]}{|\boldsymbol{x}_i-\boldsymbol{x}_j|\cdot|\boldsymbol{x}_i-\boldsymbol{x}_k|}(\xi(\boldsymbol{x}_i)-\xi(\boldsymbol{x}_j))M_jM_k,
\end{aligned}
\tag{27}
$$

where we assume $w''(0)$ is finite. From Equation (19), we obtain for scalar ϕ and vector $\boldsymbol{a}$

$$
[\nabla\phi]_i=-\frac{1}{\rho(\boldsymbol{x}_i)}\sum_{j\neq i}w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{\boldsymbol{x}_i-\boldsymbol{x}_j}{|\boldsymbol{x}_i-\boldsymbol{x}_j|}(\phi(\boldsymbol{x}_i)-\phi(\boldsymbol{x}_j))M_j,\tag{28}
$$

$$
[\nabla\cdot\boldsymbol{a}]_i=-\frac{1}{\rho(\boldsymbol{x}_i)}\sum_{j\neq i}w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{(\boldsymbol{x}_i-\boldsymbol{x}_j)\cdot(\boldsymbol{a}(\boldsymbol{x}_i)-\boldsymbol{a}(\boldsymbol{x}_j))}{|\boldsymbol{x}_i-\boldsymbol{x}_j|}M_j.\tag{29}
$$

From Equation (27), we obtain for scalar ϕ

$$\begin{aligned}
&\left[\nabla^2\phi(\boldsymbol{x})\right]_i \rho(\boldsymbol{x}_i) \\
&= -\sum_{j\neq i}\left[w''\left(\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|\right)\frac{\left(x_i^\alpha-x_j^\alpha\right)\left(x_i^\alpha-x_j^\alpha\right)}{\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|^2} + w'\left(\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|\right)\left(\frac{d}{\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|}-\frac{\left(x_i^\alpha-x_j^\alpha\right)\left(x_i^\alpha-x_j^\alpha\right)}{\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|^3}\right)\right]\left(\phi(\boldsymbol{x})-\phi\left(\boldsymbol{x}_j\right)\right)M_j \\
&\quad + \frac{1}{\rho(\boldsymbol{x}_i)}\sum_{j\neq i}\sum_{k\neq i} w'\left(\left|\boldsymbol{x}_i-\boldsymbol{x}_k\right|\right) w'\left(\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|\right)\frac{\left[\left(x_i^\alpha-x_j^\alpha\right)\left(x_i^\alpha-x_k^\alpha\right)+\left(x_i^\alpha-x_j^\alpha\right)\left(x_i^\alpha-x_k^\alpha\right)\right]}{\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|\cdot\left|\boldsymbol{x}_i-\boldsymbol{x}_k\right|}\left(\phi(\boldsymbol{x}_i)-\phi\left(\boldsymbol{x}_j\right)\right)M_jM_k \\
&= -\sum_{j\neq i}\left[w''\left(\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|\right) + w'\left(\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|\right)\frac{d-1}{\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|}\right]\left(\phi(\boldsymbol{x}_i)-\phi\left(\boldsymbol{x}_j\right)\right)M_j \\
&\quad + 2\frac{1}{\rho(\boldsymbol{x}_i)}\sum_{j\neq i}\sum_{k\neq i} w'\left(\left|\boldsymbol{x}_i-\boldsymbol{x}_k\right|\right) w'\left(\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|\right)\frac{\left(\boldsymbol{x}_i-\boldsymbol{x}_j\right)\cdot\left(\boldsymbol{x}_i-\boldsymbol{x}_k\right)}{\left|\boldsymbol{x}_i-\boldsymbol{x}_j\right|\cdot\left|\boldsymbol{x}_i-\boldsymbol{x}_k\right|}\left(\phi(\boldsymbol{x}_i)-\phi\left(\boldsymbol{x}_j\right)\right)M_jM_k,
\end{aligned} \tag{30}$$

where d is the number of the dimension.

3. Application to Vortex Blob Method (VBM)

The basic equations for an incompressible viscous flow are given by Navier-Stokes equation [7]:

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} + \rho\boldsymbol{u}\cdot\nabla\boldsymbol{u} = -\nabla p + \mu\nabla^2\boldsymbol{u}, \tag{31}$$

or

$$\rho\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = -\nabla p + \mu\nabla^2\boldsymbol{u}, \tag{32}$$

where μ is the coefficient of viscosity, and the continuity equation:

$$\nabla\cdot\boldsymbol{u} = 0. \tag{33}$$

The pressure p satisfy

$$\nabla^2 p = -\rho\nabla\left(\boldsymbol{u}\cdot\nabla\boldsymbol{u}\right). \tag{34}$$

The viscous flow is determined by Equations (31) or (32), (33) and (34).

If we introduce vorticity $\boldsymbol{\omega}$, the basic equations can be rewritten as

$$\boldsymbol{\omega} = \nabla\times\boldsymbol{u},\ \ \nabla\cdot\boldsymbol{u} = 0, \tag{35}$$

$$\rho\frac{\partial\boldsymbol{\omega}}{\partial t} + \rho\boldsymbol{u}\cdot\nabla\boldsymbol{\omega} = \rho\boldsymbol{\omega}\cdot\nabla\boldsymbol{u} + \mu\Delta\boldsymbol{\omega}. \tag{36}$$

Equation (36) can be written as

$$\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t} = \boldsymbol{\omega}\cdot\nabla\boldsymbol{u} + \nu\Delta\boldsymbol{\omega}, \tag{37}$$

where $\nu = \mu/\rho$ is the kinematic viscosity.

If $\boldsymbol{\omega}$ is determined, the velocity $\boldsymbol{u}$ is obtained by an integral representation [7]:

$$\begin{aligned}
\varepsilon\boldsymbol{u}(\boldsymbol{x},t) = &-\iint_S\left[\boldsymbol{G}(\boldsymbol{x},\xi,t)\left(\boldsymbol{u}(\xi,t)\cdot\boldsymbol{n}_\xi\right) - \boldsymbol{G}(\boldsymbol{x},\xi,t)\times\left(\boldsymbol{n}_\xi\times\boldsymbol{u}(\xi,t)\right)\right]\mathrm{d}S_\xi \\
&-\iiint_V \boldsymbol{G}(\boldsymbol{x},\xi,t)\times\boldsymbol{\omega}(\xi,t)\,\mathrm{d}V_\xi + \boldsymbol{u}_\infty(\boldsymbol{x},\xi,t),
\end{aligned} \tag{38}$$

where $\varepsilon = 1, 1/2, 0$ when $\boldsymbol{x}\in V$, $\boldsymbol{x}\in S$, $\boldsymbol{x}\notin V\cup S$, respectively, and

$$\Delta_x G(\boldsymbol{x},\xi) = \delta(\boldsymbol{x},\xi), \tag{39a}$$

$$G(\boldsymbol{x},\xi)=\begin{cases}(1/2)|\boldsymbol{x}-\xi| & \text{in 1D}\\ (1/2\pi)\ln(|\boldsymbol{x}-\xi|) & \text{in 2D}\\ -(1/4\pi)|\boldsymbol{x}-\xi|^{-1} & \text{in 3D}\end{cases} \tag{39b}$$

$$\nabla_x G(\boldsymbol{x},\xi)=\boldsymbol{G}(\boldsymbol{x},\xi)=G_i(\boldsymbol{x},\xi)\boldsymbol{e}_i=\begin{cases}(1/2)(\boldsymbol{x}-\xi)/|\boldsymbol{x}-\xi| & \text{in 1D}\\ (1/2\pi)(\boldsymbol{x}-\xi)/|\boldsymbol{x}-\xi|^2 & \text{in 2D}\\ (1/4\pi)(\boldsymbol{x}-\xi)/|\boldsymbol{x}-\xi|^3 & \text{in 3D}\end{cases} \tag{39c}$$

The second term on the right hand side of Equation (38) is usually called Bio-Savart law.

Vortex Blob Method (VBM) uses Equations (37) and (38) and discretizes the vortex field as an assembly of concentrated vortices. The position of the concentrated vortices are determined by

$$\frac{\mathrm{d}\boldsymbol{x}(\boldsymbol{a},t)}{\mathrm{d}t}=\boldsymbol{u}(\boldsymbol{a},t), \tag{40}$$

where $\boldsymbol{a}$ is the Lagrangian coordinates. It becomes very important how to obtain $\boldsymbol{\omega}\cdot\nabla\boldsymbol{u}$ and $\Delta\boldsymbol{\omega}$.

In this problem, we use the vorticity $\boldsymbol{\omega}$ corresponds to the density ρ in Section 2. Then, we consider instead of Equation (1)

$$\eta(\boldsymbol{x})=\int_V w(\boldsymbol{x}-\boldsymbol{x}')\xi(\boldsymbol{x}')\omega(\boldsymbol{x}')\mathrm{d}V'. \tag{41}$$

Equation (3) is replaced by

$$\omega(\boldsymbol{x}')\mathrm{d}V'=\Omega_j,\quad j=0,1,\cdots,N-1. \tag{42}$$

Then, we have instead of Equations (8) and (9)

$$\xi(\boldsymbol{x})\omega(\boldsymbol{x})=\sum_j w(\boldsymbol{x}-\boldsymbol{x}_j)\xi(\boldsymbol{x}_j)\Omega_j. \tag{43}$$

$$\omega(\boldsymbol{x})=\int_V w(\boldsymbol{x}-\boldsymbol{x}')\omega(\boldsymbol{x}')\mathrm{d}V'=\sum_j w(\boldsymbol{x}-\boldsymbol{x}_j)\Omega_j. \tag{44}$$

In all equations in Section 2, if we replace ρ and M with $\boldsymbol{\omega}$ and Ω, respectively, we obtain the differential formulas in Section 3.

4. Mathematical Background of Moving Particle Semi-Implicit Method (MPS)

From Equations (19), (28) and (30), we have

$$\left[\frac{\partial u^\alpha(\boldsymbol{x},t)}{\partial x^\beta}\right]_i=-\frac{1}{\rho(\boldsymbol{x}_i)}\sum_{j\neq i}w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{x_i^\beta-x_j^\beta}{|\boldsymbol{x}_i-\boldsymbol{x}_j|}\left(u^\alpha(\boldsymbol{x}_i,t)-u^\alpha(\boldsymbol{x}_j,t)\right)M_j, \tag{45}$$

$$[\nabla p]_i=-\frac{1}{\rho(\boldsymbol{x}_i)}\sum_{j\neq i}\left(p(\boldsymbol{x}_i)-p(\boldsymbol{x}_j)\right)w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{\boldsymbol{x}_i-\boldsymbol{x}_j}{|\boldsymbol{x}_i-\boldsymbol{x}_j|}M_j, \tag{46}$$

$$\begin{aligned}\left[\nabla^2u^\alpha(\boldsymbol{x},t)\right]_i\rho(\boldsymbol{x}_i)=&-\sum_{j\neq i}\left[w''(|\boldsymbol{x}_i-\boldsymbol{x}_j|)+w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{d-1}{|\boldsymbol{x}_i-\boldsymbol{x}_j|}\right]\left(u^\alpha(\boldsymbol{x}_i,t)-u^\alpha(\boldsymbol{x}_j,t)\right)M_j\\&+2\frac{1}{\rho(\boldsymbol{x}_i)}\sum_{j\neq i}\sum_{k\neq i}w'(|\boldsymbol{x}_i-\boldsymbol{x}_k|)w'(|\boldsymbol{x}_i-\boldsymbol{x}_j|)\frac{(\boldsymbol{x}_i-\boldsymbol{x}_j)}{|\boldsymbol{x}_i-\boldsymbol{x}_j|}\cdot\frac{(\boldsymbol{x}_i-\boldsymbol{x}_k)}{|\boldsymbol{x}_i-\boldsymbol{x}_k|}\left(\phi(\boldsymbol{x}_i)-\phi(\boldsymbol{x}_j)\right)M_jM_k.\end{aligned} \tag{47}$$

If the weight $w(r)$ satisfies

$$\frac{\mathrm{d}w(r)}{\mathrm{d}r}\sim-\frac{1}{r}w(r)\ \text{and}\ \left[2\delta_{i1}+(1-\delta_{i1})2(d-1)\pi\right]\int_0^\infty w(r)r^{d-1}\mathrm{d}r=1, \tag{48}$$

then, we obtain

$$w(r) \sim \frac{c(d)}{r}, \tag{49}$$

where $c(d)$ is a constant. Then, we have

$$\frac{\mathrm{d}w(r)}{\mathrm{d}r} \sim -\frac{c(d)}{r^2} \sim -\frac{1}{r}w(r) \text{ and } \frac{\mathrm{d}^2 w(r)}{\mathrm{d}r^2} = \frac{2c(d)}{r^3} = \frac{2}{r^2}w(r). \tag{50}$$

Substituting Equations (49) and (50) into Equations (45), (46) and (47), we obtain

$$\left[\frac{\partial u^\alpha(\boldsymbol{x},t)}{\partial x^\beta}\right]_i \approx \frac{1}{\rho(\boldsymbol{x}_i)}\sum_j w\left(\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|\right)\frac{x_i^\beta - x_j^\beta}{\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|^2}\left(u^\alpha(\boldsymbol{x}_i,t) - u^\alpha(\boldsymbol{x}_j,t)\right)M_j, \tag{51}$$

$$[\nabla p]_i \approx \frac{1}{\rho(\boldsymbol{x}_i)}\sum_j \left(p(\boldsymbol{x}_i) - p(\boldsymbol{x}_j)\right)w\left(\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|\right)\frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|^2}M_j, \tag{52}$$

$$\begin{aligned}\left[\nabla^2 u^\alpha(\boldsymbol{x},t)\right]_i \rho(\boldsymbol{x}_i) &\approx \sum_{j\neq i} w\left(\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|\right)\frac{d-3}{\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|^2}\left(u^\alpha(\boldsymbol{x}_i,t) - u^\alpha(\boldsymbol{x}_j,t)\right)M_j \\ &+ 2\frac{1}{\rho(\boldsymbol{x}_i)}\sum_{j\neq i}\sum_{k\neq i} w\left(\left|\boldsymbol{x}_i - \boldsymbol{x}_k\right|\right)w\left(\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|\right)\frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)\cdot(\boldsymbol{x}_i - \boldsymbol{x}_k)}{\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|^2 \cdot \left|\boldsymbol{x}_i - \boldsymbol{x}_k\right|^2}\left(u^\alpha(\boldsymbol{x}_i) - u^\alpha(\boldsymbol{x}_j)\right)M_j M_k.\end{aligned} \tag{53}$$

Now, we consider a weight $w(r)$ with a small parameter $0 < \varepsilon = r_e$ and constant α:

$$w(r) = \begin{cases} \alpha\left(\dfrac{r_e}{r+\varepsilon} - \dfrac{1}{1+\varepsilon/r_e}\right) & (0 \le r \le r_e) \\ 0 & (r_e \le r), \end{cases} \tag{54}$$

where $r = |\boldsymbol{x}|$. The weight $w(r)$ has an asymptotic form:

$$w(r) = \alpha\left(\frac{r_e}{r+\varepsilon} - \frac{1}{1+\varepsilon/r_e}\right) \sim \alpha\left(\frac{r_e}{r} - 1\right) = \alpha r_e\left(\frac{1}{r} - \frac{1}{r_e}\right) \text{ when } \varepsilon < r \le r_e. \tag{55}$$

This asymptotic form is similar to $k(r)$ defined by Equation (4) in Ref. [5]. If r_e is big, $w(r)$ satisfies Equation (49). In this case, Equation (52) has similar forms as given by Equation (2) in Ref. [5], if we assume $M_j = 1, \quad j = 0,1,\cdots$.

The author has once discussed this problem from a different angle [1] and had the conclusion that the discrete differential operators used in MPS, especially, Laplace operator don't have the strict background from the mathematical viewpoint. Those operators should be considered to be a kind of experimental formulas that is derived numerically. If we apply them to irregular grids, the results vary irregularly. We should say the operator estimate the derivatives statistically. Quite naturally, the results are not unique. It may give a good estimate in one time and a wrong result in the other time. Hence, if we need to verify the accuracy, we should obtain several numerical results of the same problem using the various discretizations of the region and take the statistical average.

Although we do not deny this kind of approach, we wish to ensure the reliability. For the purpose, we need much discussion. Recently, a new paper [6] discusses the accuracy of Laplace operator in MPS in detail. Hopefully, we wish to prove mathematically, if possible, that, if we decrease the grid size zero, then, the numerical error approaches stably to zero.

5. Numerical Verification of Discretized Differential Operators of SPH with Gaussian Kernel

Numerical calculations were conducted to verify the validity of the discretized differential operators such as

Equations (28) and (30). For simplicity, one-dimensional (1D) cases were considered. The first and second derivates of a scalar function $\phi(x)$:

$$\phi(x) = \sin(2\pi x/L) \ \text{in} \ 0 \le x \le L \tag{56}$$

were calculated using Equations (28) and (30), respectively.

The region is divided into N intervals. First, we consider the uniform mesh. Let $\mathrm{d}x_i$ and x_i $(i = 0, 1, \cdots, N-1)$ be the length of the interval and the midpoint of the interval i, respectively:

$$x_i = \begin{cases} 0.5\mathrm{d}x_0 & \text{for } i = 0 \\ x_{i-1} + 0.5(\mathrm{d}x_{i-1} + \mathrm{d}x_i) & \text{for } i = 1, 2, \cdots \end{cases}, \quad \mathrm{d}V_j = \mathrm{d}x_j \ \text{ and } \ M_j = \rho \mathrm{d}V_j = \rho \mathrm{d}x. \tag{57), (58}$$

As the weight function $w(r)$, we use Gaussian function:

$$w(r) = \frac{1}{\sqrt{2\pi}\gamma} \exp\left(-\frac{r^2}{2\gamma^2}\right). \tag{59}$$

The weight function $w(r)$ satisfies

$$2\int_0^\infty w(x)\mathrm{d}x = 1. \tag{60}$$

The 1D version of Equations (28) and (30) are given by

$$\left[\frac{\mathrm{d}\phi}{\mathrm{d}x}\right]_i = -\frac{1}{\rho(\boldsymbol{x}_i)} \sum_{j \ne i} w'(|x_i - x_j|) \frac{x_i - x_j}{|x_i - x_j|} (\phi(x_i) - \phi(x_j)) M_j, \tag{61}$$

$$\begin{aligned} \left[\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}\right]_i \rho(x_i) = & -\sum_{j \ne i} w''(|x_i - x_j|)(\phi(x_i) - \phi(x_j)) M_j \\ & + 2\frac{1}{\rho(x_i)} \sum_{j \ne i} \sum_{k \ne i} w'(|x_i - x_k|) w'(|x_i - x_j|) \frac{(x_i - x_j)(x_i - x_k)}{|x_i - x_j||x_i - x_k|} (\phi(x_i) - \phi(x_j)) M_j M_k. \end{aligned} \tag{62}$$

5.1. Uniform Density and Regular Mesh

The density ρ_i and the length of element $\mathrm{d}x_i$ are given by

$$\rho_i = 1 \ \ i = 0, 1, \cdots, N-1, \tag{63a}$$

$$\mathrm{d}x_i = L/N \ \ i = 0, 1, \cdots, N-1, \tag{63b}$$

where the computational region is defined as $0 < x < L$.

5.1.1. Verification of Gradient Operator

From Equations (61), (63) and (45), we have

$$\left[\frac{\mathrm{d}\phi}{\mathrm{d}x}\right]_i = -\sum_{j \ne i} w'(|x_i - x_j|) \frac{x_i - x_j}{|x_i - x_j|} (\phi(x_i) - \phi(x_j)) M_j. \tag{64}$$

We used parameters: $L = 4$, $N = 80$ and $\gamma = \mathrm{d}x = 0.05$. The numerical results are shown in **Figure 2**. Although there exist large errors at the boundaries as shown in **Figure 2(a)**, the numerical results agree very well with the exact except the neighborhood of the boundaries. If we extend the region beyond the boundaries, the estimations within the original region are improved as shown in **Figure 2(b)**.

5.1.2. Verification of Laplace Operator

From Equation (62), we have

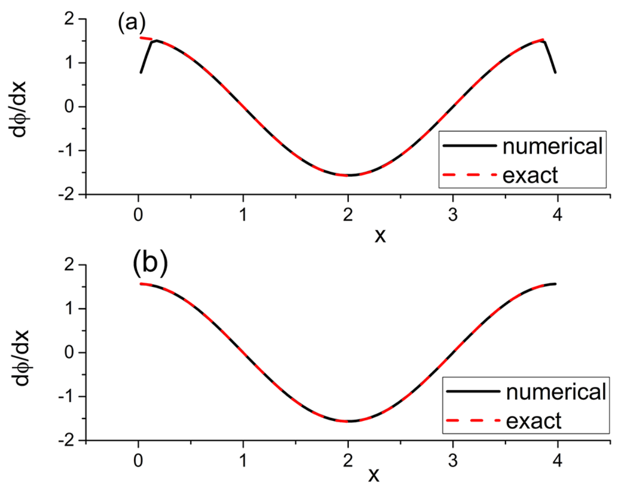


Figure 2. A comparison between calculated and exact values of dϕ/dx ((a) Not using extension of region; (b) Using extension of region).

$$\left[\frac{\mathrm{d}^2\phi(\boldsymbol{x})}{\mathrm{d}x^2}\right]_i \rho(x_i) = -\sum_{j\neq i}\left[w''\left(\left|x_i - x_j\right|\right)\right]\left(\phi(x_i)-\phi(x_j)\right)M_j$$
$$+2\frac{1}{\rho(x_i)}\sum_{j\neq i}\sum_{k\neq i} w'\left(\left|x_i-x_j\right|\right)w'\left(\left|x_i-x_k\right|\right)\frac{(x_i-x_j)\cdot(x_i-x_k)}{\left|x_i-x_j\right|\cdot\left|x_i-x_k\right|}\left(\phi(x_i)-\phi(x_j)\right)M_jM_k. \tag{65}$$

The numerical results are shown in **Figure 3**. If we extend the region beyond the boundaries, the estimations within the original region are improved.

5.2. Non-Uniform Density and Regular Mesh

The density ρ_i and the length of element $\mathrm{d}x_i$ are given by

$$\rho_i = 1+\left(x_i/L\right)^2 \quad i=0,1,\cdots,N-1, \tag{66a}$$

$$\mathrm{d}x_i = L/N, \quad i=0,1,\cdots,N-1, \tag{66b}$$

respectively. The other parameters are same as in Section 5.1.1.

The numerical results are shown in **Figure 4**. The numerical result agrees well with the exact result.

5.3. Uniform Density and Non-Uniform Mesh

The density ρ_i and the length of element $\mathrm{d}x_i$ are given by

$$\rho_i = 1 \quad i=0,1,\cdots,N-1, \tag{67a}$$

$$\mathrm{d}x_i = \alpha\frac{L}{N}(1.05)^{i-1} \quad i=0,1,\cdots,N-1, \quad \sum_{i=0}^{i=N-1}\alpha\frac{L}{N}(1.05)^{i-1} = L, \tag{67b}$$

respectively. The other parameters are same as in Section 5.1.1.

The numerical results are shown in **Figure 5**. The numerical result agrees well with the exact result.

5.4. Non-Uniform Density and Non-Uniform Mesh

The density ρ_i and the length of element $\mathrm{d}x_i$ are given by

$$\rho_i = 1+\left(x_i/L\right)^2 \quad i=0,1,\cdots,N-1, \tag{68a}$$

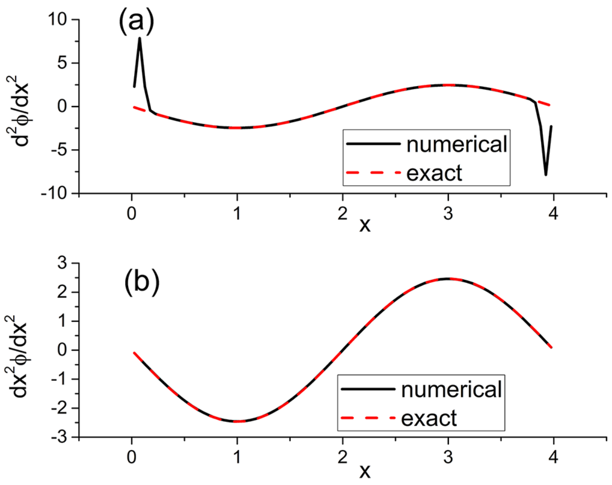


Figure 3. A comparison between calculated and exact values of $d^2\phi/dx^2$ ((a) Not using extension of region; (b) Using extension of region).

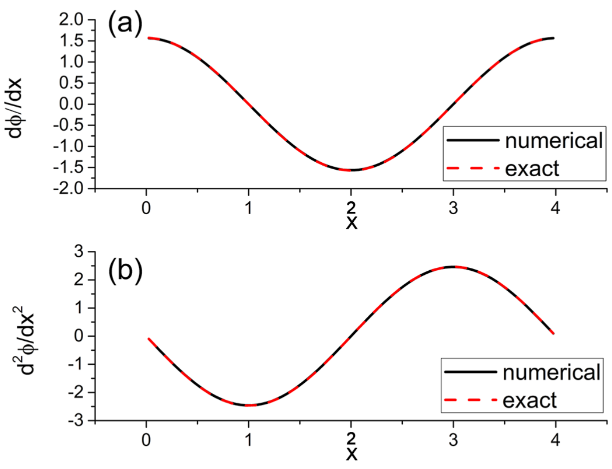


Figure 4. A comparison between calculated and exact values ((a) Gradient operator $d\phi/dx$; (b) Laplace operator $d^2\phi/dx^2$).

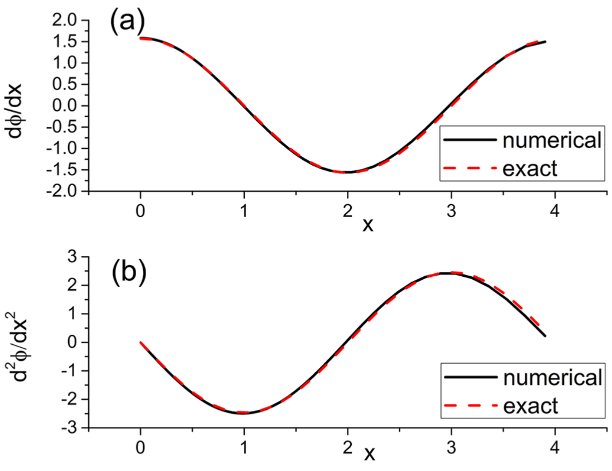


Figure 5. A comparison between calculated and exact values ((a) Gradient operator $d\phi/dx$; (b) Laplace operator $d^2\phi/dx^2$).

$$\mathrm{d}x_i = \alpha\frac{L}{N}(1.05)^{i-1} \quad i = 0,1,\cdots,N-1, \quad \sum_{i=0}^{i=N-1}\alpha\frac{L}{N}(1.05)^{i-1} = L \tag{68b}$$

respectively. The other parameter are same as in Section 5.1.1.

The numerical results are shown in Figure 6. The numerical result agrees well with the exact result.

6. Comparison between SPH and MPS

6.1. 1D Formulas for Discrete Gradient and Laplacian Operator

For convenience, we summarize the 1D discrete differential operators used in SHP and MPS as follows. From Equations (61) and (62), we have for SHP

$$\left[\frac{\mathrm{d}\phi}{\mathrm{d}x}\right]_i = -\frac{1}{\rho(\boldsymbol{x}_i)}\sum_{j\neq i} w'\left(\left|x_i - x_j\right|\right)\frac{x_i - x_j}{\left|x_i - x_j\right|}\left(\phi(x_i) - \phi(x_j)\right)M_j, \tag{69a}$$

$$\begin{aligned}\left[\nabla^2\phi\right]_i \rho(x_i) &= -\sum_{j\neq i} w''\left(\left|x_i - x_j\right|\right)\left(\phi(x_i) - \phi(x_j)\right)M_j \\ &+ 2\frac{1}{\rho(x_i)}\sum_{j\neq i}\sum_{k\neq i} w'\left(\left|x_i - x_k\right|\right)w'\left(\left|x_i - x_j\right|\right)\frac{(x_i - x_j)(x_i - x_k)}{\left|x_i - x_j\right|\left|x_i - x_k\right|}\left(\phi(x_i) - \phi(x_j)\right)M_j M_k,\end{aligned} \tag{69b}$$

$$M_i = \rho(x_i)\mathrm{d}x_i. \tag{69c}$$

From Refs. [5] and [8], we have for MPS

$$\left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_i = \frac{1}{n_i}\sum_{j\neq i}\frac{\phi_j - \phi_i}{\left|x_j - x_i\right|^2}(x_j - x_i)w\left(\left|x_j - x_i\right|\right), \tag{70a}$$

$$\left(\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}\right)_i = \frac{2}{\lambda_i n_i}\sum_{j\neq i}(\phi_j - \phi_i)w\left(\left|x_j - x_i\right|\right), \tag{70b}$$

where

$$n_i = \sum_{j\neq i} w\left(\left|x_j - x_i\right|\right), \tag{71a}$$

$$\lambda_i = \frac{\sum_{j\neq i}\left|x_j - x_i\right|^2 w\left(\left|x_j - x_i\right|\right)}{\sum_{j\neq i} w\left(\left|x_j - x_i\right|\right)}. \tag{71b}$$

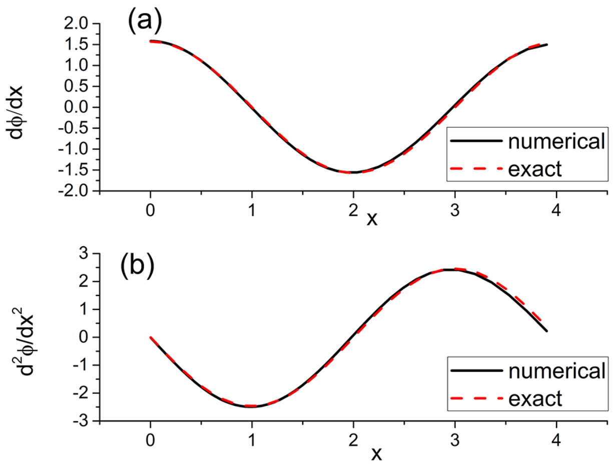


Figure 6. A comparison between calculated and exact values ((a) Gradient operator $\mathrm{d}\phi/\mathrm{d}x$; (b) Laplace operator $\mathrm{d}^2\phi/\mathrm{d}x^2$).

6.2. Effects of Mesh and Weight on Estimation of Gradient and Laplacian of a Given Function

We summarize the meshes and weights used in the following in Table 2 and Table 3, respectively.

The parameters used in Sections 6.2.1-6.2.5 are given below.

$$L=4,\quad N=40,\quad \rho=1,\quad \gamma_i=\mathrm{d}x_i \text{ for SPH0},$$

$$h_i=8\mathrm{d}x_i \text{ for SPH1 and SPH2, and } 2\mathrm{d}x_i \text{ for MPS},$$

$$N_{end\_cor}=8 \text{ for SPH0, SPH1 and SPH2, and 2 for MPS}.$$

In Sections 6.2.1-6.2.5, the computational results are shown in Figures 7-11. The results are summarized Table 4 and Table 5 in Section 6.2.6.

6.2.1. Regular Mesh

As shown in Figure 7, there are no big errors in all methods SPH0, SPH1, SPH2 and MPS. The accuracy of SPH0 is very high.

Table 2. Classification of mesh.

Name	$\mathrm{d}x_i$
Regular	$\mathrm{d}x_i=\dfrac{L}{N}$
Algebraic	$\mathrm{d}x_i=\alpha\left(\dfrac{L}{N}+i\dfrac{L}{100}\right),\quad \sum_{i=0}^{i=N-1}\mathrm{d}x_i=L$
Geometric	$\mathrm{d}x_i=\alpha\dfrac{L}{N}1.05^{i-1},\quad \sum_{i=0}^{i=N-1}\mathrm{d}x_i=L$
Random	$\mathrm{d}x_i=\alpha\dfrac{L}{N}\left(1+0.0625\,\mathrm{drand}(i)\right),\quad \sum_{i=0}^{i=N-1}\mathrm{d}x_i=L$
Sinusoidal	$\mathrm{d}x_i=\alpha\dfrac{L}{N}\left(1+0.5\sin\left(4\pi\dfrac{i}{N}\right)\right),\quad \sum_{i=0}^{i=N-1}\mathrm{d}x_i=L$

Table 3. Classification of weight.

Name	Weight
SPH0 … Gauss	$w(\lvert x\rvert)=\dfrac{1}{\sqrt{2\pi\gamma}}\exp\left(-\dfrac{\lvert x\rvert^2}{2\gamma^2}\right),\quad \lvert x\rvert<\infty$
SPH1 … Lucy	$w(\lvert x\rvert)=\begin{cases}\dfrac{5}{4}\dfrac{1}{\sigma}\left(1+3\dfrac{\lvert x\rvert}{\sigma}\right)\left(1-\dfrac{\lvert x\rvert}{\sigma}\right)^3, & \lvert x\rvert\le\sigma\\ 0, & \lvert x\rvert>\sigma\end{cases}$
SPH2 … Opt.kernel	$w(\lvert x\rvert)=\dfrac{4}{3h}\begin{cases}1-6\left(\dfrac{\lvert x\rvert}{h}\right)^2+6\left(\dfrac{\lvert x\rvert}{h}\right)^3, & 0\le\lvert x\rvert<0.5h\\ 2\left(1-\dfrac{\lvert x\rvert}{h}\right)^3, & 0.5h\le\lvert x\rvert<h\\ 0, & h\le\lvert x\rvert\end{cases}$
MPS	$w(\lvert x\rvert)=\begin{cases}h/\lvert x\rvert-1, & 0\le\lvert x\rvert<h\\ 0, & h\le\lvert x\rvert\end{cases}$

Table 4. Summary on dϕ/dx (○: good, △: not good, ×: bad).

	Regular	Algebraic	Geometric	Random	Sinusoidal
SPH0	○	○	○	○	△
SPH1	○	△	△	○	×
SPH2	○	△	△	○	×
MPS	○	○	×	○	○

Table 5. Summary on d$^2\phi$/dx^2 (○: good, △: not good, ×: bad).

	Regular	Algebraic	Geometric	Random	Sinusoidal
SPH0	○	○	○	△	△
SPH1	△	△	△	△	×
SPH2	○	△	△	△	×
MPS	○	×	×	○	×

6.2.2. Algebraic Mesh
As shown in **Figure 8**, a big error occurs in $\mathrm{d}^2\phi/\mathrm{d}x^2$ of MPS.

6.2.3. Geometric Mesh
As shown in **Figure 9**, a big error occurs in $\mathrm{d}^2\phi/\mathrm{d}x^2$ of MPS.

6.2.4. Random Mesh
As shown in **Figure 10**, the errors are rather small in all methods SPH0, SPH1, SPH2 and MPS not only for $\mathrm{d}\phi/\mathrm{d}x$ but also for $\mathrm{d}^2\phi/\mathrm{d}x^2$. The errors in MPS are surprisingly small.

6.2.5. Sinusoidal Mesh
As shown in **Figure 11**, the errors are rather big in all methods SPH0, SPH1, SPH2 and MPS not only for $\mathrm{d}\phi/\mathrm{d}x$ but also for $\mathrm{d}^2\phi/\mathrm{d}x^2$ except $\mathrm{d}\phi/\mathrm{d}x$ in MPS. The errors in SHP0 are smaller than those in the other methods.

6.2.6. Summary of Results
From the above mentioned numerical results, the following summaries are obtained.

6.3. Application to Solution of Initial Value Problem

6.3.1. 1D Fluid Motion without Pressure and Viscosity
The motion of vast number of particles distributed in space under the action of the gravitational force may be treated as a fluid motion without pressure and viscosity [9] [10]:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0, \tag{72a}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = -\frac{\partial \Pi}{\partial x}, \tag{72b}$$

$$\frac{\partial^2 \Pi}{\partial x^2} = 4\pi G \rho, \tag{72c}$$

where ρ, u and Π is the density, velocity and gravitational potential. G is the gravitational constant.

The solution of the problem defined above by Eulerian method is given as follows:

1) At time t, assume ρ, u and Π are given.

2) $\partial\rho/\partial t$ and $\partial u/\partial t$ are obtained from Equations (72a) and (72b), and Π is obtained from Equation (72c).

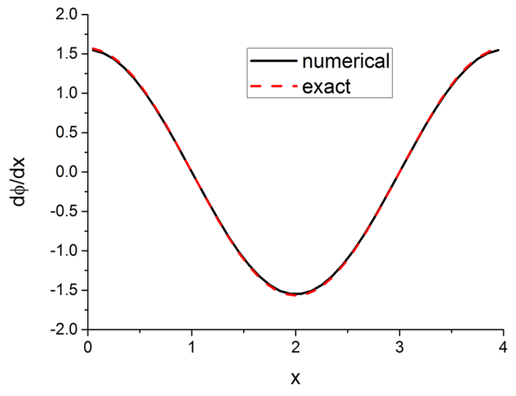
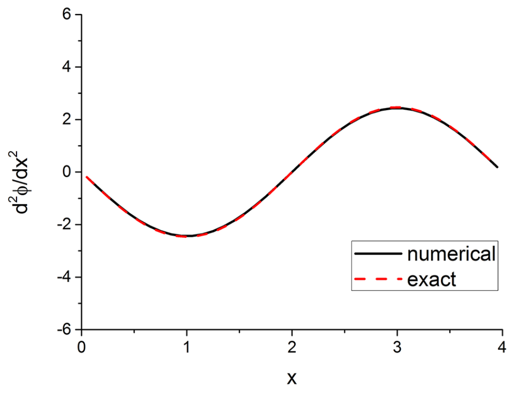
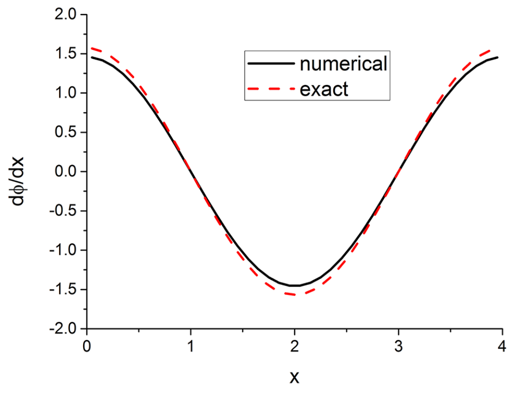
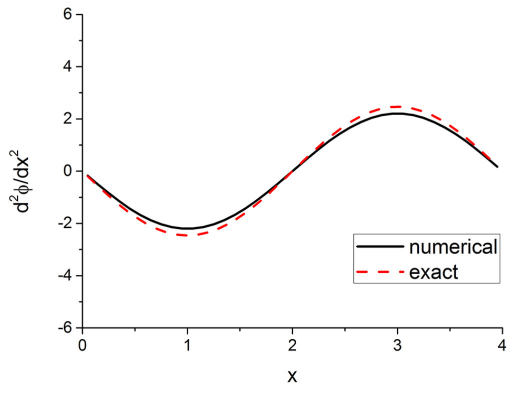
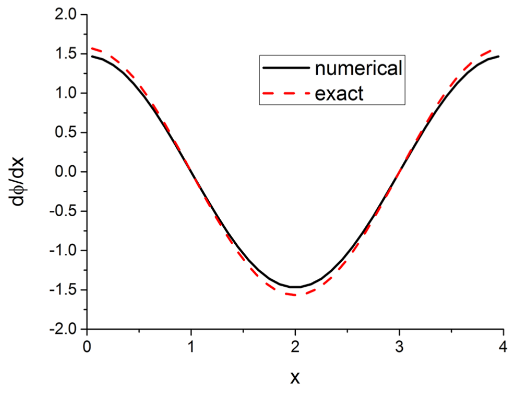
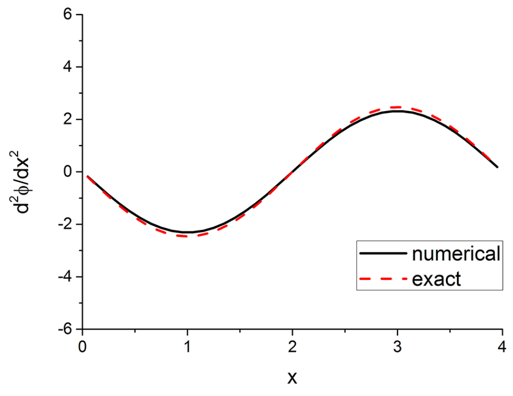
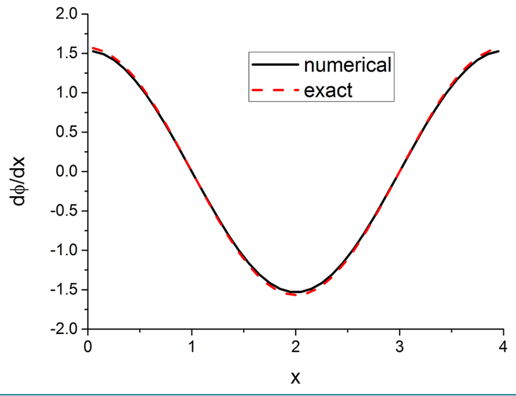
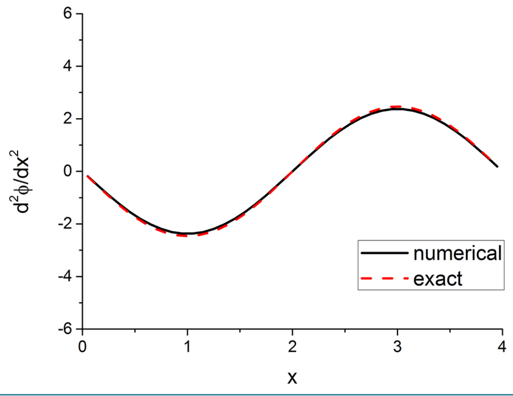
	$d\phi/dx$	$d^2\phi/dx^2$
SPH0		
SPH1		
SPH2		
MPS		

Figure 7. Regular mesh.

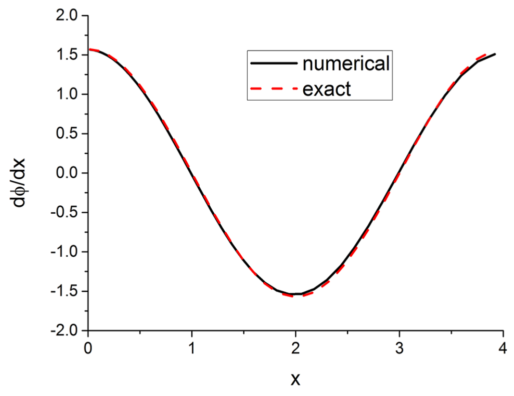
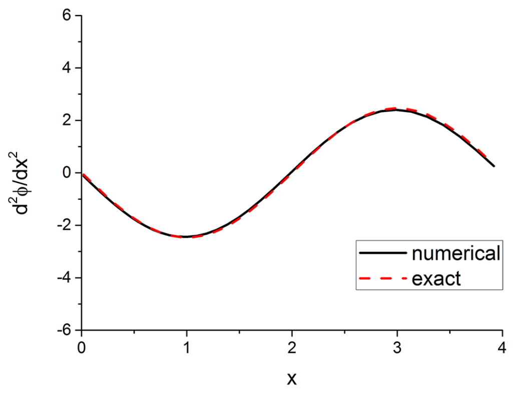
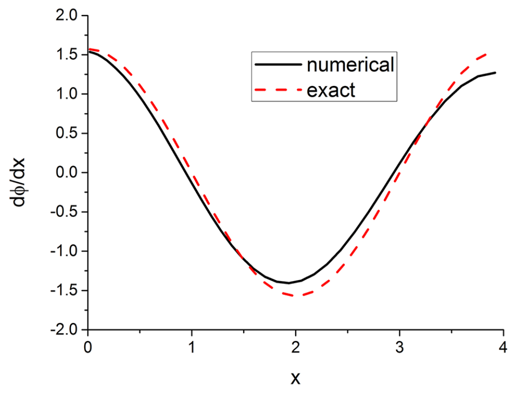
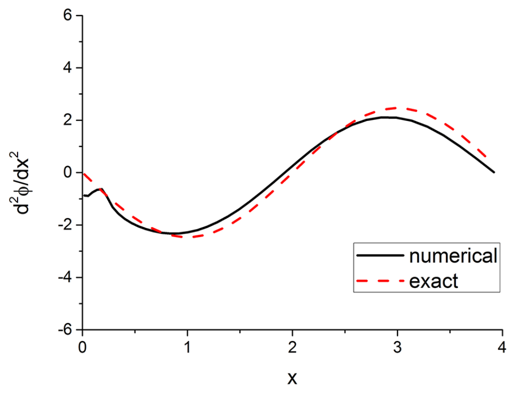
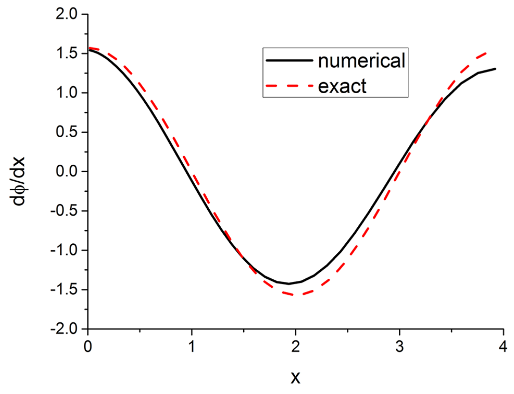
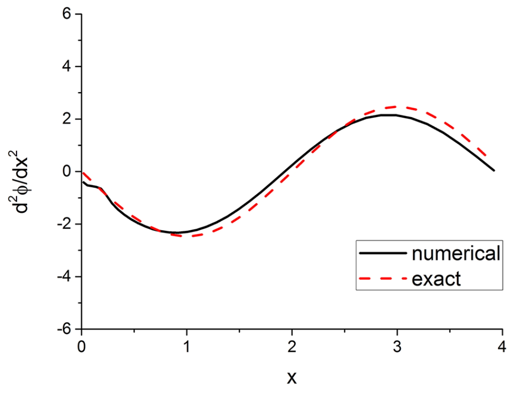
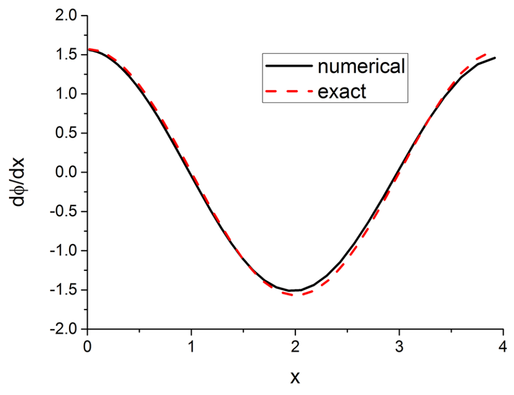
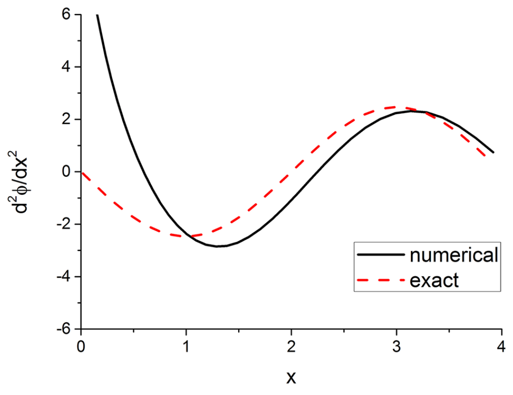
	$d\phi/dx$	$d^2\phi/dx^2$
SPH0		
SPH1		
SPH2		
MPS		

Figure 8. Algebraic mesh.

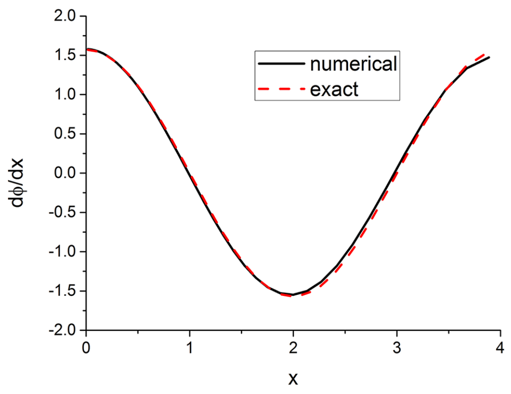
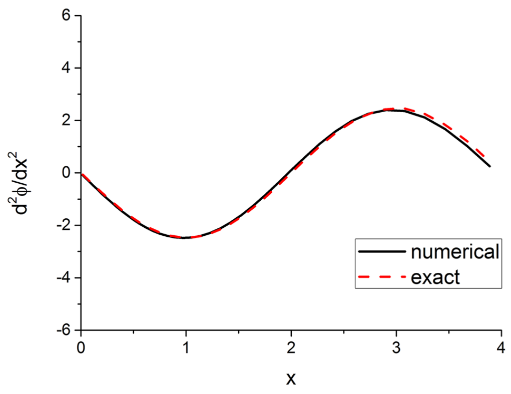
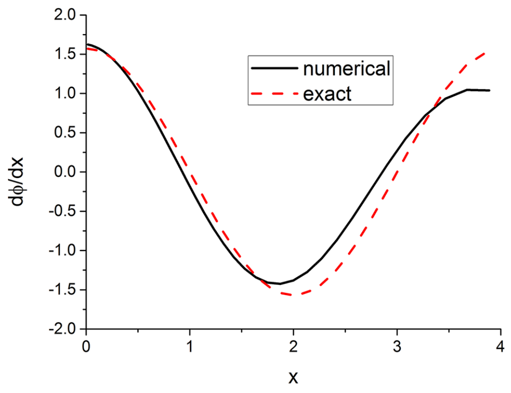
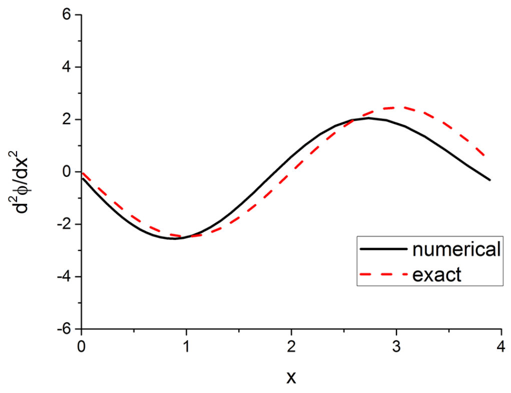
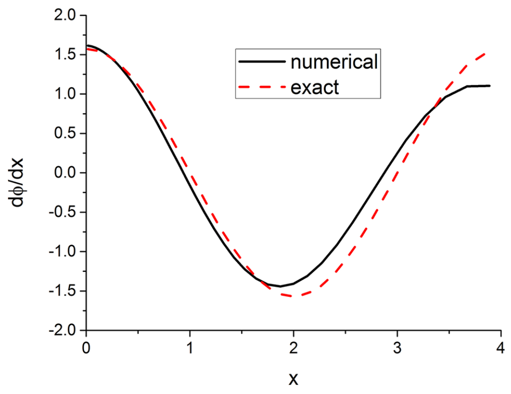
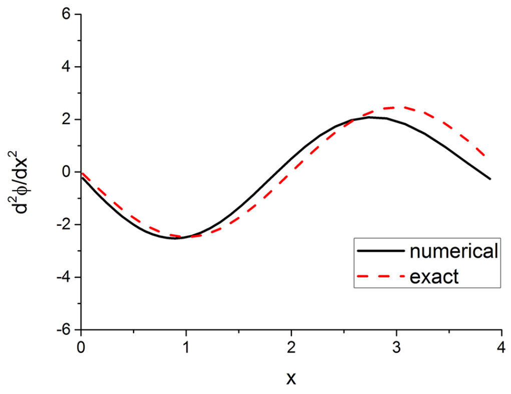
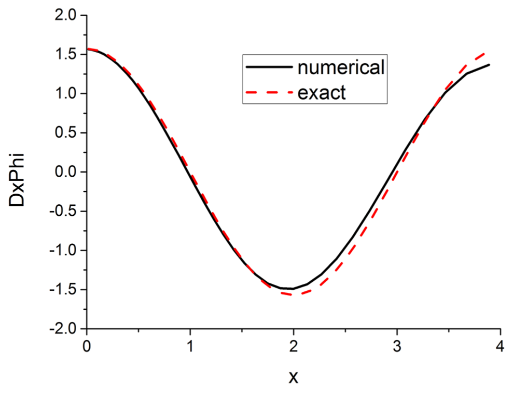
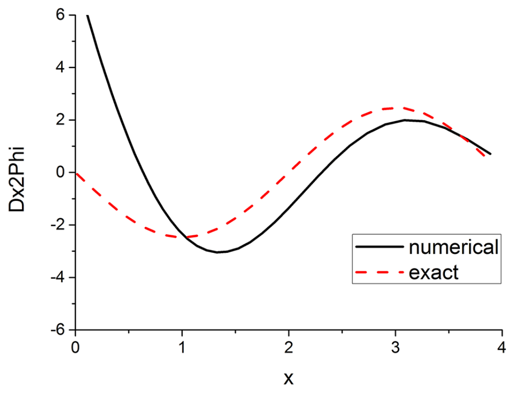
	$\mathrm{d}\phi/\mathrm{d}x$	$\mathrm{d}^2\phi/\mathrm{d}x^2$
SPH0		
SPH1		
SPH2		
MPS		

Figure 9. Geometric mesh.

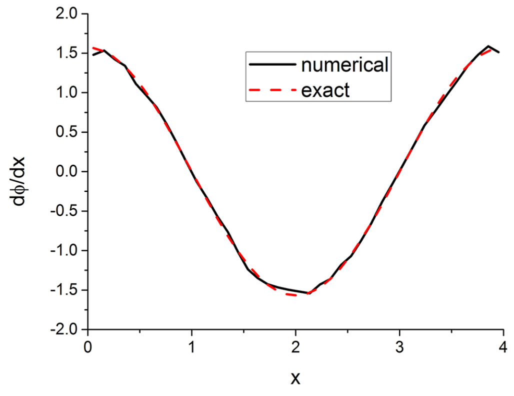
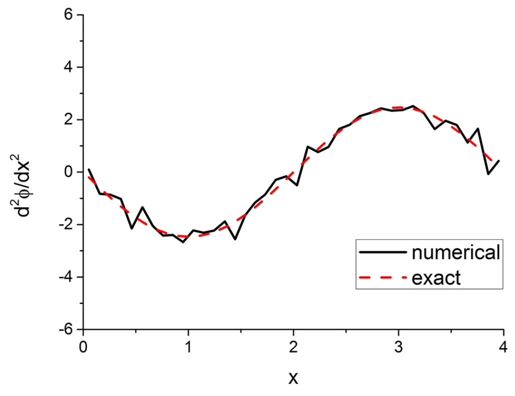
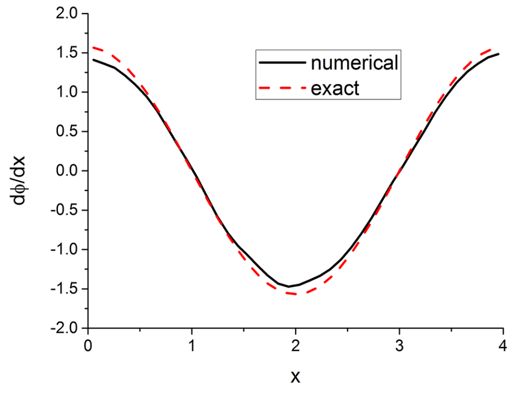
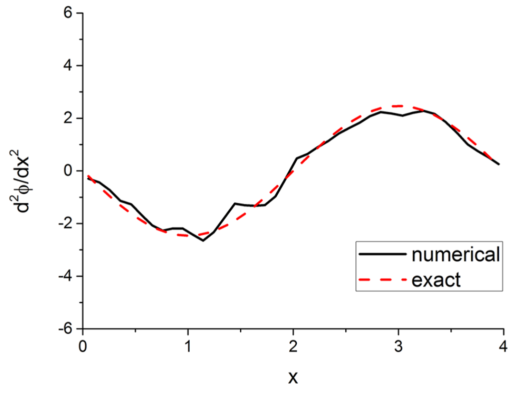
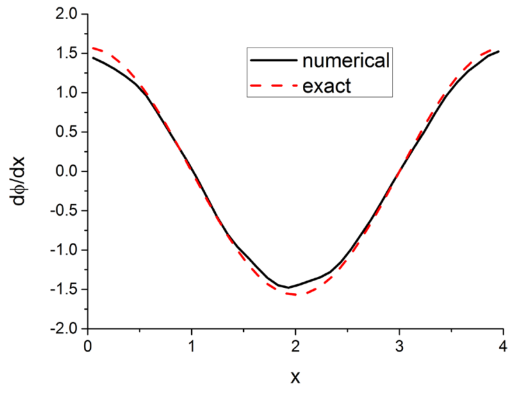
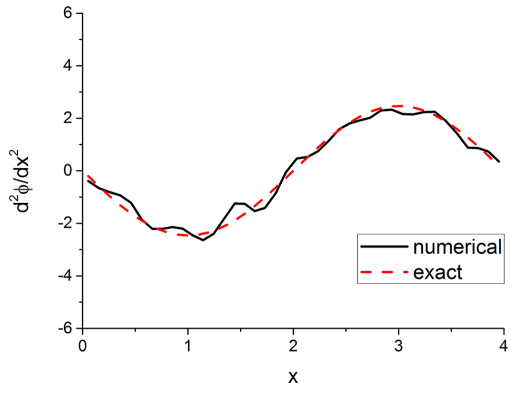
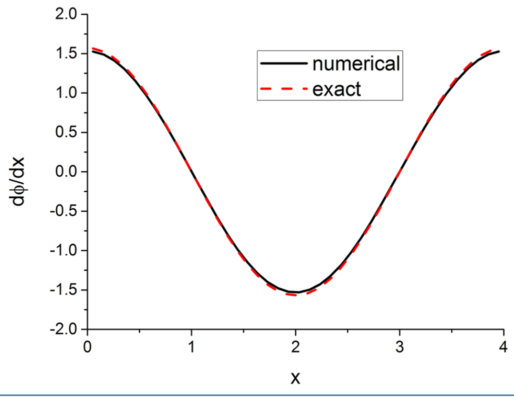
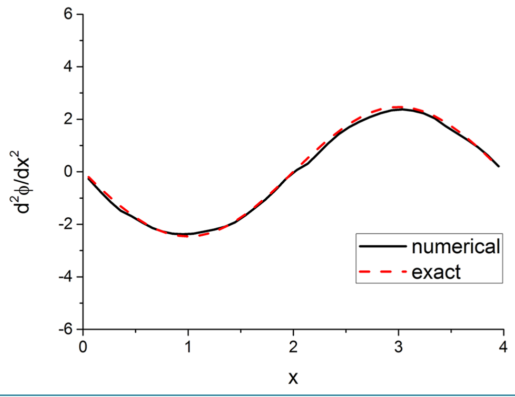
	$d\phi/dx$	$d^2\phi/dx^2$
SPH0		
SPH1		
SPH2		
MPS		

Figure 10. Random mesh.

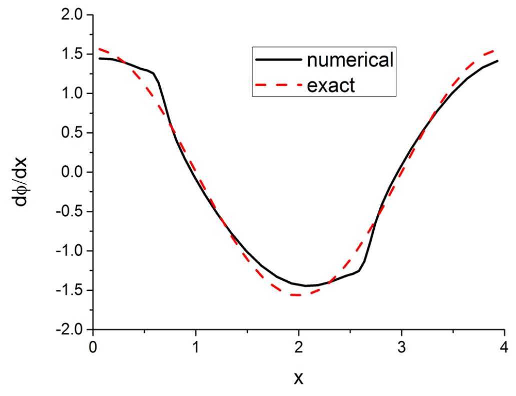
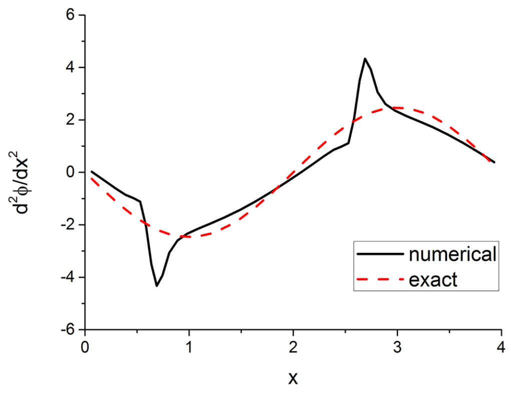
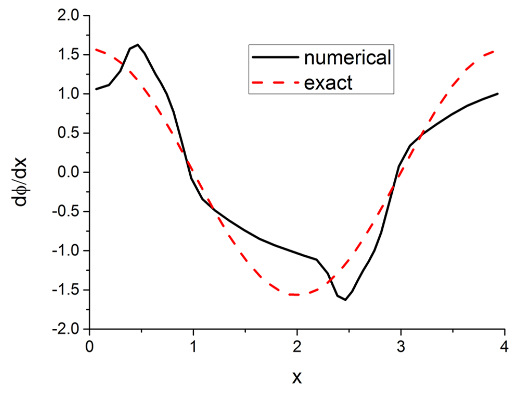
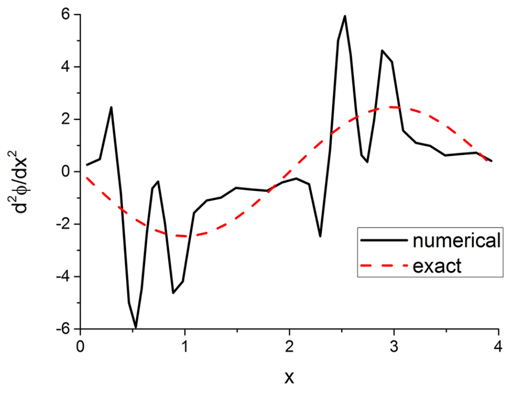
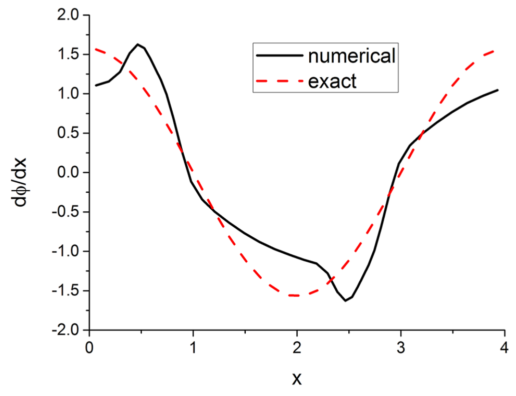
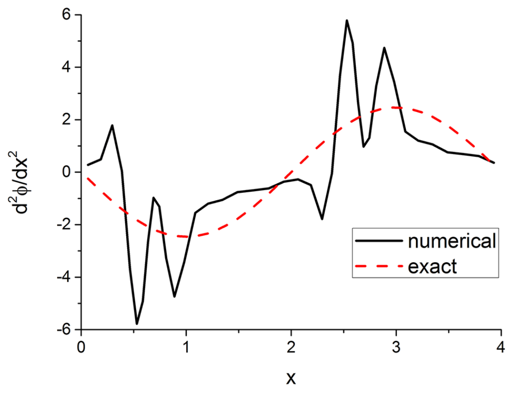
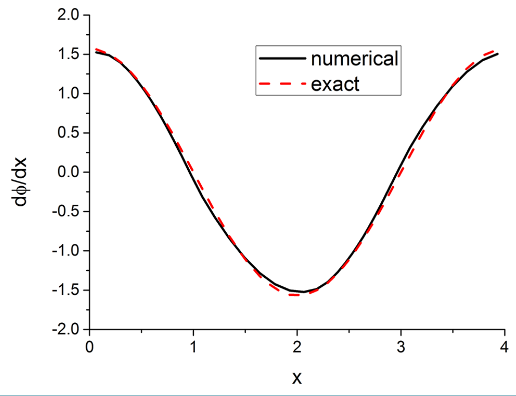
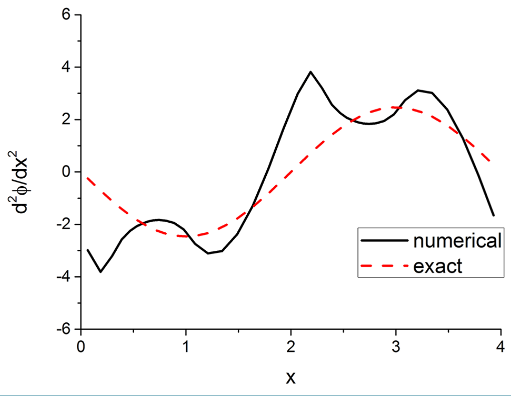
	$d\phi/dx$	$d^2\phi/dx^2$
SPH0		
SPH1		
SPH2		
MPS		

Figure 11. Sinusoidal mesh.

3) ρ, and u at time $t+\mathrm{d}t$ is calculated.

4) Repeat this process.

In the solution by Lagrangian method, first, the equations in Eulerian form are transformed into Laglangian form:

$$\frac{Dx}{Dt} = u, \tag{73a}$$

$$\frac{Du}{Dt} \equiv \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = -\frac{\partial \Pi}{\partial x}, \tag{73b}$$

$$\frac{D\rho}{Dt} \equiv \frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} = -\rho\frac{\partial u}{\partial x}, \tag{73c}$$

$$\frac{\partial^2 \Pi}{\partial x^2} = 4\pi G\rho. \tag{73d}$$

The following procedures give the solution of the problem defined above by Lagrangian method:

1) At time t, assume ρ, u and Π are given.

2) Dx/Dt, $D\rho/Dt$ and Du/Dt is obtained from Equations (73a), (73b) and (73c), and Π is obtained from Equation (73d).

3) x, ρ and u of the material point at time $t+\mathrm{d}t$ is calculated.

4) Repeat this process.

If x_i is obtained, then, $\mathrm{d}x_i$ and ρ_i is obtained as shown below:

$$\mathrm{d}x_i = \frac{1}{2}(x_{i+1} - x_i) + \frac{1}{2}(x_i - x_{i-1}) = \frac{1}{2}(x_{i+1} - x_{i-1}) \tag{74a}$$

and

$$\rho_i = \frac{M_i}{\mathrm{d}x_i}. \tag{74b}$$

Hence, from the theoretical viewpoint, this problem can be solved without using the gradient operator. However, we use the gradient operator to obtain ρ_i using the continuity equation.

1) Trapezoidal Distribution of the Initial Density

The initial conditions are given by

$$x_i = x_{0i} = (i+0.5)L/N, \tag{75a}$$

$$\rho_i = \begin{cases} 1 + \dfrac{10}{L}(x_i - 0.35L) + 0.025 & \text{when } 0.25L \le x_i < 0.35L \\ 1 + 0.025 & \text{when } 0.35L \le x_i < 0.65L \\ 1 - \dfrac{10}{L}(x_i - 0.65L) + 0.025 & \text{when } 0.65L \le x_i < 0.75L \\ 0 + 0.025 & \text{otherwise,} \end{cases} \tag{75b}$$

$$u_i = 0. \tag{75c}$$

The boundary conditions are specified as

$$\rho_0 = \rho_1,\ \rho_{N-1} = \rho_{N-2}; \tag{76a}$$

$$u_0 = u_1,\ u_{N-1} = u_{N-2}. \tag{76b}$$

The computational conditions are as follows:

$$L = 4,\ N = 41,\ \nu = 0.0,\ \mathrm{d}t = 0.00025,\ t_{end} = 30000\mathrm{d}t,\ G = 0.0015,$$

$$\gamma_i = \mathrm{d}x_i \text{ for SPH0},\ h_i = 2\mathrm{d}x_i \text{ for MPS},\ N_{end\_cor} = 8 \text{ for SPH0 and 2 for MPS}.$$

The numerical results are shown in **Figure 12**. The FDM (Finite Difference Method) uses the central differ-

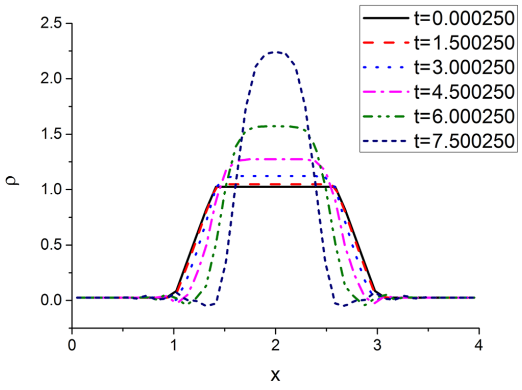
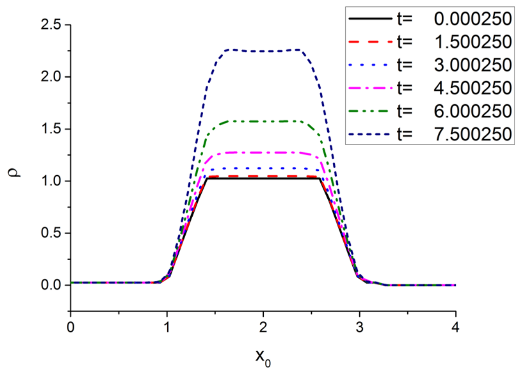
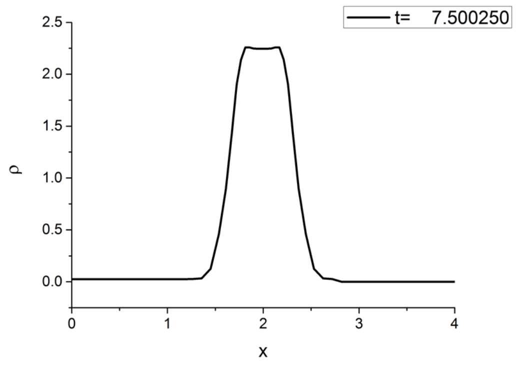
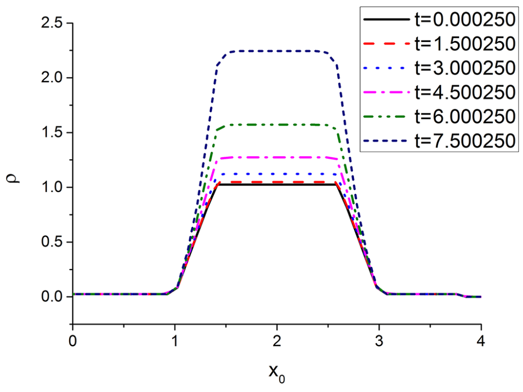
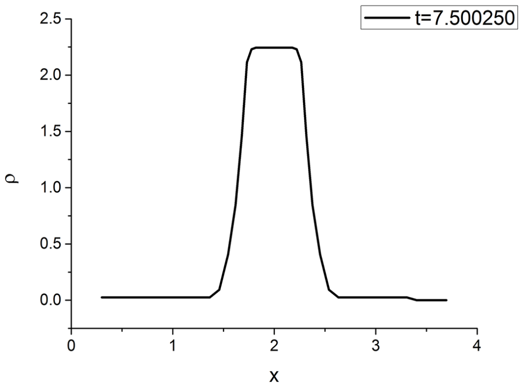
	ρ in x_0	ρ in x
FDM		
SPH0		
MPS		

Figure 12. Trapezoidal distribution of the initial density.

ences for the first and second space derivatives, and the precision of FDM is considered high. The Eulerian solution is used for FDM, and the Lagrangian solution is used for SPH0 and MPS. The distribution pattern of MPS is slightly different from those of FDM and SPH0.

2) Rectangular Distribution of the Initial Density with *G* = 0.001

The initial conditions are given by

$$x_i = x_{0i} = (i+0.5)L/N\ , \tag{77a}$$

$$\rho_i = \begin{cases} 0+0.5 & \text{when } \ x_i < 0.25L \\ 1+0.5 & \text{when } \ 0.25L \le x_i \le 0.75L \\ 0+0.5 & \text{when } \ 0.75L \le x_i, \end{cases} \tag{77b}$$

$$u_i = 0\,. \tag{77c}$$

The boundary conditions are specified as

$$\rho_0 = \rho_1,\ \ \rho_{N-1} = \rho_{N-2}; \tag{78a}$$

$$u_0 = u_1,\ \ u_{N-1} = u_{N-2}. \tag{78b}$$

The computational conditions are as follows:

$$L = 4\,,\quad N = 41\,,\quad \nu = 0.0\,,\quad \mathrm{d}t = 0.00025\,,\quad t_{end} = 30000\mathrm{d}t\,,\quad G = 0.001\,,$$

$$\gamma_i = \mathrm{d}x_i \text{ for SPH0}\,,\quad h_i = 2\mathrm{d}x_i \text{ for MPS}\,,\quad N_{end\_cor} = 8 \text{ for SPH0 and 2 for MPS}\,.$$

The numerical results are shown in **Figure 13**. The FDM uses the central differences for the first and second space derivatives, and the precision of FDM is considered high, if we neglect the spurious oscillation. The distribution pattern of MPS is slightly different from those of FDM and SPH0.

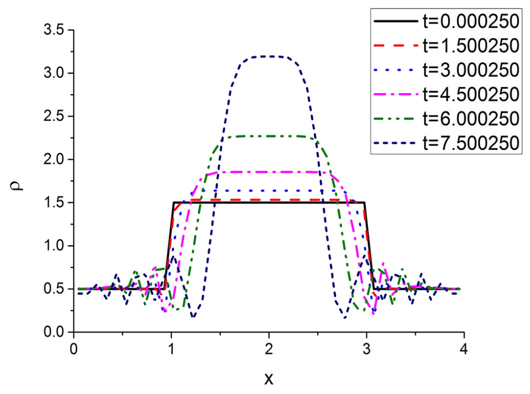
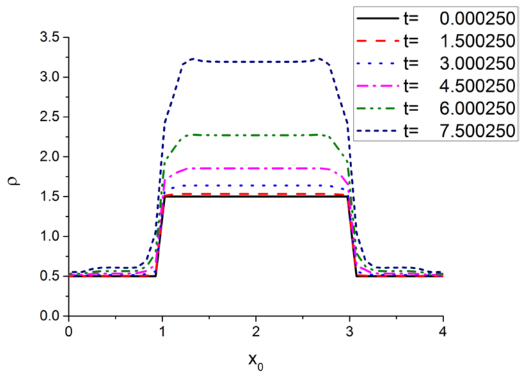
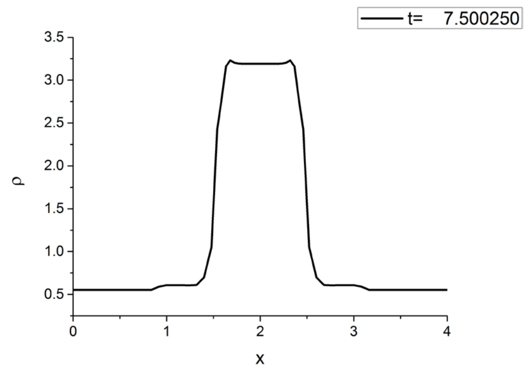
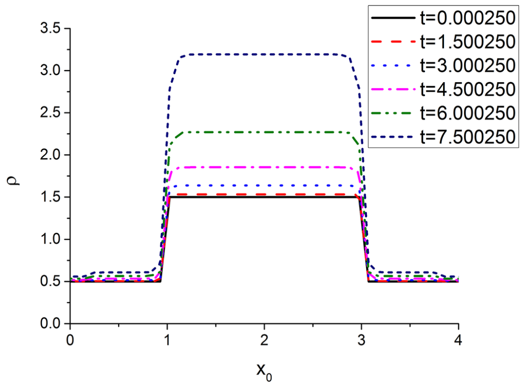
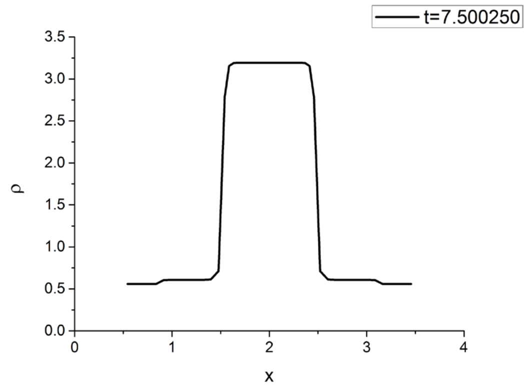
	ρ in x_0	ρ in x
FDM		
SPH0		
MPS		

Figure 13. Rectangular distribution of the initial density with G = 0.001.

3) Rectangular Distribution of the Initial Density with *G* = 0.0015

The initial conditions are given by

$$x_i = x_{0i} = (i+0.5)L/N, \tag{79a}$$

$$\rho_i = \begin{cases} 0+0.025 & \text{when } x_i < 0.35L \\ 1+0.025 & \text{when } 0.35L \le x_i < 0.75L, \\ 0+0.025 & \text{when } 0.35L \le x_i \end{cases} \tag{79b}$$

$$u_i = 0. \tag{79c}$$

The boundary condition are specified as

$$\rho_0 = \rho_1, \ \rho_{N-1} = \rho_{N-2}; \tag{80a}$$

$$u_0 = u_1, \ u_{N-1} = u_{N-2}. \tag{80b}$$

Computational condition

$$L = 4, \ N = 41, \ \nu = 0.0, \ \mathrm{d}t = 0.00025, \ t_{end} = 30000\mathrm{d}t, \ G = 0.0015,$$

$$\gamma_i = \mathrm{d}x_i \text{ for SPH0}, \ h_i = 2\mathrm{d}x_i \text{ for MPS}, \ N_{end\_cor} = 8 \text{ for SPH0 and 2 for MPS}.$$

The numerical results are shown in **Figure 14**. The FDM uses the central differences for the first and second space derivatives, and the precision of FDM is considered high, if we neglect the spurious oscillation. The distribution pattern of MPS is different from those of FDM and SPH0.

6.3.2. 1D Fluid Motion without Pressure but with Viscosity

In the present section, the viscosity is introduced:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0, \tag{81a}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = -\frac{\partial \Pi}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2}, \tag{81b}$$

$$\frac{\partial^2 \Pi}{\partial x^2} = 4\pi G\rho, \tag{81c}$$

where ν is the kinematic viscosity.

Since the discrete Laplacian operator of SPH0 generates a big error at the discontinuity, the initial density distribution was smoothened using the five point running average, and, in the second example below, an small artificial numerical viscosity μ = 0.0000005 in case of N = 41 was added at every time step:

$$\rho_i \to \rho_i + \mu\frac{1}{\mathrm{d}x_i^2}(\rho_{i+1} - 2\rho_i + \rho_{i-1}), \tag{82a}$$

$$u_i \to u_i + \mu\frac{1}{\mathrm{d}x_i^2}(u_{i+1} - 2u_i + u_{i-1}). \tag{82b}$$

In the third examples below, a big difference has occurred between SPH and MPS solutions.

1) Exponential Distribution of the Initial Density

The initial conditions are given by

$$x_i = x_{0i} = (i+0.5)L/N, \tag{83a}$$

$$\rho_i = \exp\left(-\left(\frac{x_i - 0.5L}{0.2L}\right)^2\right), \tag{83b}$$

$$u_i = 0. \tag{83c}$$

The boundary conditions are specified as

$$\rho_0 = \rho_1, \ \rho_{N-1} = \rho_{N-2}; \tag{84a}$$

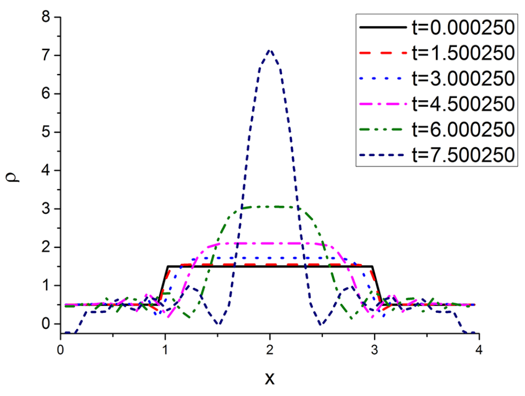
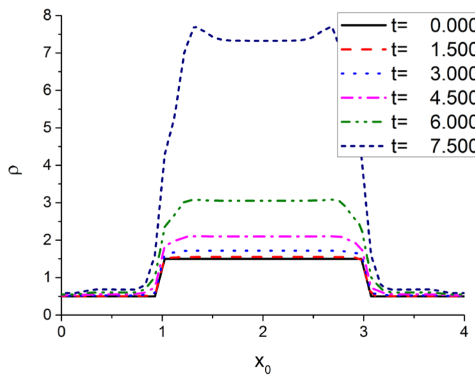
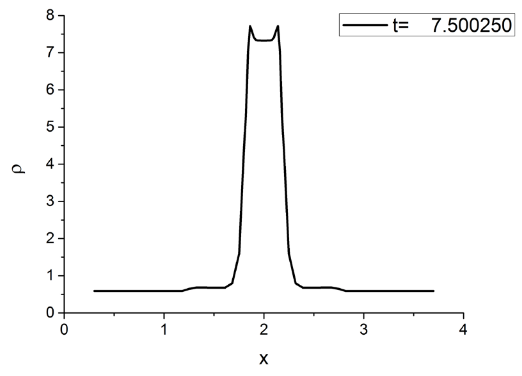
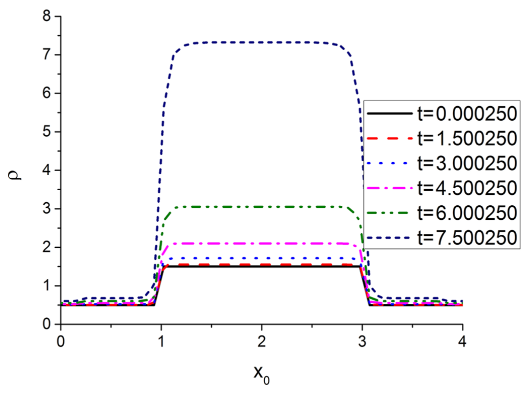
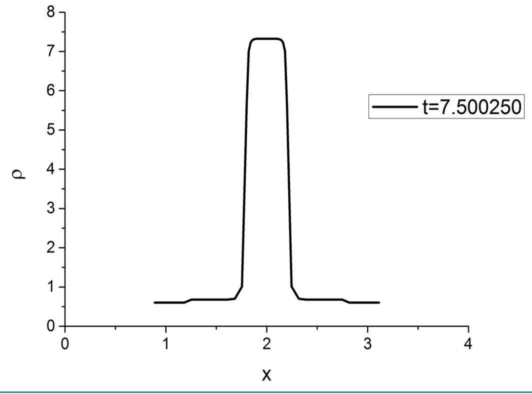
	ρ in x_0	ρ in x
FDM		
SPH0		
MPS		

Figure 14. Rectangular distribution of the initial density with $G = 0.0015$.

$$u_0 = u_1,\ u_{N-1} = u_{N-2}. \tag{84b}$$

The computational conditions are as follows:

$$L = 4,\quad N = 41,\quad \nu = 0.1,\quad dt = 0.00025,\quad t_{end} = 20000dt,\quad \gamma_i = \mathrm{d}x_i \text{ for SPH0},$$
$$h_i = 2\mathrm{d}x_i \text{ for MPS},\quad G = 0.005,\quad N_{end\_cor} = 8 \text{ for SPH0 and 2 for MPS}.$$

The numerical results are shown in **Figure 15**. The FDM uses the central differences for the first and second space derivatives, and the precision of FDM is considered high. In this example, the FDM, SPH and MPS solutions becomes similar.

2) The Trapezoidal Distribution of the Initial Density

The initial conditions are given by

$$x_i = x_{0i} = (i + 0.5) L/N, \tag{85a}$$

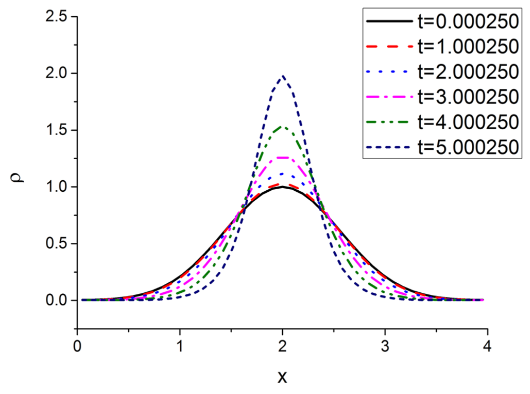
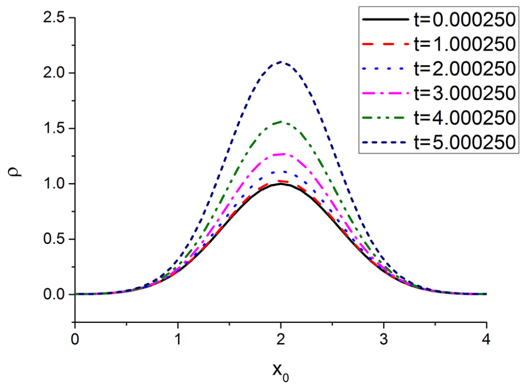
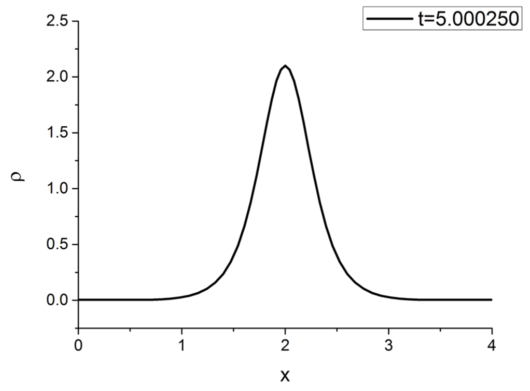
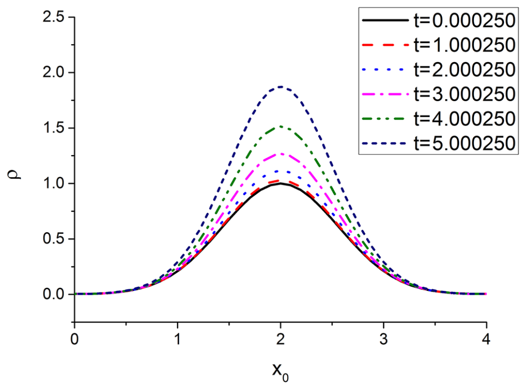
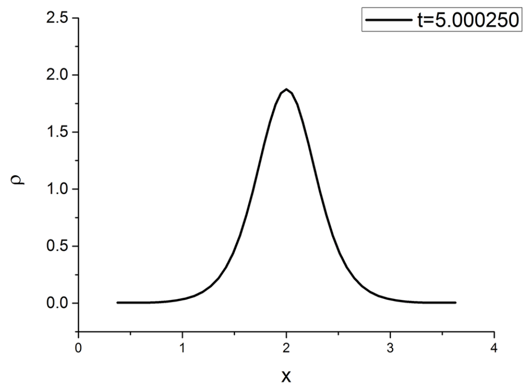
	ρ in x_0	ρ in x
FDM		
SPH0		
MPS		

Figure 15. Exponential distribution of the initial density.

$$\rho_i = \begin{cases} 1+\dfrac{10}{L}\left(x_i-0.35L\right)+0.025 & \text{when } 0.25L \le x_i < 0.35L \\ 1+0.025 & \text{when } 0.35L \le x_i < 0.65L \\ 1-\dfrac{10}{L}\left(x_i-0.65L\right)+0.025 & \text{when } 0.65L \le x_i < 0.75L \\ 0+0.025 & \text{otherwise,} \end{cases} \tag{85b}$$

$$u_i = 0\,, \tag{85c}$$

where the initial density distribution was smoothened using the five point running average:

$$\frac{1}{5}\left(\rho_{i-2}+\rho_{i-1}+\rho_i+\rho_{i+1}+\rho_{i+2}\right) \to \rho_i\,. \tag{86}$$

The boundary conditions are specified as

$$\rho_0 = \rho_1,\ \rho_{N-1} = \rho_{N-2}; \tag{87a}$$

$$u_0 = u_1,\ u_{N-1} = u_{N-2}. \tag{87b}$$

The computational conditions are as follows:

$$L = 4,\quad N = 41,\quad \nu = 0.02,\quad \mathrm{d}t = 0.00025,\quad t_{end} = 30000\mathrm{d}t,\quad G = 0.0015,$$

$$\gamma_i = \mathrm{d}x_i \text{ for SPH0},\quad h_i = 2\mathrm{d}x_i \text{ for MPS},\quad N_{end\_cor} = 4 \text{ for SPH0 and 2 for MPS}.$$

The numerical results are shown in **Figure 16**. The FDM uses the central differences for the first and second space derivatives, and the precision of FDM is considered high. In this example, small difference is observed in the FDM, SPH and MPS solutions.

3) The Flat-Slope-Flat Distribution of the Initial Velocity

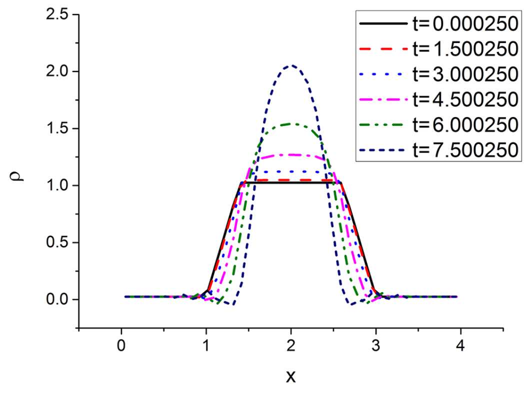
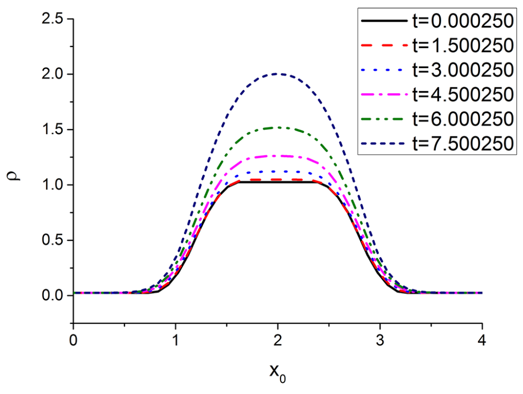
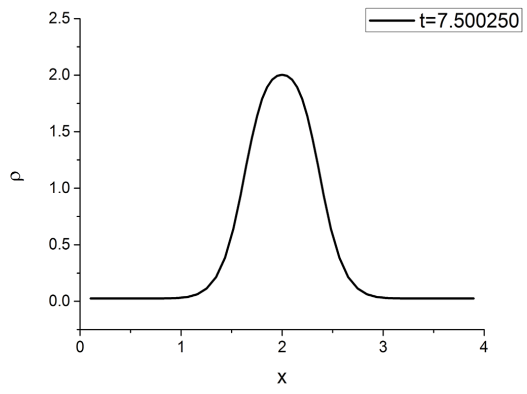
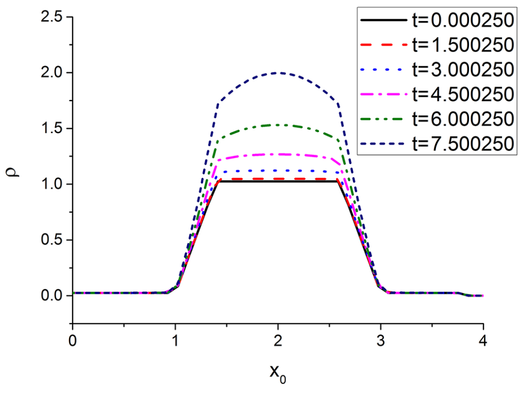
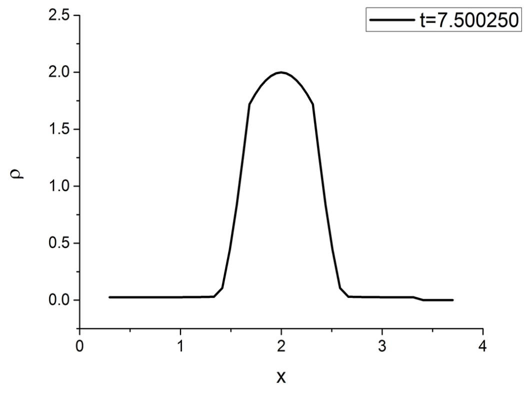
	ρ in x_0	ρ in x
FDM		
SPH0		
MPS		

Figure 16. Trapezoidal distribution of the initial density.

The initial conditions are given by

$$x_i = x_{0i} = (i+0.5)L/N, \tag{88a}$$

$$\rho_i = 1, \tag{88b}$$

$$u_i = \begin{cases} 0.5 & \text{when } x_i < 0.25L \\ -\frac{2}{L}(x_i - 0.5L) & \text{when } 0.25L \le x_i \le 0.75L \\ -0.5 & \text{when } 0.75L \le x_i \end{cases} \tag{88c}$$

The boundary conditions are specified as

$$\rho_0 = 1,\ \rho_{N-1} = 1; \tag{89a}$$

$$u_0 = 0.5,\ u_{N-1} = -0.5. \tag{89b}$$

The computational conditions are as follows:

$$L = 4,\ N = 41,\ \nu = 0.02,\ \mathrm{d}t = 0.00025,\ t_{end} = 10000\mathrm{d}t,\ G = 0,\ \gamma_i = \mathrm{d}x_i \text{ for SPH0},$$
$$h_i = 2\mathrm{d}x_i \text{ for MPS},\ N_{end\_cor} = 8 \text{ for SPH0 and 2 for MPS}.$$

Since we assume $G = 0$ in this example, the equation becomes Burgers equation.

a) Comparison of FDM Solution with the Analytical One

The central differences were used for the first and second space derivatives. As shown in **Figure 17**. The FDM solution is a very good approximation of the analytical solution [11].

b) Comparison of FDM, SPH0 and MPS Solutions

Figure 18 and **Figure 19** show the comparisons of ρ and u among FDM, SPH0 and MPS solutions. SPH0 and MPS solutions do not give good approximations. With respect to the velocity u, the difference between SPH0 and MPS is big.

7. Modified Gaussian Weight

Gaussian-type weights of finite support with C1 continuity are given as follows:

$$w(x) = \begin{cases} \alpha \exp\left(-\frac{x^2}{2\gamma^2}\right)\left(1-\frac{x^2}{h^2}\right)^2 = \alpha\left(1-\frac{2x^2}{h^2}+\frac{x^4}{h^4}\right)\exp\left(-\frac{x^2}{2\gamma^2}\right) \\ \alpha \exp\left(-\frac{x^2}{2\gamma^2}\right)\cos^2\left(\frac{\pi x}{2h}\right), \end{cases} \tag{90a), (90b}$$

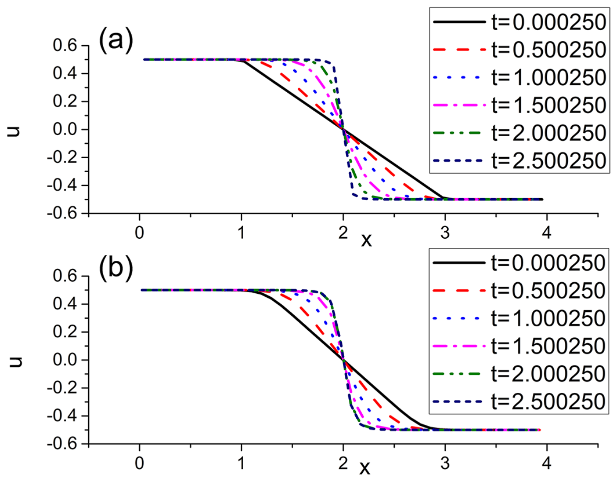


Figure 17. Comparison of FDM solution with the analytical one ((a) FDM solution; (b) Exact solution).

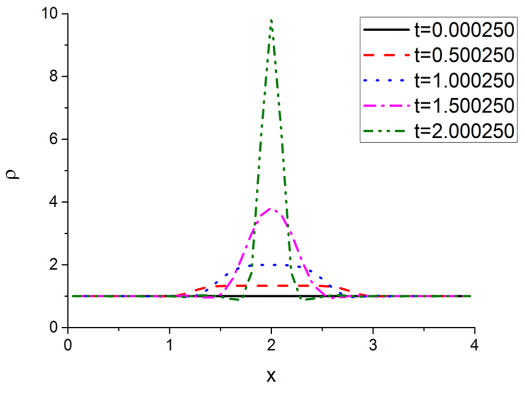
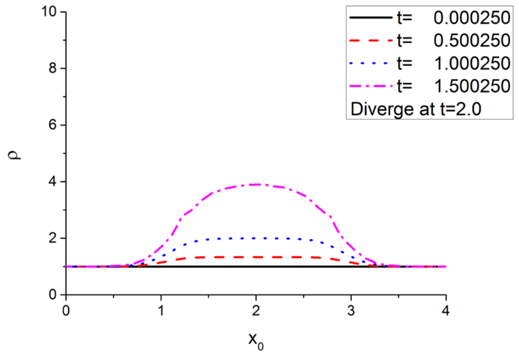
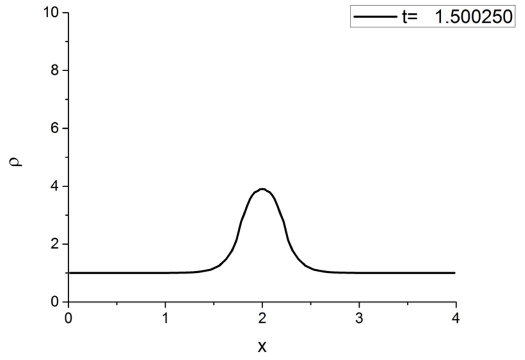
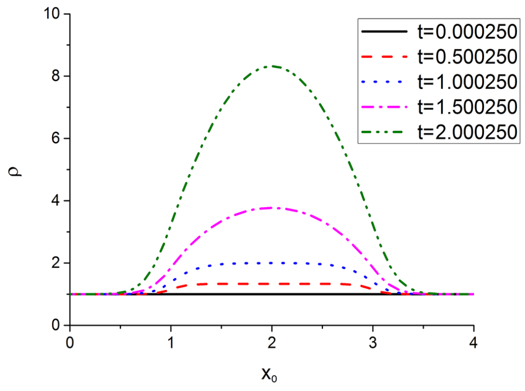
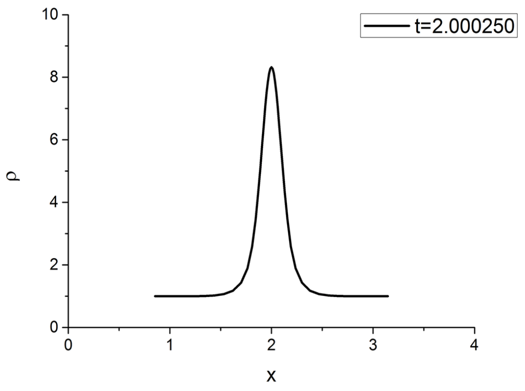
	ρ in x_0	ρ in x
FDM		
SPH0		
MPS		

Figure 18. Comparison of density ρ.

where α satisfies

$$1=\int_{-h}^{h} w(x)\,\mathrm{d}x = 2\int_0^h w(x)\,\mathrm{d}x = \begin{cases} 2\alpha\int_0^h \exp\left(-\dfrac{x^2}{2\gamma^2}\right)\left(1-\dfrac{x^2}{h^2}\right)^2 \mathrm{d}x \\ 2\alpha\int_0^h \exp\left(-\dfrac{x^2}{2\gamma^2}\right)\cos^2\left(\dfrac{\pi x}{2h}\right)\mathrm{d}x \end{cases} = \begin{cases} 2\alpha h\int_0^1 \exp\left(-\dfrac{h^2x^2}{2\gamma^2}\right)\left(1-x^2\right)^2 \mathrm{d}x \\ 2\alpha h\int_0^1 \exp\left(-\dfrac{h^2x^2}{2\gamma^2}\right)\cos^2\left(\dfrac{\pi x}{2}\right)\mathrm{d}x. \end{cases} \tag{91}$$

The first and the second derivatives of $w(x)$ given by Equation (90a) are given by

$$w'(x)=\alpha\left(-\left(\frac{4}{h^2}+\frac{1}{\gamma^2}\right)x+\left(\frac{4}{h^4}+\frac{2}{\gamma^2h^2}\right)x^3-\frac{x^5}{\gamma^2h^4}\right)\exp\left(-\frac{x^2}{2\gamma^2}\right), \tag{92a}$$

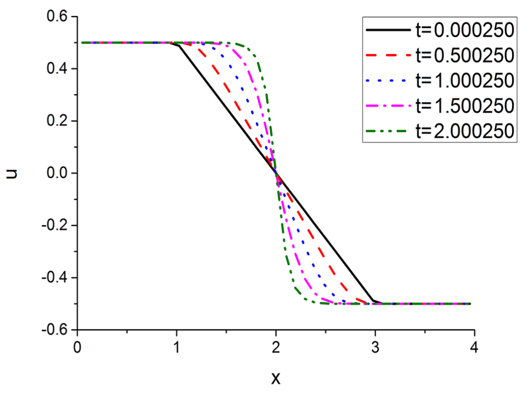
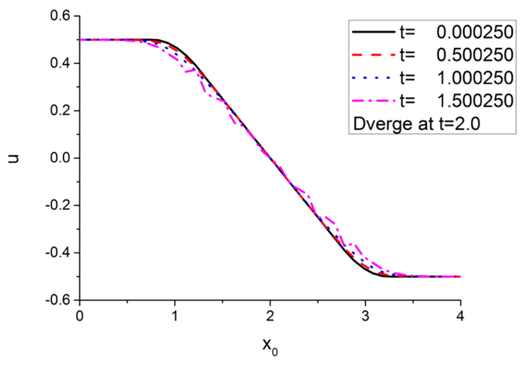
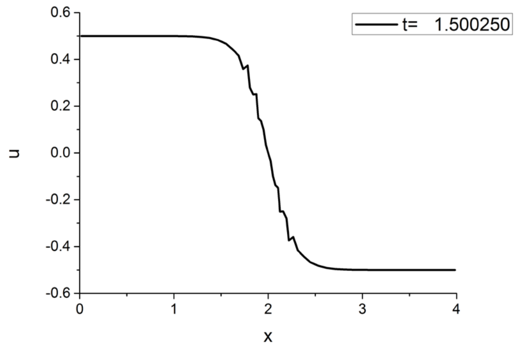
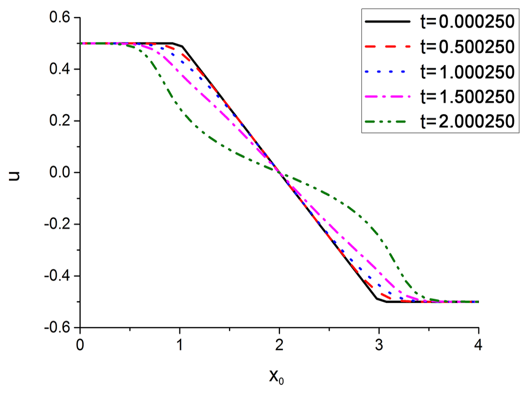
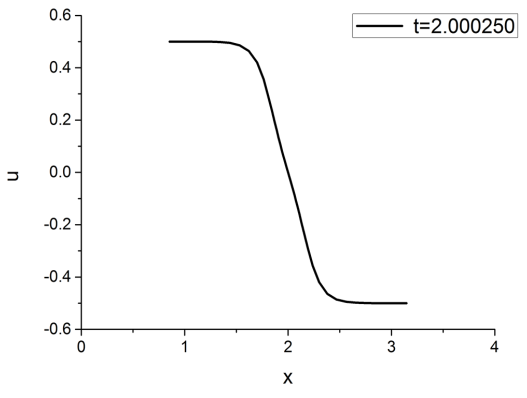
	u in x_0	u in x
FDM		
SPH0		
MPS		

Figure 19. Comparison of velocity *u*.

$$w''(x)=\alpha\left(-\left(\frac{4}{h^2}+\frac{1}{\gamma^2}\right)+\left(\frac{12}{h^4}+\frac{10}{\gamma^2h^2}+\frac{1}{\gamma^4}\right)x^2-\left(\frac{9}{\gamma^2h^4}+\frac{2}{\gamma^4h^2}\right)x^4+\frac{1}{\gamma^4h^4}x^6\right)\exp\left(-\frac{x^2}{2\gamma^2}\right). \tag{92b}$$

Numerical examples are shown in **Figure 20**. When *h* is small, the accuracy becomes low. As has already pointed out in Section 5.1.1, there exist large errors at the boundaries. The numerical results agree well with the exact ones on overall. In the examples, γ is equal to dx = L/N = 0.1 and the accuracy seems sufficient when *h* is bigger than or equal to 4dx.

8. Conclusions

The author has once shown how to obtain the partial derivative in an irregular mesh using the moving least

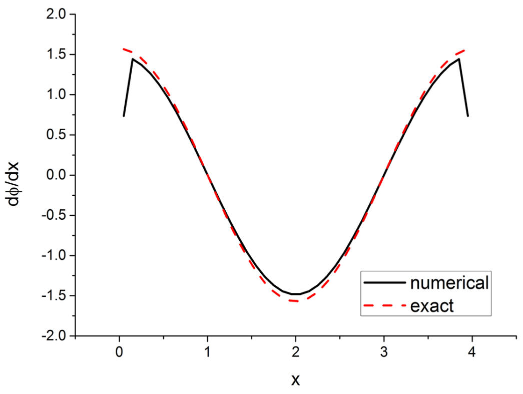
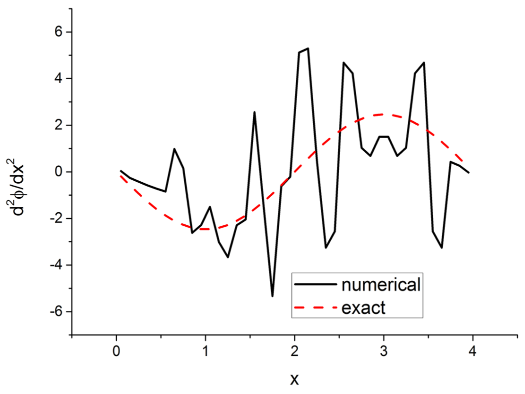
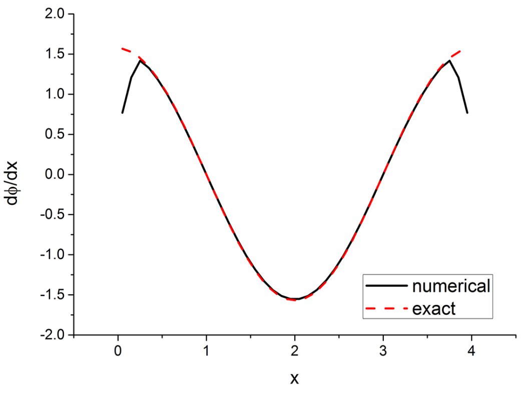
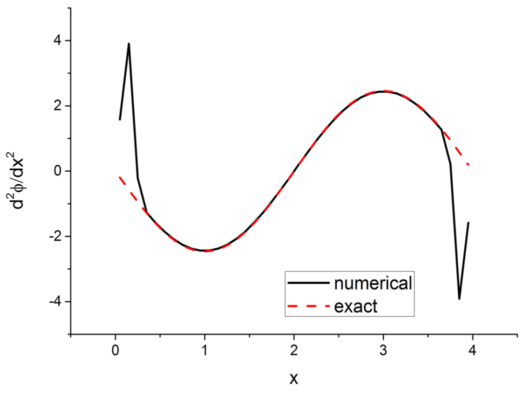
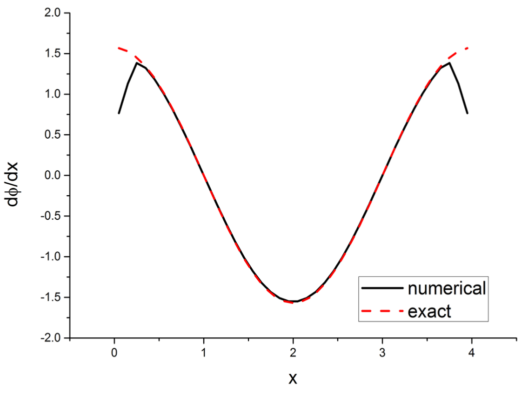
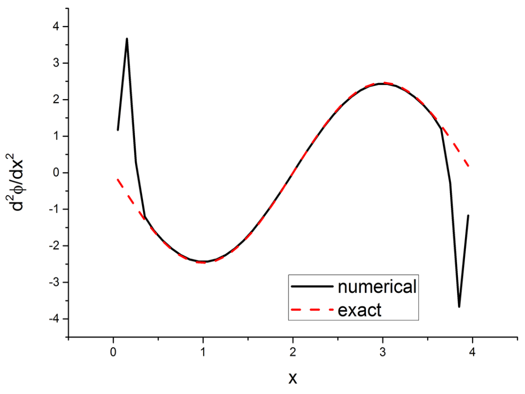
h	$\mathrm{d}\phi/\mathrm{d}x$	$\mathrm{d}^2\phi/\mathrm{d}x^2$
0.2		
0.4		
0.8		

Figure 20. Effect of support $2h$ ($\phi = \sin(2\pi x/L)$; $L = 4$, $N = 40$, $\gamma = 0.1$, $h = 0.2$, 0.4, 0.8).

square method [1]. A similar problem is discussed from a different viewpoint in SPH. Gingold & Monaghan [2] and Lucy [3] have developed Smooth Particle Hydrodynamics method (SPH). We have extended SPH theoretically in the present paper.

In Vortex Blob Method (VBM), the abduction is approximated reasonably. However, the diffusion of vortices can't be approximated precisely. We have shown in the present paper that this problem can be solved, if we apply the ideas developed by SPH.

Moving Particle Semi-implicit method (MPS) uses a similar dscretization of the initial and boundary value problem as SPH and VBM. However, the mathematical background of MPS is not sufficient. We have shown in the present paper that the mathematical background of the discrete gradient operator is strengthened by applying the ideas developed by SPH. However, that of the discrete Laplacian operator could not given.

The discrete gradient and Laplacian operators of SPH include the first, first and second derivatives of the weight function with respect to the space coordinates, respectively. On the other hand, those of MPS include only the weight function itself. This may be the biggest difference between the discrete differential operators in SPH and MPH.

In the present paper, solutions by FDM (Finite Difference Method), SPH and MPS are compared numerically in detail.

1) Effects of mesh on the discrete gradient and Laplacian operators were studied. MPS showed good results with respect to the random mesh.

2) The FDM, SPH and MPS were applied to the initial value problems, and the effects of the difference of the solution method were studied. In some cases, the solutions of initial value problem showed a difference between SPH and MPS.

3) The discrete Laplacian operator of SPH is sensitive to the spacial discontinuities of the solution function. Hence, in the case of the initial value problem, the discontinuity in the initial condition should be smoothened beforehand, and the small amount of the artificial viscosity should be introduced.

4) The author has once studied a mathematical background of MPS theoretically [1] and did some numerical calculations. In very limited cases, the discrete Laplacian operator of MPS can be obtained mathematically. However, the generalization to the general mesh was not obtained. Hence, we are obliged to apply the Laplacian operator as a bold approximation in case of the general mesh.

5) Recently, Ng, Hwang and Sheu [6] discussed the accuracy of the discrete Laplacian operator of MPS theoretically and numerically. They clarified an important aspect of the properties of the operator. As one of the properties, they pointed out in 2 in conclusion of Ref. [6] that MPS gave generally a favorable result in case of the irregular mesh. We also had a similar impression as expressed in (1) above.

The support of Gaussian weight in SPH is infinite. In the present paper, weights of a Gaussian-type of finite support with C1 continuity were also given.

Acknowledgements

The author thanks sincerely Prof. G. Yagawa, Toyo Univ., Japan for his support and encouragement.

References

[1] Isshiki, H. (2011) Discrete Differential Operators on Irregular Nodes (DDIN). *International Journal for Numerical Methods in Engineering*, **88**, 1323-1343. http://dx.doi.org/10.1002/nme.3225

[2] Gingold, R.A. and Monaghan, J.J. (1977) Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars. *Monthly Notices of the Royal Astronomical Society*, **181**, 375-389.
http://articles.adsabs.harvard.edu/cgi-bin/nph-iarticle_query?1977MNRAS.181..375G&amp;data_type=PDF_HIGH&amp;whole_paper=YES&amp;type=PRINTER&amp;filetype=.pdf
http://dx.doi.org/10.1093/mnras/181.3.375

[3] Lucy, L.B. (1977) A Numerical Approach to the Testing of the Fission Hypothesis. *The Astronomical Journal*, **82**, 1013-1024.
http://articles.adsabs.harvard.edu/cgi-bin/nph-iarticle_query?1977AJ.....82.1013L&defaultprint=YES&filetype=.pdf
http://dx.doi.org/10.1086/112164

[4] Chorin, A.J. (1973) Numerical Study of Slightly Viscous Flow. *Journal of Fluid Mechanics*, **57**, 785-796.
http://dx.doi.org/10.1017/S0022112073002016

[5] Yokoyama, M., Kubota, Y., Kikuchi, K., Yagawa, G. and Mochizuki, O. (2014) Some Remarks on Surface Conditions of Solid Body Plunging into Water with Particle Method. *Advanced Modeling and Simulation in Engineering Sciences*, **1**, 2-14. http://www.amses-journal.com/content/pdf/2213-7467-1-9.pdf

[6] Ng, K.C., Hwang, Y.H. and Sheu, T.W.H. (2014) On the Accuracy Assessment of Laplacian Models in MPS. *Computer Physics Communications*, **185**, 2412-2426.

[7] Isshiki, H., Nagata, S. and Imai, Y. (2014) Solution of Viscous Flow around a Circular Cylinder by a New Integral Representation Method (NIRM). *The Association for Japan Exchange and Teaching*, **2**, 60-82.
file:///C:/Users/l/Downloads/983-5001-1-PB%20(2).pdf

[8] Koshizuka, S. and Oka, Y. (1996) Moving Particle Semi-Implicit Method for Fragmentation of Incompressible Fluid. *Nuclear Science and Engineering*, **123**, 421-434.

[9] Moscardini, L. and Dolag, K. (2011) Chapter 4: Cosmology with Numerical Simulations. In: Matarrese, S., Colpi, M.,

Gorini, V. and Moschella, U., Eds., *Dark Matter and Dark Energy*, Springer, Berlin, 217-237.

[10] Monaghan, J.J. (1992) Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics*, **30**, 543-573. http://dx.doi.org/10.1146/annurev.aa.30.090192.002551

[11] Wikipedia, Burgers' Equation. http://en.wikipedia.org/wiki/Burgers'_equation

American Journal of Computational Mathematics, 2014, 4, 446-454
Published Online December 2014 in SciRes. http://www.scirp.org/journal/ajcm
http://dx.doi.org/10.4236/ajcm.2014.45037

Accuracy and Computational Cost of Interpolation Schemes While Performing *N*-Body Simulations

Shafiq Ur Rehman[1,2]

[1]Department of Mathematics, The University of Auckland, Auckland, New Zealand
[2]Department of Mathematics, University of Engineering and Technology, Lahore, Pakistan
Email: mailto:srehman@uet.edu.pk, srehman@uet.edu.pk

Received 31 October 2014; revised 28 November 2014; accepted 15 December 2014

Abstract

The continuous approximations play a vital role in *N*-body simulations. We constructed three different types, namely, one-step (cubic and quintic Hermite), two-step, and three-step Hermite interpolation schemes. The continuous approximations obtained by Hermite interpolation schemes and interpolants for ODEX2 and ERKN integrators are discussed in this paper. The primary focus of this paper is to measure the accuracy and computational cost of different types of interpolation schemes for a variety of gravitational problems. The gravitational problems consist of Kepler's two-body problem and the more realistic problem involving the Sun and four gas-giants—Jupiter, Saturn, Uranus, and Neptune. The numerical experiments are performed for the different integrators together with one-step, two-step, and three-step Hermite interpolation schemes, as well as the interpolants.

Keywords

***N*-Body Simulation, Integrators, Interpolation Schemes**

1. Numerical Integrators and Interpolants

Explicit Runge-Kutta-Nyström methods (ERKN) were introduced by E. J. Nyström in 1925 [1]. Here, we are using two variable-step-size ERKN integrators: Integrator ERKN689 is a nine stage, 6-8 FSAL pair [2] and ERKN101217 is a seventeen stage, 10-12 non-FSAL pair [2]. Dormand and Prince [3] and then Baker *et al.* [4] developed continuous approximation with embedded Runge-Kutta-Nyström methods, in which a third RKN process of order p^* was used to approximate the solutions, $y\left(t_{n-1+\alpha}\right)$ and $y'\left(t_{n-1+\alpha}\right)$, where $t_{n-1+\alpha} = t_{n-1} + \alpha h_{n-1}$

with α typically in (0, 1]. For ERKN101217, we used three existing interpolants: a 23-stage interpolant with $p^* = 10$, a 26-stage interpolant with $p^* = 11$, and a 29-stage interpolant with $p^* = 12$. The coefficients for these interpolants are not tabulated in this paper but are freely available on-line [4]. For ERKN689, we used an 8th-order interpolant with 12 stages. The coefficients for the continuous approximation of ERKN689 were provided by P. W. Sharp (private communication).

For the direct numerical approximation of systems of second-order ODEs, Hairer, Nørsett and Wanner [5] developed an extrapolation code ODEX2 based on the explicit midpoint rule with order selection and step-size control. The ODEX2 integrator is good for all tolerances, especially for high arithmetic precision, for example, 10^{-20} or 10^{-30}. For ODEX2 integrator we used the built-in interpolant.

Störmer methods are an important class of numerical methods for solving systems of second-order ordinary differential equations. Störmer methods were introduced by Störmer [6]. These methods have long been utilized for accurate long-term simulations of the solar system [7]. Grazier [8] recommended a fixed-step-size Störmer method of order 13 that used backward differences in summed form, summing from the highest to lowest differences. In this paper we consider an order-13, fixed-step-size Störmer method and refer to it as the $\bar{S}$-13 integrator.

1.1. Hermite Interpolation Schemes

Hermite interpolation uses derivative and function values and is named after Charles Hermite (1822-1901). We used four schemes: one-step (cubic and quintic Hermite), two-step and three-step Hermite interpolation schemes. The cubic Hermite interpolation polynomial is of degree 3, while the quintic, two-step and three-step Hermite interpolation polynomials are of degrees 5, 8 and 11, respectively. The interpolation schemes are derived using a Newton divided difference approach, which is described in Section 1.1.1. There is a second approach, which we call the direct approach that is frequently used by other researchers; for example, see [9]. This approach is particularly suited for cubic and quintic Hermite interpolation schemes, and we describe it in Sections 1.1.2 and 1.1.3.

1.1.1. Newton Divided Difference Approach

To determine the interpolating polynomial $P_m(t)$ for the m points (t_j, y_j), $j = 0, 1, \cdots, m-1$, using the Newton divided difference (NDD) approach, we write $P_m(t)$ as

$$P_m(t) = a_0 + a_1(t - t_0) + a_2(t - t_0)(t - t_1) + \cdots + a_m(t - t_0)\cdots(t - t_{m-1}),$$

where the a's are calculated from the divided differences. The i^{th} divided difference can be calculated using

$$f[t_0, t_1, \cdots, t_i] = \frac{f[t_1, \cdots, t_i] - f[t_0, \cdots, t_{i-1}]}{(t_i - t_0)},$$

see **Table 1**. We now discuss how the NDD must be modified when derivative values are used. Let us consider the first-order differences in **Table 1**. For example, if $t_1 = t_0$ then we have

$$\lim_{t_1 \to t_0} f[t_0, t_1] = f'(t_0).$$

Similarly, for the second-order differences in **Table 1**, if, for example, $t_2 = t_0$ then we find

$$\lim_{t_2 \to t_0} f[t_0, t_1, t_2] = \frac{f''(t_0)}{2}.$$

Hence, for Hermite interpolation schemes we can use the NDD approach if the derivatives replace the corresponding divided differences.

1.1.2. Cubic Hermite Interpolation

In this section and the next, we describe the direct approach for obtaining the cubic and quintic Hermite polynomials. The cubic Hermite interpolation polynomial $P_3(t)$ for the time-step from $t = t_{n-1}$ to $t = t_n$ interpolates the data (t_{n-i}, y_{n-i}) and (t_{n-i}, y'_{n-i}) at time t_{n-i}, for $i = 1$ and 0. In the direct approach, the cubic Hermite interpolation polynomial is written as

Table 1. An illustrative Newton divided difference table.

j	t_j	$f(t_j)$					
0	t_0	y_0					
			$f[t_0,t_1]$				
1	t_1	y_1		$f[t_0,t_1,t_2]$			
			$f[t_1,t_2]$		$f[t_0,t_1,t_2,t_3]$		
2	t_2	y_2		$f[t_1,t_2,t_3]$		$f[t_0,t_1,t_2,t_3,t_4]$	
			$f[t_2,t_3]$		$f[t_1,t_2,t_3,t_4]$		$f[t_0,t_1,t_2,t_3,t_4,t_5]$
3	t_3	y_3		$f[t_2,t_3,t_4]$		$f[t_1,t_2,t_3,t_4,t_5]$	
			$f[t_3,t_4]$		$f[t_2,t_3,t_4,t_5]$		
4	t_4	y_4		$f[t_3,t_4,t_5]$			
			$f[t_4,t_5]$				
5	t_5	y_5					

$$P_3(t) = a_0 y_{n-1} + a_1 H y'_{n-1} + a_2 y_n + a_3 H y'_n,$$

where,

$$a_0 = (\tau - 1)^2 (2\tau + 1),$$

$$a_1 = (\tau - 1)^2 \tau,$$

$$a_2 = \tau^2 (3 - 2\tau),$$

$$a_3 = \tau^2 (\tau - 1),$$

and $H = t_n - t_{n-1}$ with $\tau = (t - t_{n-1})/H$. Since the values of y and y' at both ends of each step are interpolated, the piecewise defined approximation $y_{\text{num}}(t)$ formed from the cubic Hermite polynomial is continuous and has a continuous first derivative.

1.1.3. Quintic Hermite Interpolation

The quintic Hermite interpolation polynomial $P_5(t)$ for the time-step from $t = t_{n-1}$ to $t = t_n$ interpolates the data (t_{n-i}, y_{n-i}), (t_{n-i}, y'_{n-i}), and (t_{n-i}, y''_{n-i}) at time t_{n-i}, for $i = 1$ and 0. As for cubic Hermite interpolation, the quintic Hermite interpolation polynomial can be derived using a direct approach and written as

$$P_5(t) = a_0 y_{n-1} + a_1 H y'_{n-1} + a_2 H^2 y''_{n-1} + a_3 y_n + a_4 H y'_n + a_5 H^2 y''_n,$$

where,

$$a_0 = (1 - \tau)^3 (6\tau^2 + 3\tau + 1),$$

$$a_1 = (1 - \tau)^3 \tau (3\tau + 1),$$

$$a_2 = (1 - \tau)^3 \tau^2 / 2,$$

$$a_3 = \tau^3 (6\tau^2 - 15\tau + 10),$$

$$a_4 = \tau^3 (1 - \tau)(3\tau - 4),$$

$$a_5 = \tau^3 (\tau - 1)^2 / 2,$$

and $H = t_n - t_{n-1}$ with $\tau = (t - t_{n-1})/H$, as before. Since the values of y, y', and y'' at both ends of each step are interpolated, the piecewise defined approximation $y_{\text{num}}(t)$ formed from the quintic Hermite polynomial is continuous and has continuous first and second derivatives.

1.1.4. Two-Step Hermite Interpolation Polynomial

The two-step Hermite interpolation polynomial $P_8(t)$ for the two-time-steps from $t = t_{n-2}$ to $t = t_n$ interpolates the data (t_{n-i}, y_{n-i}), (t_{n-i}, y'_{n-i}), and (t_{n-i}, y''_{n-i}) at time t_{n-i}, with $i = 2,1,$ and 0. The two-step Hermite interpolation polynomial $P_8(t)$ can then be written in Horner's nested multiplication form as

$$P_8(t) = D_{11} + (t - t_{n-2})\Big(D_{22} + (t - t_{n-2})\Big(D_{33} + (t - t_{n-2})\Big(D_{44} + (t - t_{n-1})\Big(D_{55} + (t - t_{n-1})$$
$$\times\Big(D_{66} + (t - t_{n-1})\Big(D_{77} + (t - t_n)\Big(D_{88} + (t - t_n)D_{99}\Big)\Big)\Big)\Big)\Big)\Big)\Big),$$

where the coefficients D_{ii}, $i = 1,\cdots,9$, are obtained using a NDD table. Since the values of y, y', and y'' at both ends of each step are interpolated, the piecewise defined approximation $y_{\text{num}}(t)$ formed from the two-step Hermite polynomial is continuous and has continuous first and second derivatives.

1.1.5. Three-Step Hermite Interpolation Polynomial

Similarly, the three-step Hermite interpolation polynomial interpolates the data (t_{n-i}, y_{n-i}), (t_{n-i}, y'_{n-i}), and (t_{n-i}, y''_{n-i}) at time t_{n-i}, for $i = 3,2,1,$ and 0. Hence, it is the degree-11 polynomial $P_{11}(t)$ defined over a three-time-step from $t = t_{n-3}$ to $t = t_n$. Using Horner's nested multiplication form, we can write $P_{11}(t)$ as

$$P_{11} = D_{11} + (t - t_{n-3})\Big(D_{22} + (t - t_{n-3})\Big(D_{33} + (t - t_{n-3})\Big(D_{44} + (t - t_{n-2})\Big(D_{55} + (t - t_{n-2})$$
$$\times\Big(D_{66} + (t - t_{n-2})\Big(D_{77} + (t - t_{n-1})\Big(D_{88} + (t - t_{n-1})\Big(D_{99} + (t - t_{n-1})\Big(D_{1010} + (t - t_n)$$
$$\times\Big(D_{1111} + (t - t_n)D_{1212}\Big)\Big)\Big)\Big)\Big)\Big)\Big)\Big)\Big)\Big)\Big).$$

As for the two-step Hermite interpolation polynomial, the coefficients D_{ii}, $i = 1,\cdots,12$, of $P_{11}(t)$ are obtained using NDD. Since the values of y, y', and y'' at both ends of each step are interpolated, the piecewise defined approximation $y_{\text{num}}(t)$ formed from the three-step Hermite polynomial is continuous and has continuous first and second derivatives.

We compared the maximum error in position and the CPU-time for P_3 and P_5 evaluated using NDD and the direct approach. The comparison was done for one period of Kepler problem for eccentricities of 0.05 to 0.9 (see Section 2.1 for more details on the experiment), and the Jovian problem [10].

For the two-body problem, no significant differences in the maximum error as CPU-time were observed between these two approaches. For the Jovian problem, the direct approach takes approximately half the CPU-time of the NDD approach. The coefficients of the polynomial for the NDD approach depend on the components of the solution vector. For the direct approach the coefficients are independent of the components, so they can be used as a vector to approximate polynomials and that will save CPU-time.

In the rest of the paper, cubic and quintic Hermite interpolation schemes are implemented using the direct approach. For two-step and three-step Hermite interpolation schemes we implemented the NDD approach, because it is really difficult to find the coefficients for the direct approach.

2. Numerical Experiments

Here, we examine the error growth in the position and velocity for the Kepler problem. The experiments for short-term integrations are performed using four different types of interpolation schemes applied to the Kepler problem over the interval of 2π.

2.1. Kepler Problem with Different Eccentricities

The solution to the Kepler problem is periodic with period 2π. We do not have to calculate the reference solution, so the Kepler problem is well suited for testing the accuracy of integration over a short time interval. This assumes the step-size is chosen so that $t = 2\pi$ is hit exactly.

The error in the position and velocity of the Kepler problem is given by the L_2-norm

$$E_r(t) = \left\| r_{\text{num}}(t) - r_{\text{true}}(t) \right\|_2,$$
$$E_v(t) = \left\| v_{\text{num}}(t) - v_{\text{true}}(t) \right\|_2,$$

where $r_{\text{num}}(t)$ and $r_{\text{true}}(t)$ are the vectors of the numerical and true solutions, and $v_{\text{num}}(t)$ and $v_{\text{true}}(t)$ are the vectors of the derivatives to the numerical and true solutions, respectively.

The graphs in **Figure 1** are for experiments performed with the cubic, quintic, two-step, and three-step Hermite interpolation schemes applied to the Kepler problem over the interval $[0, 2\pi]$ for eccentricities in the range [0.05, 0.9]; note that, in reality, planets and test particles do not have eccentricity 0, and we used 0.05 as an approximate upper bound for the eccentricities of the Jovian planets. The selection of these interpolation schemes is motivated by the fact that they can be used with all the integrators described in this paper. The interpolants, on the other hand, are related to specific integrators; for example, the 12-stage interpolant can only be used with the ERKN689 integrator. For the experiments shown in **Figure 1**, the interval of integration is subdivided into 30 evenly spaced sub-intervals; experiments with different numbers of sub-intervals are recorded in **Table 2**. We then evaluate the position and velocity at 10 evenly spaced points on each sub-interval using different interpolation schemes. Note that we also tested with up to 100 sample points and observed a variation in the error of not more than 1%. The information, such as, positions, velocities, and times, are saved in separate files. In a post-processing step, we then calculate the errors in the positions and velocities with respect to the analytical solution that we obtain at the stored values of time. The velocity polynomials for all these interpolation schemes are obtained by differentiating their corresponding position polynomials.

From **Figure 1**, we observe a clear pattern; as the eccentricity increases, the maximum error in the position also increases. The variation in the error is understandable, because the error depends upon the eccentricity. To illustrate this fact, recall that the analytical solution to the Kepler problem is given by

$$\left(y_1(t), y_2(t)\right) = \left[\cos(\eta) - e, \sqrt{1-e^2}\,\sin(\eta)\right]^{\mathrm{T}}, \tag{1}$$

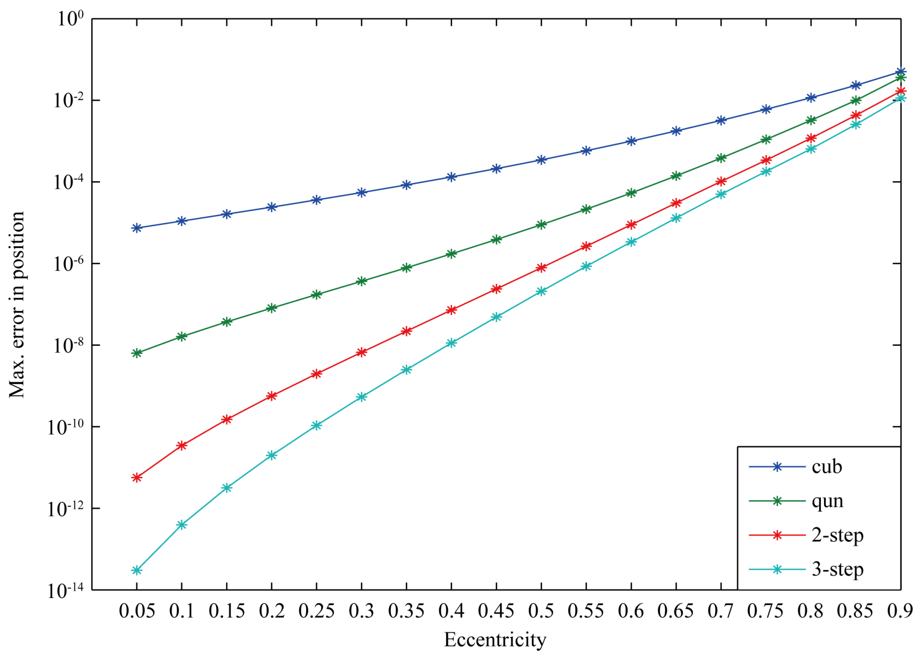


Figure 1. The maximum global error in position for the cubic, quintic, 2-step, and 3-step Hermite interpolation schemes against different eccentricities applied to the two-body problem over a period of 2π.

Table 2. The maximum global error in position for eccentricity 0.05 attained by different interpolation schemes applied to the Kepler problem over the interval [0, 2π] with four choices of numbers of evenly spaced sub-intervals. The dash means the combination is not used.

N_{sub}	Cubic	Quintic	2-step	3-step
17	0.71×10^{-06}	0.19×10^{-08}	0.87×10^{-11}	0.21×10^{-12}
79	0.15×10^{-08}	0.19×10^{-12}	0.56×10^{-16}	0.55×10^{-16}
255	0.14×10^{-10}	0.18×10^{-15}	0.55×10^{-16}	-
1080	0.44×10^{-13}	0.56×10^{-16}	-	-

where η is the eccentric anomaly satisfying $t=\eta-e\sin(\eta)$ and the interpolation error, for example, for the y_1-component of this solution can be written as $\alpha h^{p+1}\frac{y_1^{(p+1)}(\xi)}{(p+1)!}$. The first three derivatives of y_1 are

$$\frac{\mathrm{d}y_1}{\mathrm{d}t}=\frac{-\sin(\eta)}{1-e\cos(\eta)},$$

$$\frac{\mathrm{d}^2y_1}{\mathrm{d}t^2}=\frac{e-\cos(\eta)}{(1-e\cos(\eta))^3},$$

$$\frac{\mathrm{d}^3y_1}{\mathrm{d}t^3}=\frac{\sin(\eta)\left(1+2e\cos(\eta)-3e^2\right)}{(1-e\cos(\eta))^5}.$$

It is clear that these and all subsequent derivatives are expected to involve the factor $1/(1-e\cos(\eta))$. Since the minimum value of $|1-e\cos(\eta)|$ gets smaller and smaller as e increases, it is expected that the error increases as e increases; a similar argument holds for the y_2-component. Indeed, for all interpolation schemes the minimum error in the position occurs at eccentricity 0.05 and the maximum error at eccentricity 0.9 in **Figure 1**. We also observe in **Figure 1** that, for small eccentricity like e = 0.05, the difference in the errors between consecutive interpolation schemes is approximately two orders of magnitude. As e increases to 1, this difference decreases and all four errors in **Figure 1** appear to converge. We also computed the error in the velocity and found that it is nearly two orders of magnitude larger than the error in the position. These experiments were also performed in quadruple-precision, but there was hardly any difference between the estimated errors obtained in double- and quadruple-precision. For example, using a 3-step interpolation scheme with eccentricity 0.05 and 0.9, the differences between the estimated errors in the position obtained in double- and quadruple-precision are 4.40×10^{-15} and 1.67×10^{-14}, respectively. We conclude that the interpolation schemes are not affected a great deal by the round-off error when using 30 evenly spaced sub-intervals.

As mentioned earlier, the same sets of experiments described in **Figure 1**, were also done with different numbers of sub-intervals. We experimented with 17, 79, 255, and 1080 evenly spaced sub-intervals over the interval $[0,2\pi]$. The associated errors for e = 0.05 are shown in **Table 2**. This particular selection of the number of sub-intervals is due to the fact that we wish to maintain the best observed accuracy of the integrators ODEX2, ERKN101217, ERKN689, and $\bar{S}$-13; see [10] and note that we use a time-step of 4 days for $\bar{S}$-13. For example, the ODEX2 integrator applied to the Jovian problem achieves best accuracy using tolerance 10^{-16} and an average time-step of approximately 260 days over one million years. Since Jupiter's orbital period is approximately 4320 Earth days, a time-step of 260 days gives approximately 17 steps.

The results in **Table 2** show reasonably good agreement with the expected values calculated from the orders of the polynomial, discounting the possible increase in round-off error from using a higher-order interpolation scheme and a large number of sub-intervals. For example, using the cubic Hermite interpolation scheme, and going from 17 to 79 sub-intervals, the expected value is $(17/79)^4\times\left(0.71\times10^{-06}\right)\approx1.52\times10^{-09}$ which has very good agreement with the value 0.15×10^{-08} mentioned in **Table 2**.

From **Table 2** we find that the accuracy for a given interpolation scheme improves if the number of sub-intervals increases. We also deduce from **Table 2** that it makes no sense to use any of the four interpolation schemes with ODEX2 if the required maximum global error is 10^{-15} and 17 sub-intervals are used. For 79 sub-intervals (used for ERKN101217) only the 2-step and 3-step Hermite interpolation schemes achieve the required accuracy. Similarly, for ERKN689, the quintic and 2-step interpolation schemes achieve the required accuracy, whereas for the $\bar{S}$-13 integrator, only the quintic Hermite interpolation scheme does.

2.2. Computational Cost of Interpolation Schemes

Let us now consider the CPU-time by looking at individual interpolation schemes. Our expectation, at least for the interpolation schemes, is that the CPU-time is proportional to the number of multiplications. If one interpolation scheme uses twice as many multiplications then the CPU-time is expected to be twice as large. There will not be many divisions, and the number of subtractions and additions is typically proportional to the number of multiplications. Normally, when timing a program, an overhead is introduced. Therefore, care has been taken not to include such overheads in the final results. We also checked reproducibility of the results and observed a maximum variation of not more than 2.5%.

As discussed earlier, there are two different approaches to form interpolation schemes. Here, the experiments are performed using a direct approach for cubic and quintic Hermite interpolation, and the Newton divided difference approach for 2-step and 3-step Hermite interpolation schemes. In most cases, interpolation schemes are split into two subroutines, one for finding the coefficients and one for evaluating the polynomials. For ERKN689 and ERKN101217, with the interpolants we have additional stage derivatives (function evaluations). Overall, we have three different groups of interpolation schemes:

1) For cubic and quintic Hermite interpolation schemes, we evaluate the coefficients of the polynomial, which are independent of the components, and the polynomial as one subroutine.

2) For 2-step and 3-step Hermite interpolation schemes, we have two subroutines:

a) The calculation of the coefficients by forming a Newton divided difference table;

b) The evaluation of the polynomial.

3) For the interpolants, we have three subroutines:

a) The evaluation of the coefficients $b^{\star}$;

b) The evaluation of the additional stage derivatives;

c) The evaluation of the polynomial using a) and b).

For ODEX2, the pieces of information required to form the interpolant are considered part of the integration, and we only consider the evaluation of the polynomial; see **Table 5**.

Since the coefficients of the polynomials for cubic and quintic interpolations are independent of the components, the experiments for these interpolation schemes are performed as one unit. As can be seen from the formulae in Sections 1.1.2 and 1.1.3, the quintic Hermite interpolation scheme uses approximately 93% more multiplications than cubic Hermite interpolation when applied to the Jovian problem. When we did our experiment, we found that the quintic Hermite interpolation scheme uses approximately 96% more CPU-time than cubic Hermite interpolation, which is in good agreement with the expected value.

Table 3 shows the CPU-time for finding the stage derivatives of the pairs (without the cost of additional stage derivatives) used in ERKN689 and ERKN101217 when applied to the Jovian problem. With ERKN689 we use the property FSAL (first same as last), so that we need only 8 derivative evaluations per step. Similarly, the 12-stage interpolant has effectively 11 stage derivatives. Observe from **Table 3** that the average CPU-time consumed per stage is approximately 8.74×10^{-07} and 8.71×10^{-07} for ERKN689 and ERKN101217, respectively. Therefore, the expected CPU-time for ERKN689 with a 12-stage interpolant is approximately 9.61×10^{-06}. For ERKN101217, the expected CPU-time for finding coefficients is approximately 2.00×10^{-05}, 2.26×10^{-05}, and 2.52×10^{-05} with 23-stage, 26-stage, and 29-stage interpolants, respectively.

Table 4 gives the CPU time needed to find the coefficients of the interpolation schemes and evaluate all the derivatives for the interpolants when solving the Jovian problem. We observe that all values in **Table 4** have reasonably good agreement with the prescribed values for CPU-time. Note also that the 3-step Hermite interpolation scheme in **Table 4** uses approximately 96% more multiplications than the 2-step Hermite interpolation scheme, which is reasonably well matched by our finding of 93%.

Table 5 shows the CPU-time for evaluating the position and velocity components using the different interpolation polynomials. The 3-step interpolation scheme uses approximately 76% more CPU-time than the 2-step interpolation, which is again in agreement with the CPU-times observed in **Table 4**. Similarly, the difference in

Table 3. The CPU-time in seconds for evaluating the stage derivatives (without the cost of additional function evaluations) for ERKN689 and ERKN101217 applied to the Jovian problem.

Integrator	ERKN689	ERKN101217
	9-stage	17-stage
CPU-time	6.99×10^{-06}	1.48×10^{-05}

Table 4. The CPU-time in seconds for finding the coefficients of the interpolation schemes and evaluating all the stage derivatives of the interpolants applied to the Jovian problem.

Interpolation			ERKN689		ERKN101217	
Polynomial	2-step	3-step	12-stage	23-stage	26-stage	29-stage
CPU-time	2.83×10^{-6}	5.54×10^{-6}	9.54×10^{-6}	2.00×10^{-5}	2.32×10^{-5}	2.66×10^{-5}

Table 5. The CPU-time in seconds for evaluating the position and velocity polynomials using different interpolation polynomials applied to the Jovian problem.

Interpolation			ERKN68		ERKN101217		ODEX2
Polynomial	2-step	3-step	12-stage	23-stage	26-stage	29-stage	Interpolant
CPU-time	6.7×10^{-7}	1.18×10^{-6}	2.97×10^{-7}	4.42×10^{-7}	4.96×10^{-7}	5.50×10^{-7}	1.48×10^{-6}

CPU-time between the 23-stage and 29-stage interpolants is twice the difference between the 23-stage and 26-stage interpolants which is in good agreement with the difference observed in **Table 4**.

3. Summary

The primary objective of this paper was to discuss the accuracy and computational cost of different interpolation schemes while performing N-body simulations. The interpolation schemes play a vital role in these kinds of simulations. We constructed three different types, namely, one-step (cubic and quintic Hermite), two-step and three-step Hermite interpolation schemes. For short-term simulations, we investigated the performance of these interpolation schemes applied to the Kepler problem over the interval [0, 2] for eccentricities in the range [0.05, 0.9]. We observed that the maximum error in position was monotonically increasing as a function of eccentricity. For a given number of sub-intervals we used in this paper, the higher-order interpolation schemes achieve better accuracy and for a given interpolation scheme the accuracy improves if the number of sub-intervals is increased. We also investigated the CPU-time by looking at individual interpolation schemes. Our expectation, at least for the interpolation schemes, was that the CPU-time was proportional to the number of multiplications. For example, the quintic Hermite interpolation scheme uses approximately 93% more multiplications than cubic Hermite interpolation when applied to the Jovian problem. When we did our experiment, we found that the quintic Hermite interpolation scheme used approximately 96% more CPU-time than cubic Hermite interpolation, which was in good agreement with the expected value. We also checked reproducibility of the results and observed a maximum variation of not more than 2.5%.

Acknowledgements

The author is grateful to the Higher Education Commission (HEC) of Pakistan for providing the funding to carry out this research. Special thanks go to Dr. P. W. Sharp and Prof. H. M. Osinga for their valuable suggestions, discussions, and guidance throughout this research.

References

[1] Nyström, E.J. (1925) Über die numerische Integration von Differentialgleichungen. *Acta Societatis Scientiarum Fennicae*, **50**, 1-54.

[2] Dormand, J., El-Mikkawy, M.E.A. and Prince, P. (1987) Higher Order Embedded Runge-Kutta-Nyström Formulae. *IMA Journal of Numerical Analysis*, **7**, 423-430. http://dx.doi.org/10.1093/imanum/7.4.423

[3] Dormand, J.R. and Prince, P.J. (1987) New Runge-Kutta Algorithms for Numerical Simulation in Dynamical Astronomy. *Celestial Mechanics*, **18**, 223-232. http://dx.doi.org/10.1007/BF01230162

[4] Baker, T.S., Dormand, J.R. and Prince, P.J. (1999) Continuous Approximation with Embedded Runge-Kutta-Nyström Methods. *Applied Numerical Mathematics*, **29**, 171-188. http://dx.doi.org/10.1016/S0168-9274(98)00065-8

[5] Hairer, E., Nørsett, S.P. and Wanner, G. (1987) Solving Ordinary Differential Equations I: Nonstiff Problems. Springer-Verlag, Berlin.

[6] Störmer, C. (1907) Sur les trajectoires des corpuscles électrisés. *Acta Societatis Scientiarum Fennicae*, **24**, 221-247.

[7] Grazier, K.R., Newman, W.I., Kaula, W.M. and Hyman, J.M. (1999) Dynamical Evolution of Planetesimals in Outer Solar System. *ICARUS*, **140**, 341-352. http://dx.doi.org/10.1006/icar.1999.6146

[8] Grazier, K.R. (1997) The Stability of Planetesimal Niches in the Outer Solar System: A Numerical Investigation. Ph.D. Thesis, University of California, Berkeley.

[9] Grazier, K.R., Newman, W.I. and Sharp, P.W. (2013) A Multirate Störmer Algorithm for Close Encounters. *The Astronomical Journal*, **145**, 112-119. http://dx.doi.org/10.1088/0004-6256/145/4/112

[10] Rehman, S. (2013) Jovian Problem: Performance of Some High-Order Numerical Integrators. *American Journal of Computational Mathematics*, **3**, 195-204. http://dx.doi.org/10.4236/ajcm.2013.33028

American Journal of Computational Mathematics, 2014, 4, 455-463
Published Online December 2014 in SciRes. http://www.scirp.org/journal/ajcm
http://dx.doi.org/10.4236/ajcm.2014.45038

Exact Traveling Wave Solutions for the System of Shallow Water Wave Equations and Modified Liouville Equation Using Extended Jacobian Elliptic Function Expansion Method

Emad H. M. Zahran[1], Mostafa M. A. Khater[2*]

[1]Department of Mathematical and Physical Engineering, College of Engineering, University of Benha, Shubra, Egypt
[2]Department of Mathematics, Faculty of Science, Mansoura University, Mansoura, Egypt
Email: *mostafa.Khater2024@yahoo.com

Received 11 September 2014; revised 21 October 2014; accepted 15 November 2014

Abstract

In this work, an extended Jacobian elliptic function expansion method is proposed for constructing the exact solutions of nonlinear evolution equations. The validity and reliability of the method are tested by its applications to the system of shallow water wave equations and modified Liouville equation which play an important role in mathematical physics.

Keywords

Extended Jacobian Elliptic Function Expansion Method, The System of Shallow Water Wave Equations, Modified Liouville Equation, Traveling Wave Solutions, Solitary Wave Solutions

1. Introduction

The nonlinear partial differential equations of mathematical physics are major subjects in physical science [1]. Exact solutions for these equations play an important role in many phenomena in physics such as fluid mechanics, hydrodynamics, optics, plasma physics and so on. Recently many new approaches for finding these solu-

*Corresponding author.

tions have been proposed, for example, tanh-sech method [2]-[4], extended tanh-method [5]-[7], sine-cosine method [8]-[10], homogeneous balance method [11] [12], F-expansion method [13]-[15], exp-function method [16], the modified simple equation method [17], the $\exp(-\phi(\xi))$-expansion method [18], $\left(\frac{G'}{G}\right)$-expansion method [19]-[22], Jacobi elliptic function method [23]-[26] and so on.

The objective of this article is to apply the extended Jacobian elliptic function expansion method for finding the exact traveling wave solution the system of shallow water wave equations and modified Liouville equation which play an important role in mathematical physics.

The rest of this paper is organized as follows: In Section 2, we give the description of the extended Jacobi elliptic function expansion method. In Section 3, we use this method to find the exact solutions of the nonlinear evolution equations pointed out above. In Section 4, conclusions are given.

2. Description of Method

Consider the following nonlinear evolution equation

$$F(u, u_t, u_x, u_{tt}, u_{xx}, \cdots) = 0, \tag{1}$$

where F is polynomial in $u(x,t)$ and its partial derivatives in which the highest order derivatives and nonlinear terms are involved. In the following, we give the main steps of this method [23]-[26].

Step 1. Using the transformation

$$u = u(\xi), \quad \xi = x - ct, \tag{2}$$

where c is wave speed, to reduce Equation (1) to the following ODE:

$$P(u, u', u'', u''', \cdots) = 0, \tag{3}$$

where P is a polynomial in $u(\xi)$ and its total derivatives, while $' = \frac{\mathrm{d}}{\mathrm{d}\xi}'$.

Step 2. Making good use of ten Jacobian elliptic functions, we assume that (3) has the solutions in these forms:

$$u(\xi) = a_0 + \sum_{j=1}^{N} f_i^{j-1}(\xi)\left[a_j f_i(\xi) + b_j g_i(\xi)\right], \quad i = 1, 2, 3, \cdots, \tag{4}$$

with

$$\begin{aligned}
&f_1(\xi) = sn\xi, \quad g_1(\xi) = cn\xi,\\
&f_2(\xi) = sn\xi, \quad g_2(\xi) = dn\xi,\\
&f_3(\xi) = ns\xi, \quad g_3(\xi) = cs\xi,\\
&f_4(\xi) = ns\xi, \quad g_4(\xi) = ds\xi,\\
&f_5(\xi) = sc\xi, \quad g_5(\xi) = nc\xi,\\
&f_6(\xi) = sd\xi, \quad g_6(\xi) = nd\xi,
\end{aligned} \tag{5}$$

where $sn\xi$, $cn\xi$, $dn\xi$, are the Jacobian elliptic sine function, the jacobian elliptic cosine function and the Jacobian elliptic function of the third kind and other Jacobian functions which is denoted by Glaisher's symbols and are generated by these three kinds of functions, namely

$$\begin{aligned}
&ns\xi = \frac{1}{sn\xi}, \quad nc\xi = \frac{1}{cn\xi}, \quad nd\xi = \frac{1}{dn\xi}, \quad sc\xi = \frac{cn\xi}{sn\xi},\\
&cs\xi = \frac{sn\xi}{cn\xi}, \quad ds\xi = \frac{dn\xi}{sn\xi}, \quad sd\xi = \frac{sn\xi}{dn\xi},
\end{aligned} \tag{6}$$

that have the relations

$$\begin{aligned}
&sn^2\xi + cn^2\xi = 1, \quad dn^2\xi + m^2 sn^2\xi = 1, \quad ns^2\xi = 1 + cs^2\xi,\\
&ns^2\xi = m^2 + ds^2\xi, \quad sc^2\xi + 1 = nc^2\xi, \quad m^2 sd^2 + 1 = nd^2\xi,
\end{aligned} \tag{7}$$

with the modulus m $(0 < m < 1)$. In addition we know that

$$\frac{d}{d\xi} sn\xi = cn\xi dn\xi, \quad \frac{d}{d\xi} cn\xi = -sn\xi dn\xi, \quad \frac{d}{d\xi} dn\xi = -m^2 sn\xi cn\xi. \tag{8}$$

The derivatives of other Jacobian elliptic functions are obtained by using Equation (8). To balance the highest order linear term with nonlinear term we define the degree of u as $D[u] = n$ which gives rise to the degrees of other expressions as

$$D\left[\frac{d^q u}{d\xi^q}\right] = n + q, \quad D\left[u^p \left(\frac{d^q u}{d\xi^q}\right)^s\right] = np + s(n+q). \tag{9}$$

According the rules, we can balance the highest order linear term and nonlinear term in Equation (3) so that n in Equation (4) can be determined.

Noticed that $sn\xi \to \tanh\xi$, $cn\xi \to \operatorname{sech}\xi$, $dn\xi \to \operatorname{sech}\xi$ when the modulus $m \to 1$ and $sn\xi \to \sin\xi$, $cn\xi \to \cos\xi$, $dn\xi \to 1$ when the modulus $m \to 0$, we can obtain the corresponding solitary wave solutions and triangle function solutions, respectively, while when therefore Equation (5) degenerate as the following forms

$$u(\xi) = a_0 + \sum_{j=1}^{N} \tanh^{j-1}(\xi)\left[a_j \tanh(\xi) + b_j \operatorname{sech}(\xi)\right], \tag{10}$$

$$u(\xi) = a_0 + \sum_{j=1}^{N} \coth^{j-1}(\xi)\left[a_j \coth(\xi) + b_j \coth(\xi)\right], \tag{11}$$

$$u(\xi) = a_0 + \sum_{j=1}^{N} \tan^{j-1}(\xi)\left[a_j \tan(\xi) + b_j \sec(\xi)\right], \tag{12}$$

$$u(\xi) = a_0 + \sum_{j=1}^{N} \cot^{j-1}(\xi)\left[a_j \cot(\xi) + b_j \csc(\xi)\right]. \tag{13}$$

Therefore the extended Jacobian elliptic function expansion method is more general than sine-cosine method, the tan-function method and Jacobian elliptic function expansion method.

3. Application

3.1. Example 1: The System of Shallow Water Wave Equations

We first consider the system of the shallow water wave equation [27] in order to demonstrate the $\exp(-\phi(\xi))$-expansion method

$$\begin{cases} u_t + (uv)_x + v_{xxx} = 0, \\ v_t + u_x + vv_x = 0. \end{cases} \tag{14}$$

We use the wave transformation $u(x,t) = u(\xi)$, $\xi = x - ct$ to reduce Equations (14) to the following nonlinear system of ordinary differential equations:

$$\begin{cases} -cu' + vu' + uv' + v''' = 0, \\ u' - cv' + vv' = 0, \end{cases} \tag{15}$$

where by integrating once the second equation with zero constant of integration, we find

$$u = cv - \frac{v^2}{2} \tag{16}$$

substituting Equation (16) into the first equation of Equation (15) we obtain

$$v''' + \left(3cv - \frac{3v^2}{2} - c^2\right)v' = 0. \tag{17}$$

Integrating Equation (17) with zero constant of integration, we find

$$v'' + \frac{3}{2}cv^2 - \frac{1}{2}v^3 - c^2 v = 0. \tag{18}$$

Balancing v'' and v^3 in Equation (18) yields, $N + 2 = 3N \Rightarrow N = 1$. This suggests the choice of $v(\xi)$ in Equation (18) as

$$u = a_0 + a_1 sn + b_1 cn, \tag{19}$$

where a_0, a_1 and b_1 are constant such that $a_1 \neq 0$ or $b_1 \neq 0$. From (19), it is easy to see that

$$u' = a_1 cndn - b_1 sndn, \tag{20}$$

$$u'' = -m^2 sna_1 + 2a_1 sn^3 m^2 + 2m^2 sn^2 cnb_1 - a_1 sn - b_1 cn. \tag{21}$$

Substituting Equations (19) and (21) into Equation (18) and equating all coefficients of sn^3, $sn^2 cn$, sn^2, $sncn$, sn, cn, sn^0 respectively to zero, we obtain:

$$2a_1 m^2 - \frac{1}{2}a_1^3 + \frac{3}{2}a_1 b_1^2 = 0, \tag{22}$$

$$2m^2 b_1 - \frac{3}{2}a_1^2 b_1 + \frac{1}{2}b_1^3 = 0, \tag{23}$$

$$\frac{3}{2}ca_1^2 - \frac{3}{2}cb_1^2 - \frac{3}{2}a_0 a_1^2 + \frac{3}{2}a_0 b_1^2 = 0, \tag{24}$$

$$3ca_1 b_1 - 3a_0 a_1 b_1 = 0, \tag{25}$$

$$-a_1 m^2 - a_1 + 3ca_0 a_1 - \frac{3}{2}a_0^2 a_1 - \frac{3}{2}a_1 b_1^2 - c^2 a_1 = 0, \tag{26}$$

$$-b_1 + 3ca_0 b_1 - \frac{3}{2}a_0^2 b_1 - \frac{1}{2}b_1^3 - c^2 b_1 = 0, \tag{27}$$

$$\frac{3}{2}c\left(a_0^2 + b_1^2\right) - \frac{1}{2}a_0^3 - \frac{3}{2}a_0 b_1^2 - c^2 a_0 = 0. \tag{28}$$

Solving the above system with the aid of Maple or Mathematica, we have the following solution:

Case 1.

$$c = a_0 = \pm\sqrt{2m^2 + 2},\ a_1 = \pm 2m,\ b_1 = 0.$$

So that the solution of Equation (18) can be written as

$$u = \pm\sqrt{2m^2 + 2} \pm 2msn, \tag{29}$$

when $m = 1$, the solution can be in the form

$$u = \pm 2 \pm 2\tanh(\xi). \tag{30}$$

Case 2.

$$c = a_0 = \pm\sqrt{2 - m^2},\ a_1 = \pm m,\ b_1 = \pm im.$$

So that the solution of Equation (18) can be written as

$$u = \pm\sqrt{2 - m^2} \pm msn \pm imcn, \tag{31}$$

when $m = 1$, the solution can be in the form

$$u = \pm 1 \pm \tanh(\xi) \pm i\operatorname{sech}(\xi). \tag{32}$$

Case 3.

$$c = a_0 = \pm\sqrt{2 - 4m^2},\ a_1 = 0,\ b_1 = \pm 2im.$$

So that the solution of Equation (18) can be written as

$$u = \pm\sqrt{2-4m^2} \pm 2imcn, \tag{33}$$

when $m=1$, the solution can be in the form

$$u = \pm\sqrt{-2} \pm 2i\operatorname{sech}(\xi). \tag{34}$$

3.2. Example 2: Modified Liouville Equation

Now, let us consider the modified Liouville equation [28].

$$a^2u_{xx} - u_{tt} + be^{\beta u} = 0, \tag{35}$$

respectively, where a, β and b are non zero and arbitrary coefficients. Using the wave transformation $u(x,t) = u(\xi)$, $\xi = kx + \omega t$, $v = e^{\beta u}$, to reduce Equation (35) to be in the form:

$$\left(\frac{k^2a^2}{\beta} - \frac{\omega^2}{\beta}\right)v''v - \left(\frac{k^2a^2}{\beta} - \frac{\omega^2}{\beta}\right)v'^2 + bv^3 = 0. \tag{36}$$

Balancing $v''v$ and v^3 in Equation (36) yields, $N+2+N = 3N \Rightarrow N = 2$. Consequently, we have the formal solution:

$$u(\xi) = a_0 + a_1sn + b_1cn + a_2sn^2 + b_2sncn, \tag{37}$$

where a_0, a_1, a_2 are constants to be determined, such that $a_2 \neq 0$ or $b_2 \neq 0$. It is easy to see that

$$u' = a_1cndn - b_1sndn + 2dna_2sncn - 2dnb_2sn^2 + dnb_2, \tag{38}$$

$$\begin{aligned} u'' = {} & -m^2sna_1 + 2a_1sn^3m^2 + 2m^2sn^2cnb_1 - 4a_2m^2sn^2 + 6a_2sn^4m^2 + 6m^2sn^3cnb_2 \\ & -m^2sncnb_2 - a_1sn - b_1cn + 2a_2 - 4a_2sn^2 - 4b_2sncn. \end{aligned} \tag{39}$$

Substituting (37) and (39) into Equation (36) and equating all the coefficients of sn^6, sn^5cn, sn^5, sn^4cn, sn^4, sn^3cn, sn^3, sn^2cn, sn^2, $sncn$, sn, cn, sn^0 to zero, we deduce respectively

$$\left(\frac{k^2a^2}{\beta} - \frac{\omega^2}{\beta}\right)\left(-2m^2b_2^2 + 2a_2^2m^2\right) + b\left(a_2^3 - 3a_2b_2^2\right) = 0, \tag{40}$$

$$4\left(\frac{k^2a^2}{\beta} - \frac{\omega^2}{\beta}\right)a_2m^2b_2 + b\left(3a_2^2b_2 - b_2^3\right) = 0, \tag{41}$$

$$\left(\frac{k^2a^2}{\beta} - \frac{\omega^2}{\beta}\right)\left(-4m^2b_1b_2 + 4a_1m^2a_2\right) + b\left(3a_1a_2^2 - 6b_1a_2b_2 - 3a_1b_2^2\right) = 0, \tag{42}$$

$$\left(\frac{k^2a^2}{\beta} - \frac{\omega^2}{\beta}\right)\left(4a_1m^2b_2 + 4m^2b_1a_2\right) + b\left(-3b_1b_2^2 + 3b_1a_2^2 + 6a_1a_2b_2\right) = 0, \tag{43}$$

$$\left(\frac{k^2a^2}{\beta} - \frac{\omega^2}{\beta}\right)\left(-m^2b_1^2 + 2m^2b_2^2 + a_1{}^2m^2 + 6a_2m^2a_0\right) + b\left(3a_1^2a_2 + 3a_0a_2^2 - 6a_1b_1b_2 + 3a_2b_2^2 - 3b_1^2a_2 - 3a_0b_2^2\right) = 0, \tag{44}$$

$$\left(\frac{k^2a^2}{\beta} - \frac{\omega^2}{\beta}\right)\left(2a_1m^2b_1 + 6m^2b_2a_0 - a_2m^2b_2\right) + b\left(-3b_1^2b_2 + 3a_1^2b_2 + b_2^3 + 6a_0a_2b_2 + 6a_1b_1a_2\right) = 0, \tag{45}$$

$$\begin{aligned} & \left(\frac{k^2a^2}{\beta} - \frac{\omega^2}{\beta}\right)\left(b_1b_2 - a_1a_2 - a_1m^2a_2 + 2a_1m^2a_0 + 7m^2b_1b_2\right) \\ & + b\left(a_1^3 - 3a_1b_1^2 - 6a_0b_1b_2 + 6a_0a_1a_2 + 3a_1b_2^2 + 6b_1a_2b_2\right) = 0, \end{aligned} \tag{46}$$

$$\left(\frac{k^2a^2}{\beta}-\frac{\omega^2}{\beta}\right)\left(-b_1a_2-a_1b_2+2m^2b_1a_0-4m^2b_1a_2\right)+b\left(3a_1^2b_1-b_1^3+3b_1b_2^2+6a_0a_1b_2+6a_0b_1a_2\right)=0, \quad (47)$$

$$\left(\frac{k^2a^2}{\beta}-\frac{\omega^2}{\beta}\right)\left(-4a_2a_0+2m^2b_1^2-4a_2m^2a_0-2a_2^2\right)+b\left(3a_0a_1^2-3a_0b_1^2+3a_2a_0^2+3b_1^2a_2+3a_0b_2^2+6a_1b_1b_2\right)=0, \quad (48)$$

$$\left(\frac{k^2a^2}{\beta}-\frac{\omega^2}{\beta}\right)\left(-2a_2b_2-4b_2a_0-m^2b_2a_0-a_1m^2b_1\right)+b\left(6a_0a_1b_1+3b_2a_0^2+3b_1^2b_2\right)=0, \quad (49)$$

$$\left(\frac{k^2a^2}{\beta}-\frac{\omega^2}{\beta}\right)\left(-a_0b_1+2b_1a_2-2a_1b_2\right)+b\left(3a_0^2b_1+b_1^3\right)=0, \quad (50)$$

$$\left(\frac{k^2a^2}{\beta}-\frac{\omega^2}{\beta}\right)\left(-a_0a_1-3b_1b_2-2a_1a_2-a_1m^2a_0-m^2b_1b_2\right)+b\left(3a_0^2a_1+3a_1b_1^2+6a_0b_1b_2\right)=0, \quad (51)$$

$$\left(\frac{k^2a^2}{\beta}-\frac{\omega^2}{\beta}\right)\left(-b_2^2-b_1^2+2a_2a_0-a_1^2\right)+b\left(a_0^3+3a_0b_1^2\right)=0, \quad (52)$$

Solving the above system with the aid of Maple or Mathematica, we have the following solution:

$$a=a,\ b=\frac{-2\left(k^2a^2-\omega^2\right)}{\beta a_2},\ k=k,\ m=\pm1,\ \beta=\beta,\ a_0=-a_2,\ a_1=0,\ a_2=a_2,\ b_1=b_2=0.$$

So that the solve of Equation (36) can be written in the form

$$v=\frac{2\left(k^2a^2-\omega^2\right)}{\beta b}-\frac{2\left(k^2a^2-\omega^2\right)}{\beta b}sn^2, \quad (53)$$

$$u=\frac{1}{\beta}\ln\left(\frac{2\left(k^2a^2-\omega^2\right)}{\beta b}-\frac{2\left(k^2a^2-\omega^2\right)}{\beta b}sn^2\right). \quad (54)$$

When $m=1$, the solution can be in the form

$$v=\frac{2\left(k^2a^2-\omega^2\right)}{\beta b}-\frac{2\left(k^2a^2-\omega^2\right)}{\beta b}\tanh^2(\xi), \quad (55)$$

$$u=\frac{1}{\beta}\ln\left(\frac{2\left(k^2a^2-\omega^2\right)}{\beta b}-\frac{2\left(k^2a^2-\omega^2\right)}{\beta b}\tanh^2(\xi)\right). \quad (56)$$

4. Conclusions

We establish exact solutions for the system of shallow water wave equations and modified Liouville equation which are two of the most fascinating problems of modern mathematical physics.

The extended Jacobian elliptic function expansion method has been successfully used to find the exact traveling wave solutions of some nonlinear evolution equations. As an application, the traveling wave solutions for the system of shallow water wave equations and modified Liouville equation, have been constructed using the extended Jacobian elliptic function expansion method. Let us compare between our results obtained in the present article with the well-known results obtained by other authors using different methods as follows: our results of the system of shallow water wave equations and modified Liouville equation are new and different from those obtained in [27] and [28] and **Figure 1** and **Figure 2** show the solitary wave solution of Equations

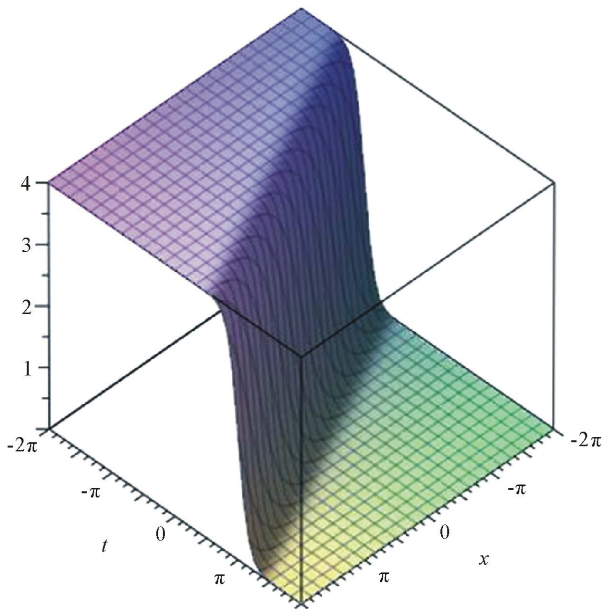


Figure 1. Solitary wave solution of Equation (30).

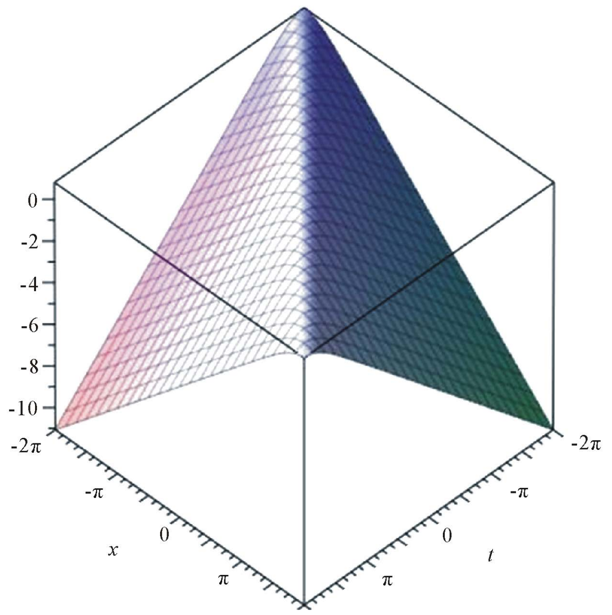


Figure 2. Solitary wave solution of Equation (56).

(30) and (56). It can be concluded that this method is reliable and proposes a variety of exact solutions NPDEs. The performance of this method is effective and can be applied to many other nonlinear evolution equations.

References

[1] Ablowitz, M.J. and Segur, H. (1981) Solitions and Inverse Scattering Transform. SIAM, Philadelphia. http://dx.doi.org/10.1137/1.9781611970883

[2] Malfliet, W. (1992) Solitary Wave Solutions of Nonlinear Wave Equation. *American Journal of Physics*, **60**, 650-654. http://dx.doi.org/10.1119/1.17120

[3] Malfliet, W. and Hereman, W. (1996) The tanh Method: Exact Solutions of Nonlinear Evolution and Wave Equations. *Physica Scripta*, **54**, 563-568. http://dx.doi.org/10.1088/0031-8949/54/6/003

[4] Wazwaz, A.M. (2004) The tanh Method for Travelling Wave Solutions of Nonlinear Equations. *Applied Mathematics and Computation*, **154**, 714-723. http://dx.doi.org/10.1016/S0096-3003(03)00745-8

[5] El-Wakil, S.A. and Abdou, M.A. (2007) New Exact Travelling Wave Solutions Using Modified Extented tanh-Function Method. *Chaos*, *Solitons & Fractals*, **31**, 840-852. http://dx.doi.org/10.1016/j.chaos.2005.10.032

[6] Fan, E. (2000) Extended tanh-Function Method and Its Applications to Nonlinear Equations. *Physics Letters A*, **277**, 212-218. http://dx.doi.org/10.1016/S0375-9601(00)00725-8

[7] Wazwaz, A.M. (2007) The Extended tanh Method for Abundant Solitary Wave Solutions of Nonlinear Wave Equations. *Applied Mathematics and Computation*, **187**, 1131-1142. http://dx.doi.org/10.1016/j.amc.2006.09.013

[8] Wazwaz, A.M. (2005) Exact Solutions to the Double sinh-Gordon Equation by the tanh Method and a Variable Separated ODE Method. *Computers & Mathematics with Applications*, **50**, 1685-1696. http://dx.doi.org/10.1016/j.camwa.2005.05.010

[9] Wazwaz, A.M. (2004) A sine-cosine Method for Handling Nonlinear Wave Equations. *Mathematical and Computer Modelling*, **40**, 499-508. http://dx.doi.org/10.1016/j.mcm.2003.12.010

[10] Yan, C. (1996) A Simple Transformation for Nonlinear Waves. *Physics Letters A*, **224**, 77-84. http://dx.doi.org/10.1016/S0375-9601(96)00770-0

[11] Fan, E. and Zhang, H. (1998) A Note on the Homogeneous Balance Method. *Physics Letters A*, **246**, 403-406. http://dx.doi.org/10.1016/S0375-9601(98)00547-7

[12] Wang, M.L. (1996) Exact Solutions for a Compound KdV-Burgers Equation. *Physics Letters A*, **213**, 279-287. http://dx.doi.org/10.1016/0375-9601(96)00103-X

[13] Abdou, M.A. (2007) The Extended F-Expansion Method and Its Application for a Class of Nonlinear Evolution Equations. *Chaos*, *Solitons & Fractals*, **31**, 95-104. http://dx.doi.org/10.1016/j.chaos.2005.09.030

[14] Ren, Y.J. and Zhang, H.Q. (2006) A Generalized F-Expansion Method to Find Abundant Families of Jacobi Elliptic Function Solutions of the (2 + 1)-Dimensional Nizhnik-Novikov-Veselov Equation. *Chaos*, *Solitons & Fractals*, **27**, 959-979. http://dx.doi.org/10.1016/j.chaos.2005.04.063

[15] Zhang, J.L., Wang, M.L., Wang, Y.M. and Fang, Z.D. (2006) The Improved F-Expansion Method and Its Applications. *Physics Letters A*, **350**, 103-109. http://dx.doi.org/10.1016/j.physleta.2005.10.099

[16] He, J.H. and Wu, X.H. (2006) Exp-Function Method for Nonlinear Wave Equations. *Chaos*, *Solitons & Fractals*, **30**, 700-708. http://dx.doi.org/10.1016/j.chaos.2006.03.020

[17] Zahran, E.H.M. and Khater, M.M.A. (2014) The Modified Simple Equation Method and Its Applications for Solving Some Nonlinear Evolutions Equations in Mathematical Physics. *Jokull Journal*, **64**.

[18] Abdelrahman, M.A.E., Zahran, E.H.M. and Khater, M.M.A. (2014) Exact Traveling Wave Solutions for Power Law and Kerr Law Non Linearity Using the $\exp\left(-\phi(\xi)\right)$-Expansion Method. *Global Journal of Science Frontier Research*, **14-F**.

[19] Wang, M.L., Zhang, J.L. and Li, X.Z. (2008) The $\left(\frac{G'}{G}\right)$-Expansion Method and Travelling Wave Solutions of Nonlinear Evolutions Equations in Mathematical Physics. *Physics Letters A*, **372**, 417-423. http://dx.doi.org/10.1016/j.physleta.2007.07.051

[20] Zhang, S., Tong, J.L. and Wang, W. (2008) A Generalized $\left(\frac{G'}{G}\right)$-Expansion Method for the mKdv Equation with Variable Coefficients. *Physics Letters A*, **372**, 2254-2257. http://dx.doi.org/10.1016/j.physleta.2007.11.026

[21] Zayed, E.M.E. and Gepreel, K.A. (2009) The $\left(\frac{G'}{G}\right)$-Expansion Method for Finding Traveling Wave Solutions of Nonlinear Partial Differential Equations in Mathematical Physics. *Journal of Mathematical Physics*, **50**, Article ID: 013502. http://dx.doi.org/10.1063/1.3033750

[22] Zahran, E.H.M. and Khater, M.M.A. (2014) Exact Solution to Some Nonlinear Evolution Equations by the $\left(\frac{G'}{G}\right)$-Expansion Method. *Jokull Journal*, **64**.

[23] Zaki, S.I. (2000) Solitary Wave Interactions for the Modified Equal width Wave Equation. *Computer Physics Communications*, **126**, 219-213. http://dx.doi.org/10.1016/S0010-4655(99)00471-3

[24] Fan, E. and Zhang, J. (2002) Applications of the Jacobi Elliptic Function Method to Special-Type Nonlinear Equations. *Physics Letters A*, **305**, 383-392. http://dx.doi.org/10.1016/S0375-9601(02)01516-5

[25] Liu, S., Fu, Z., Liu, S. and Zhao, Q. (2001) Jacobi Elliptic Function Expansion Method and Periodic Wave Solutions of Nonlinear Wave Equations. *Physics Letters A*, **289**, 69-74. http://dx.doi.org/10.1016/S0375-9601(01)00580-1

[26] Zhao, X.Q., Zhi, H.Y. and Zhang, H.Q. (2006) Improved Jacobi-Function Method with Symbolic Computation to Construct New Double-Periodic Solutions for the Generalized Ito System. *Chaos*, *Solitons & Fractals*, **28**, 112-126. http://dx.doi.org/10.1016/j.chaos.2005.05.016

[27] Dolapci, İ.T. and Yildirim, A. (2013) Some Exact Solutions to the Generalized Korteweg-de Vries Equation and the System of Shallow Water Wave Equation. *Nonlinear Analysis*, *Modeling and Control*, **18**, 27-36.

[28] Salam, M.A. (2012) Traveling-Wave Solution of Modified Liouville Equation by Means of Modified Simple Equation Method. *International Scholarly Research Network ISRN Applied Mathematics*, **2012**, Article ID: 565247.

American Journal of Computational Mathematics, 2014, 4, 464-473
Published Online December 2014 in SciRes. http://www.scirp.org/journal/ajcm
http://dx.doi.org/10.4236/ajcm.2014.45039

Combining Algebraic and Numerical Techniques for Computing Matrix Determinant

Mohammad M. Tabanjeh

Department of Mathematics and Computer Science, Virginia State University, Petersburg, USA
Email: mtabanjeh@vsu.edu

Received 22 October 2014; revised 1 December 2014; accepted 9 December 2014

Abstract

Computing the sign of the determinant or the value of the determinant of an $n \times n$ matrix A is a classical well-know problem and it is a challenge for both numerical and algebraic methods. In this paper, we review, modify and combine various techniques of numerical linear algebra and rational algebraic computations (with no error) to achieve our main goal of decreasing the bit-precision for computing det$A$ or its sign and enable us to obtain the solution with few arithmetic operations. In particular, we improved the precision $\lceil H\log_2 p \rceil$ bits of the p-adic lifting algorithm ($H = 2^h$ for a natural number h), which may exceed the computer precision β (see Section 5.2), to at most $\lceil \beta \rceil$ bits (see Section 6). The computational cost of the p-adic lifting can be performed in $O(hn^4)$. We reduced this cost to $O(n^3)$ by employing the faster p-adic lifting technique (see Section 5.3).

Keywords

Matrix Determinant, Sign of the Determinant, *p*-Adic Lifting, Modular Determinant, Matrix Factorization, Bit-Precision

1. Introduction

Computation of the sign of the determinant of a matrix and even the determinant itself is a challenge for both numerical and algebraic methods. That is, to testing whether $\det(A) > 0$, $\det(A) < 0$, or $\det A = 0$ for an $n \times n$ matrix A. In computational geometry, most decisions are based on the signs of the determinants. Among the geometric applications, in which the sign of the determinant needs to be evaluated, are the computations of convex hulls, Voronoi diagrams, testing whether the line intervals of a given family have nonempty common intersection, and finding the orientation of a high dimensional polyhedron. We refer the reader to [1]-[5], and to the bibliography therein for earlier work. These applications have motivated extensive algorithmic work on computing the value and particulary the sign of the determinant. One of the well-known numerical algorithms is Clarkson's algorithm. In [6], Clarkson uses an adaption of Gram-Schmidt procedure for computing an orthogonal basis and employs approximate arithmetics. His algorithm runs in time $O\left(d^3 b\right)$ and uses $2b+1.5d$ bits to represent the values for a $d \times d$ determinant with b-bit integer entries. On the other hand, the authors of [7] proposed a method for evaluating the signs of the determinant and bounding the precision in the case where $n \le 3$. Their algorithm asymptotic worst case is worse than that of Clarkson. However, it is simpler and uses respectively b and $(b+1)$-bit arithmetic. In this paper we examine two different approaches of computing the determinant of a matrix or testing the sign of the determinant; the numerical and the algebraic approach. In particular, numerical algorithms for computing various factorizations of a matrix A which include the orthogonal factorization $A = QR$ and the triangular factorizations $A = LU$, $A = P_r LU$, and $A = P_r LUP_c$ seem to be the most effective algorithms for computing the sign of $\det(A)$ provided that $|\det A|$ is large enough relative to the computer precision. Alternatively, the algebraic techniques for computing $\det A$ modulo an integer M based on the Chinese remainder theorem and the p-adic lifting for a prime p use lower precision or less arithmetic operations. In fact, the Chinese remainder theorem required less arithmetic operations than the p-adic lifting but with higher precision due to (9). We also demonstrate some effective approaches of combining algebraic and numerical techniques in order to decrease the precision of the computation of the p-adic lifting and to introduce an alternative technique to reduce the arithmetic operations. If $\det A$ is small, then obtaining the sign of the determinant with lower precision can be done by effective algebraic methods of Section 4. Although, the Chinese remainder algorithm approach requires low precision computations provided that the determinant is bounded from above by a fixed large number. This motivated us to generalize to the case of any input matrix A and to decrease the precision at the final stage of the Chinese remaindering at the price of a minimal increase in the arithmetic cost. Furthermore, in Section 5 we extend the work to an algorithm that computes $(\det A) \bmod M$ using low precision by relying on the p-adic lifting rather than the Chinese remaindering.

2. Definitions, Basic Facts, and Matrix Norms

Definition 1. *For an n × n matrix A, the determinant of A is given by either*

$$\det(A) = |A| = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij}), \text{ for } i = 1, \cdots, n, \quad \text{or} \quad \det(A) = |A| = \sum_{i=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij}), \text{ for } j = 1, \cdots, n,$$

where A_{ij} *is* $(n-1)$*-by-*$(n-1)$ *matrix obtained by deleting the i-th row and j-th column of A.*

Fact 1. *Well-known properties of the determinant include* $\det(AB) = \det(A)\det(B)$, $\det(A^{\mathrm{T}}) = \det(A)$, *and* $\det(cA) = c^n \det(A)$ *for any two matrices* $A, B \in \mathbb{R}^{n\times n}$ *and* $c \in \mathbb{R}$.

Definition 2. *If A is a triangular matrix of order n, then its determinant is the product of the entries on the main diagonal.*

Definition 3. *For a matrix A of size m × n, we define* 1*-norm*: $\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}|$, ∞*-norm*: $\|A\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}|$, *p-norms*: $\|A\|_p = \sup_{x\ne 0} \frac{\|Ax\|_p}{\|x\|_p} = \sup_{x\ne 0} \left\| A\left(\frac{x}{\|x\|_p}\right)\right\|_p = \max_{\|x\|_p = 1} \|Ax\|_p$, *and* 2*-norm*:

$$\|A\|_2 = \max_{\|x\|_2 \ne 0} \frac{\|Ax\|_2}{\|x\|_2}.$$

3. Numerical Computations of Matrix Determinant

We find the following factorizations of an $n \times n$ matrix A whose entries are either integer, real, or rational by Gaussian elimination.

$$A = LU \tag{1}$$

$$A = P_r LU \tag{2}$$

$$A = P_r LUP_c. \tag{3}$$

We reduce A to its upper triangular form U. The matrix L is a unit lower triangular whose diagonal entries are all 1 and the remaining entries of L can be filled with the negatives of the multiples used in the elementary operations. P_r is a permutation matrix results from row interchange and P_c is a permutation matrix results from column interchange.

Fact 2. *If P is a permutation matrix, then* $\det P = \pm 1$. *Moreover,* $\det I = 1$, *where I is an identity matrix.*

3.1. Triangular Factorizations

The following algorithm computes $\det(A)$ or its sign based on the factorizations (1)-(3).

Algorithm 1. *Triangular factorizations.*

Input: *An n × n matrix A.*

Output: $\det(A)$.

Computations:

1) *Compute the matrices L, U satisfying Equation* (1), *or* P_r, *L, U satisfying Equation* (2), *or* P_r, *L, U,* P_c *satisfying Equation* (3).

2) *Compute* $\det L$, $\det U$ *for Equation* (1), *or* $\det P_r$, $\det L$, $\det U$ *for Equation* (2), *or* $\det P_r$, $\det L$, $\det U$, $\det P_c$ *for Equation* (3).

3) *Compute and output* $\det A = (\det L)(\det U)$ *for Equation* (1), *or* $\det A = (\det P_r)(\det L)(\det U)$ *for Equation* (2), *or* $\det(A) = (\det P_r)(\det L)(\det U)(\det P_c)$ *for Equation* (3).

One can easily output the sign of $\det A$ from the above algorithm. We only need to compute the sign of $\det P_r$, $\det L$, $\det U$, and $\det P_c$ at stage 2 and multiply these signs. The value of $\det P_r$ and $\det P_c$ can be easily found by tracking the number of row or column interchanges in Gaussian elimination process which will be either 1 or −1 since those are permutation matrices. As L is a unit lower triangular matrix, $\det L = 1$. Finally, the computations of $\det U$ required $n-1$ multiplications since U is an upper triangular matrix. Therefore, the overall arithmetic operations of Algorithm 1 will be dominated by the computational cost at stage 1, that is, $\frac{n(n-1)(2n-1)}{6} = \sum_{i=1}^{n-1} i^2$ multiplications, the same number for additions, subtractions, and comparisons, but $\frac{n(n-1)}{2} = \sum_{i=1}^{n-1} i$ divisions. However, Algorithm 1 uses $\sum_{i=1}^{n-1} i$ comparisons to compute (2) rather than $\sum_{i=1}^{n-1} i^2$.

3.2. Orthogonal Factorization

Definition 4. *A square matrix Q is called an orthogonal matrix if* $Q^{\mathrm{T}}Q = I$. *Equivalently,* $Q^{-1} = Q^{\mathrm{T}}$, *where* Q^{T} *is the transpose of Q.*

Lemma 1. *If Q is an orthogonal matrix, then* $\det(Q) = \pm 1$.

Proof. Since Q is orthogonal, then $Q^{\mathrm{T}}Q = I$. Now $|Q^{\mathrm{T}}Q| = |I| = 1$ and thus $|Q||Q^{\mathrm{T}}| = 1$, but $|Q^{\mathrm{T}}| = |Q|$ which implies that $1 = |Q||Q| = |Q|^2$, hence $|Q| = \pm 1$.

Definition 5. *If* $A \in \mathbb{R}^{m\times n}$, *then there exists an orthogonal matrix* $Q \in \mathbb{R}^{m\times m}$ *and an upper triangular matrix* $R \in \mathbb{R}^{m\times n}$ *so that*

$$A = QR. \tag{4}$$

Algorithm 2. *QR-factorization.*

Input: *An n × n matrix A.*

Output: $\det(A)$.

Computations:

1) *Compute an orthogonal matrix Q satisfying the Equation* (4).

2) *Compute* $\det Q$.

3) *Compute the matrix* $R = Q^{\mathrm{T}}A = (r_{i,j})$.

4) *Compute* $\det(A) = (\det Q)\prod_i r_{i,i}$, *where* $r_{i,i}$ *are the main diagonal entries of R.*

Here, we consider two well-known effective algorithms for computing the QR factorization; the Householder and the Given algorithms. The Householder transformation (or reflection) is a matrix of the form $H = I - 2\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}$ where $\|\boldsymbol{v}\|_2 = 1$.

Lemma 2. *H is symmetric and orthogonal. That is,* $H^{\mathrm{T}} = H$ *and* $H^{\mathrm{T}} \cdot H = I$.

Proof. $H = I - 2\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}$, and since $I^{\mathrm{T}} = I$ and $(\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}})^{\mathrm{T}} = \boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}$, then $H^{\mathrm{T}} = I^{\mathrm{T}} - (2\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}})^{\mathrm{T}} = I - 2\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}} = H$. Hence,

H is symmetric. On the other hand, $H^{\mathrm{T}}H = \left(I - 2\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}\right)\left(I - 2\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}\right) = I - 4\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}} + 4v\left(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v}\right)\boldsymbol{v}^{\mathrm{T}} = I - 4\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}} + 4\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}} = I$. Therefore, H is orthogonal, that is, $H^{-1} = H^{\mathrm{T}}$.

The Householder algorithm computes $Q = \prod_{i}^{n-1} H_i$ as a product of $n-1$ matrices of Householder transformations, and the upper triangular matrix $Q^{\mathrm{T}}A = R$.

Lemma 3. $\det(Q) = (-1)^{n-1}$ *for a matrix Q computed by the Householder algorithm.*

Proof. Since $H_i = I - 2\boldsymbol{v}_i\boldsymbol{v}_i^{\mathrm{T}}$ for vectors $\boldsymbol{v}_i$, $i = 1,\cdots,n-1$, Lemma 2 implies that $H_i^{\mathrm{T}}H_i = I$ and hence $(\det H_i)^2 = 1$. Notice that $\det\left(I - 2\boldsymbol{e}_1\boldsymbol{e}_1^{\mathrm{T}}\right) = -1$ where $e_1 = (1,0,\cdots,0)^{\mathrm{T}}$. Now for $0 \le t \le 1$, we define the transformation $v(t) = t(v_i - e_1) + e_1$. Then the function $D(t) = \det\left(I - v(t)v(t)^{\mathrm{T}}\right)$ is continuous and $D^2 = 1 \quad \forall t$. Hence $D(1) = \det\left(I - 2v_1v_1^{\mathrm{T}}\right) = -1$ since $\|\boldsymbol{v}\|_2 = 1$ and $D(0) = \det\left(I - 2e_1e_1^{\mathrm{T}}\right) = -1$. This proves that $\det H_i = -1 \quad \forall i$. But $Q = \prod_{i=1}^{n-1} H_i$, we have $\det Q = (-1)^{n-1}$.

The Givens rotations can also be used to compute the *QR* factorization, where $Q = G_1 \cdots G_t$, t is the total number of rotations, G_j is the the j-th rotation matrix, and $Q^{\mathrm{T}}A = R$ is an upper triangular matrix. Since the Givens algorithm computes Q as a product of the matrices of Givens rotations, then $\det(Q) = 1$. We define $G = W_{i,i}(c,s) = W_{k,k}(c,s) = c$, $G = W_{i,k} = -W_{k,i} = s$ for two fixed integers i and k, and for a pair of nonzero real numbers c and s that satisfies $c^2 + s^2 = 1$ such that $G^{\mathrm{T}}G = I$. If the Householder is used to find the *QR*, Algorithm 2 uses $n-1$ multiplications at stage 3. It involves $\frac{4}{3}n^3 + O(n^2)$ multiplications at stage 1, and $O(n)$ evaluations of the square roots of positive numbers. If the Givens algorithm is used to find the *QR*, then it involves $2n^3 + O(n^2)$ multiplications at stage 3, $n^3 + O(n^2)$ multiplications at stage 1, and $O(n^2)$ evaluations. This shows some advantage of the Householder over the Givens algorithm. However, the Givens rotations algorithm allows us to update the *QR* of *A* successively by using only $O(kn^2)$ arithmetic operations and square root evaluations if $O(k)$ rows or columns of *A* are successively replaced by new ones. Therefore, in this case, the Givens algorithm is more effective.

4. Algebraic Computations of Matrix Determinant

4.1. Determinant Computation Based on the Chinese Remainder Theorem

Theorem 1. (*Chinese remainder theorem.*) *Let* $m_1, m_2, \cdots, m_k$ *be distinct pairwise relatively prime integers such that*

$$m_1 > m_2 > \cdots > m_k > 1. \tag{5}$$

Let $M = \prod_{i=1}^{k} m_i$, and let D denote an integer satisfying the inequality

$$0 \le D < M. \tag{6}$$

Let

$$r_i = D(\operatorname{mod} m_i). \tag{7}$$

$$M_i = \frac{M}{m_i},\ s_i \equiv M_i(\operatorname{mod} m_i),\ y_i s_i \equiv 1(\operatorname{mod} m_i),\ \forall\, i = 1,\cdots,k. \tag{8}$$

Then D is a unique integer satisfying (6) and (7). In addition,

$$D = \sum_{i=1}^{k} (M_i r_i y_i)(\operatorname{mod} M). \tag{9}$$

Algorithm 3. (*Computation of* $\det A(\operatorname{mod} M)$ *based on the Chinese remainder theorem.*)

Input: *An integer matrix A, an algorithm that computes* $\det A(\operatorname{mod} m)$ *for any fixed integer* $m > 1$, k *integers* $m_1, \cdots, m_k$ *satisfying* (5) *and are pairwise relatively prime, and* $M = m_1 m_2 \cdots m_k$.

Output: $\det A(\operatorname{mod} M)$.

Computations:

1) *Compute* $r_i = \det A(\operatorname{mod} m_i)$, $i = 1,\cdots,k$.

2) *Compute the integers* M_i, s_i, *and* y_i *as in* (8) $\forall\, i = 1,\cdots,k$.

3) *Compute and output D as in* (9).

The values of y_i in (8) are computed by the Euclidean algorithm when applied to s_i and m_i for $i = 1, \cdots, k$. Algorithm 3 uses $O(kn^3)$ arithmetic operations (ops) at stage 1, $O((k \log m_1) \log \log m_1)$ ops at stage 2, and $O(k \log^2 k)$ ops at stage 3. The computations of the algorithm are performed with precision of at most $\lceil \log_2 m_1 \rceil$ bits at stage 1, and at most $\lceil \log_2 M \rceil$ bits at stages 2 and 3. The latest precision can be decreased at the cost of slightly increasing the arithmetic operations [8].

Lemma 4. *For any pair of integers M and d such that* $M > 2|d|$, *we have*

$$d = (d \bmod M) \text{ if } (d \bmod M) < \frac{M}{2}, \text{ and } d = (d \bmod M) - M \text{ otherwise.} \tag{10}$$

Suppose that the input matrix A is filled with integers and an upper bound $d_u \geq |\det A|$ is known. Then we may choose an integer M such that $M > 2|d_u|$ and compute $\det A (\bmod M)$. Hence, $\det A$ can be recovered by using (10).

4.2. Modular Determinant

Let $A \in \mathbb{Z}^{n \times n}$ and $d = \det(A)$. In order to calculate $\det A(\bmod p)$, we choose a prime p that is bigger than $2|d|$ and perform Gaussian elimination (2) on $A(\bmod p) \in \mathbb{Z}_p^{n \times n}$. This is the same as Gaussian elimination over $\mathbb{Q}$, except that when dividing by a pivot element a we have to calculate its inverse modulo p by the extended Euclidean algorithm. This requires $\frac{2}{3}n^3 + O(n^2)$ arithmetic operations modulo p. On the other hand, if we need to compute $\det A(\bmod p)$ for the updated matrix A, we rely on the QR factorization such that $A = (QR)(\bmod p)$ and $Q^{\mathrm{T}}Q = D(\bmod p)$, where D is a diagonal matrix and R is a unit upper triangular matrix. The latest factorization can be computed by the Givens algorithm [9]. If the entries of the matrix A are much smaller than p, then we do not need to reduce modulo p the results at the initial steps of Gaussian elimination. That is, the computation can be performed in exact rational arithmetics using lower precision. In this case, one may apply the algorithm of [10] and [11] to keep the precision low. The computations modulo a fixed integer $M > 1$ can be performed with precision $\lceil \log_2 M \rceil$ bits. In such a computation, we do not need to worry about the growth of the intermediate values anymore. However, to reduce the cost of the computations, one can work with small primes modular instead of a large prime, that is, these primes can be chosen very small with logarithmic length. Then the resulting algorithm will have lower cost of $O(n^3 k)$ and can be performed in parallel. Now, one can find a bound on the $\det A$ without actually calculating the determinant. This bound is given by Hadamard's inequality which says that

$$|\det A| \leq n^{\frac{n}{2}} a^n = \left(a\sqrt{n}\right)^n, \tag{11}$$

where $a = \max_{1 \leq i, j \leq n} |a_{ij}| \in \mathbb{Z}^+$ (nonnegative integers) is the maximal absolute value of an entry of A.

5. Alternative Approach for Computing Matrix Inverses and Determinant

5.1. *p*-Adic Lifting of Matrix Inverses

In this section, we present an alternative approach for computing matrix determinant to one we discussed in earlier sections. The main goal is to decrease the precision of algebraic computations with no rounding errors. The technique relies on Newton's-Hensel's lifting (p-adic lifting) and uses $O(n^4 \log n)$ arithmetic operations. However, we will also show how to modify this technique to use order of n^3 arithmetic operations by employing the faster p-adic lifting. Given an initial approximation to the inverse, say S_0, a well-known formula for Newton's iteration rapidly improves the initial approximation to the inverse of a nonsingular $n \times n$ matrix A:

$$S_{i+1} = S_i(2I - AS_i) = 2S_i - S_i AS_i = (2I - S_i A)S_i, \quad i \geq 0. \tag{12}$$

Algorithm 4. (*p-adic lifting of matrix inverses.*)
Input: *An n × n matrix A, an integer* $p > 1$, *the matrix* $S_0 = A^{-1}(\bmod p)$, *and a natural number h.*
Output: *The matrix* $A^{-1}(\bmod p^H)$, $H = 2^h$.
Computations:
1) *Recursively compute the matrix* S_j *by the equation,*

$$S_j = S_{j-1}\left(2I - AS_{j-1}\right)\left(\bmod p^J\right), \text{ where } J = 2^j,\ j = 1,\cdots,h. \tag{13}$$

2) *Finally, outputs* S_h.

Note that, $S_j = A^{-1}\left(\bmod p^J\right)$, $J = 2^j$, $\forall j$. This follows from the equation $I - AS_j = \left(I - AS_{j-1}\right)^2\left(\bmod p^J\right)$. The above algorithm uses $O\left(n^3\right)$ arithmetic operations performed modulo p^J at the *j*-th stage, that is, a total of $O\left(hn^3\right)$ arithmetic operations with precision of at most $\lceil J\log_2 p\rceil$ bits.

5.2. *p*-Adic Lifting of Matrix Determinant

We can further extend *p*-adic lifting of matrix inverses to *p*-adic lifting of matrix determinants, that is $\det A\left(\bmod p\right)$, using the following formula [12]:

$$\det A = \frac{1}{\prod_{k=1}^{n}\left(A_k^{-1}\right)_{k,k}}. \tag{14}$$

Here, A_k denotes the $k \times k$ leading principal (north-western) submatrix of *A* so that $A_n = A$ for $k = 1,\cdots,n$. Moreover, $(B)_{k,k}$ denotes the (k,k)-th entry of a matrix *B*. In order to use Formula (14), we must have the inverses of A_k available modulo a fixed prime integer *p* for all k. A nonsingular matrix *A* modulo *p* is called strongly nonsingular if the inverses of all submatrices A_k exist modulo *p*. In general, a matrix *A* is called strongly nonsingular if *A* is nonsingular and all $k \times k$ leading principle submatrices are nonsingular. Here, we assume that *A* is strongly nonsingular for a choice of *p*. Finally, Algorithm 4 can be extended to lifting $\det A$ (see Algorithm 5).

Algorithm 5. (*p-adic lifting of matrix determinant.*)

Input: *An integer* $p > 1$, *an* $n \times n$ *matrix A, the matrices* $S_{0,k} = A_k^{-1}\left(\bmod p\right)$, *and a natural number h.*

Output: $\det A\left(\bmod p^{2H}\right)$, $H = 2^h$.

Computations:

1) *Apply Algorithm* 4 *to all pairs of matrices* A_k *and* $S_{0,k}$, (*replacing the matrices A and* S_0 *in the algorithm*), *so as to compute the matrices* $S_{h,k} = A_k^{-1}\left(\bmod p^H\right)$, *for* $k = 1,\cdots,n$.

2) *Compute the value*

$$\left(\frac{1}{\det A}\right)\bmod p^{2H} = \prod_{k=1}^{n}\left[S_{h,k}\left(2I - A_k S_{h,k}\right)\right]_{k,k}\left(\bmod p^{2H}\right). \tag{15}$$

3) *Compute and output the value* $(\det A)\left(\bmod p^{2H}\right)$, *as the reciprocal of* $\left(\frac{1}{\det A}\right)\left(\bmod p^{2H}\right)$.

The overall computational cost of Algorithm 5 at stage 1 is $O\left(hn^4\right)$ arithmetic operations performed with precision at most $\lceil H\log_2 p\rceil$ bits. However, at stage 2, the algorithm uses $O\left(n^3\right)$ operations with precision $\lceil 2H\log_2 p\rceil$ bits. At stage 3, only one single multiplication is needed. All the above operations are calculated modulo p^{2H}. Now, we will estimate the value of *H* and *p* that must satisfy the bound $p^{2H} > 2\left|\det A\right|$. But, due to Hadamard's bound (11), we have $2\left|\det(A)\right| < 2\left(a\sqrt{n}\right)^n < p^{2H}$ which implies that $2H > n\log_p a + n\log_p\sqrt{n} + \log_p 2$. Therefore, it is suffices to choose *p* and *H* satisfying the inequalities $2\left(a\sqrt{n}\right)^n < p^{2H}$ and $2H > \log_p\left(2\left(a\sqrt{a}\right)^n\right)$. In the next section, we will present an alternative faster technique to computing matrix determinant that uses only $O\left(n^3\right)$ based on the divide-and-conquer algorithm.

Example 1. *Let* $A = \begin{pmatrix} 1 & 17 & 18 \\ 1 & 18 & 19 \\ 5 & 16 & 20 \end{pmatrix}$. *Then,* $A_1 = [1]$, $A_2 = \begin{pmatrix} 1 & 17 \\ 1 & 18 \end{pmatrix}$, *and* $A_3 = \begin{pmatrix} 1 & 17 & 18 \\ 1 & 18 & 19 \\ 5 & 16 & 20 \end{pmatrix}$. *By Algorithm* 4, *compute the matrices*: $A_1^{-1} = [1]$, $A_2^{-1} = \begin{pmatrix} 18 & -17 \\ -1 & 1 \end{pmatrix}$, *and* $A_3^{-1} = \begin{pmatrix} -56 & 52 & 1 \\ -75 & 70 & 1 \\ 74 & -69 & -1 \end{pmatrix}$. *Now, we compute*

$S_{0,1} = A_1^{-1} \bmod 3 = [1]$, $S_{0,2} = A_2^{-1} \bmod 3 = \begin{pmatrix} 0 & 1 \\ 2 & 1 \end{pmatrix}$, *and* $S_{0,3} = A_3^{-1} \bmod 3 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 2 & 0 & 2 \end{pmatrix}$. *We apply Algorithm* 4 (*that is,* $S_k = S_{0,k}\left(2I - A_k S_{0,k}\right) \bmod p^J$, $J = 2^j$, $j = 1, \cdots, h$) *to all pairs of matrices*:

$$S_{1,1} = S_1 = S_{0,1}\left(2I - A_1 S_{0,1}\right) \bmod 3^2 = [1],$$

$$S_{1,2} = S_2 = S_{0,2}\left(2I - A_2 S_{0,2}\right) \bmod 3^2 = \begin{pmatrix} 0 & 1 \\ 8 & 1 \end{pmatrix},$$

and

$$S_{1,3} = S_3 = S_{0,3}\left(2I - A_3 S_{0,3}\right) \bmod 3^2 = \begin{pmatrix} 7 & 7 & 1 \\ 6 & 7 & 1 \\ 2 & 3 & 8 \end{pmatrix}.$$

We then compute the value

$$\begin{aligned} \frac{1}{\det A} \bmod p^{2H} &= \prod_{k=1}^{3} \left[S_{1,1}\left(2I - A_1 S_{1,1}\right)\right]_{1,1} \cdot \left[S_{1,2}\left(2I - A_2 S_{1,2}\right)\right]_{2,2} \cdot \left[S_{1,3}\left(2I - A_3 S_{1,3}\right)\right]_{3,3} \bmod 3^{2\times2\times1} \\ &= [1]_{1,1} \cdot \begin{pmatrix} -144 & -17 \\ -1216 & -161 \end{pmatrix}_{2,2} \cdot \begin{pmatrix} -2243 & -2783 & -2510 \\ -2100 & -2603 & -2348 \\ -2113 & -2580 & -2269 \end{pmatrix}_{3,3} \\ &= (1)(-161)(-2269) = 365309 (\bmod 81) = 80. \end{aligned}$$

Therefore, we output the value of $(\det A) \bmod 3^{2\times2\times1} = -1$, *since* $-1 \equiv 80 \bmod 81$.

5.3. Faster *p*-Adic Lifting of the Determinant

Assuming *A* is strongly nonsingular modulo *p*, block Gauss-Jordan elimination can be applied to the block 2 × 2 matrix $A = \begin{pmatrix} B & C \\ E & G \end{pmatrix}$ to obtain the well-known block factorization of *A*:

$$A = \begin{pmatrix} I & O \\ EB^{-1} & I \end{pmatrix} \begin{pmatrix} B & O \\ O & S \end{pmatrix} \begin{pmatrix} I & B^{-1}C \\ O & I \end{pmatrix},$$

where $S = G - EB^{-1}C$ is the Schur complement of *B* in *A*. The following is the divide-and-conquer algorithm for computing $\det(A)$.

Algorithm 6. (*Computing the determinant of a strongly nonsingular matrix.*)

Input: *An n × n strongly nonsingular matrix A.*

Output: $\det(A)$.

Computations:

1) *Partition A according to the block factorization above, where B is an* $\left\lfloor \frac{n}{2} \right\rfloor \times \left\lfloor \frac{n}{2} \right\rfloor$ *matrix.* ($\lfloor \cdot \rfloor$ *refers to the floor of* $\frac{n}{2}$.) *Compute* B^{-1} *and S using the matrix equation* $S = G - EB^{-1}C$.

2) *Compute* $\det(B)$ *and* $\det(S)$.

3) *Compute and output* $\det(A) = (\det B)\det S$.

Since *A* is strongly nonsingular modulo *p*, we can compute the inverse of *k* × *k* matrix by $O\left(k^3 \log H\right) = I(k)$ arithmetic operations using Algorithm 4. From the above factorization of *A*, we conclude that $D(n) \le D(\lfloor n/2 \rfloor) + D(\lceil n/2 \rceil) + I(\lfloor n/2 \rfloor)$. The computational cost of computing the determinant is $D(n) = O\left(n^3 \log H\right)$. This can be derived from the above factorization of *A* using recursive factorization applied to *B* and *S* and the inverses modulo p^{2H}. Here, $I(k)$ is the cost of computing the inverse and $D(k)$ is the cost of computing the determinant.

6. Improving the Precision of the *p*-Adic Lifting Algorithm

Definition 6. *Let e be a fixed integer such that* $\log_2 e > \beta$ *where β is the computer precision. Then any integer z, such that* $0 \le z \le e-1$, *will fit the computer precision and will be called short integer or e-integer. The integers that exceed* $e-1=\alpha$ *will be called long integers and we will associate them with the e-polynomials* $p(x)=\sum_i p_i x^i$, $0 \le p_i < e$ *for all i.*

Recall that Algorithm 4 uses $\lceil j \log_2 p \rceil$ bits at stage *j*. For large *j*, this value is going to exceed *β*. In this section we decrease the precision of the *p*-adic algorithm and keep it below *β*. In order to do this, we choose the base $e = p^K$ where *K* is a power of 2, $2K \le H$, *K* divides $H/2$, and

$$K \le \log_2\left(\frac{\alpha^2 nH}{K}\right) < \beta. \tag{16}$$

Now, let us associate the entries of the matrices $A_k \bmod p^J$ and $S_{j-1,k} \bmod p^J$ with the *e*-polynomials in *x* for $J = 2^j$ and for all *j* and *k*. These polynomials have degrees $(J/K)-1$ and take values of the entries for $x=e$. Define the polynomial matrices $A_{j,k}(e)=A_k \bmod p^J$ and $S_{j,k}(e)=S_{j,k} \bmod p^J$. Then for $J \ge K$, we associate the *p*-adic lifting step of (13) with the matrix polynomial

$$S_{j,k}(x) = 2S_{j-1,n}(x) - S_{j-1,n}(x)A_{j,n}(x)S_{j-1,n}(x)\left(\bmod x^{J/K}\right). \tag{17}$$

The input polynomial matrices are $S_{j-1,n}(x)$ and $A_{j,n}(x)$ for $j=1,2,\cdots,h$. We perform the computations in (17) modulo $x^{J/K}$. The idea here is to apply *e*-reduction to all the entries of the output matrix polynomial followed by a new reduction modulo $x^{J/K}$. The resulting entries are just polynomials with integer coefficients ranging between $1-e$ and $e-1$. This is due to the recursive *e*-reductions and then taking modular reductions again. Note that the output entries are polynomials with coefficients in the range $-\gamma$ to γ for $\gamma \le 2^\beta$ even before applying the *e*-reductions. This shows that the *β*-bit precision is sufficient in all of the computations due to (16). The same argument can be applied to the matrices $S_{j,k}$ for all *j* and $k<n$.

7. Numerical Implementation of Matrix Determinant

In this section we show numerical implementation of the determinant of *n* × *n* matrices based on the triangular factorization *LU*, P_rLU, and P_rLUP_c. Algorithm 1 computes the triangular matrices $\tilde{L}=L+E_L$ and $\tilde{U}=U+E_U$, the permutation matrices $\tilde{P}_r$ and $\tilde{P}_c$, where E_L and E_U are the perturbations of *L* and *U*. In general, we can assume that $\tilde{P}_r=P_r$ and $\tilde{P}_c=P_c$ since the rounding error is small.

Definition 7. *Let*

$$A' = \tilde{P}_r^{-1}A\tilde{P}_c^{-1},\ \tilde{L}\tilde{U} - A' = E', \tag{18}$$

and let e', *a*, $\tilde{l}$, *and* $\tilde{u}$ *denote the maximum absolute value of the entries of the matrices* E', *A*, $\tilde{L}$, *and* $\tilde{U}$. *Also,* E' *is the error matrix of the order of the roundoff in the entries of A.*

Assuming floating point arithmetic with double precision (64-bits) and round to β-bit. Then the upper bound on the magnitude of the relative rounding errors is $\epsilon = 2^{-\beta}$, where ϵ is the machine epsilon. Our goal is to estimate e'.

Theorem 2. ([13]) *For a matrix* $A=(a_{i,j})$, *let* $|A|$ *denote the matrix* $(|a_{i,j}|)$. *Then under* (18), $|E'| \le (|A'|+|\tilde{L}||\tilde{U}|)\epsilon$, *and*

$$e' \le e^+ = (a+n\tilde{l}\tilde{u})\epsilon. \tag{19}$$

From the following matrix norm property $\left(\frac{1}{n}\right)\|A\|_q \le \max_{i,j}|a_{i,j}| \le \|A\|_q$, for $q=1,2,\infty$, we obtain the bound $e' \le \|E'\|_\infty$.

Theorem 3. *Let* a^+ *denotes the maximum absolute value of all entries of the matrices* $A^{(i)}$ *computed by Gaussian elimination, which reduces* A' *to the upper triangular form and let* ϵ *as defined above. Then*

$$e' \le e_+ = \|E'\|_\infty \le n^2 a^+ \epsilon < 1.8\epsilon a n^{2+(\ln^2)/4}. \tag{20}$$

By using the following results, we can estimate the magnitude of the perturbation of $\det A$ and $\det A'$ caused by the perturbation of A'. Here we assume that $P_r = P_c = I, A' = A$ and write,

$$e = e',\ e_d = \det(A+E) - \det A. \tag{21}$$

We use the following facts to estimate e_d of (21) in Lemma 5 below.

Fact 3. $|\det A| \le \|A\|_q^n$, $q = 1,2,\infty$. $\|B\|_q \le \|A\|_q$, $q = 1,\infty$, *if* $|B| \le |A|$; $\|B\|_q \le \|A\|_q$, $q = 1,2,\infty$, *if B is a submatrix of A.* $\|A+E\|_q \le \|A\|_q + ne$, *for* $q = 1,2,\infty$; $\|A+E\|_2^2 \le \|A\|_2^2 + ne$.

Lemma 5. $|e_d| \le (\|A\|_q + ne)^{n-1} n^2 e$, for $q = 1,2,\infty.$

Combining Lemma 5 with the bound (19) enables us to extend Algorithm 1 as follows:

Algorithm 7. (*Bounding determinant.*)

Input: *An* $n\times n$ *real matrix A and a positive* ϵ.

Output: *A pair of real numbers* d_- *and* d_+ *such that* $d_- \le \det A \le d_+$.

Computations:

1) *Apply Algorithm 1 in floating point arithmetic with unit roundoff bounded by* ϵ. *Let* $\tilde{U} = U + E_U$ *denote the computed upper triangular matrix, approximating the factor U of A from* (2) *or* (3).

2) *Compute the upper bound* e_+ *of* (20) *where* $e = e'$.

3) *Substitute e₊ for e and* $\min\left(\|A\|_1, \|A\|_\infty, (\|A\|_1\|A\|_\infty)^{\frac{1}{2}}\right)$ *for* $\|A\|_q$ *in Lemma* 5 *and obtain an upper bound* e_d^+ *on* $|e_d|$.

4) *Output the values* $d_- = \det\tilde{U} - e_d^+$ *and* $d_+ = \det\tilde{U} + e_d^+$.

Example 2. *Let* $A = \begin{pmatrix} 0 & \frac{5}{3} & -\frac{31}{6} \\ \frac{19}{4} & \frac{3}{2} & \frac{16}{5} \\ \frac{17}{5} & \frac{21}{4} & \frac{12}{9} \end{pmatrix} \approx \begin{pmatrix} 0 & 1.6667 & -5.1667 \\ 4.7500 & 1.5000 & 3.2000 \\ 3.4000 & 5.2500 & 1.3333 \end{pmatrix}$. *Using Matlab, Gaussian elimination with complete pivoting gives the matrix U rounded to 4 bits:* $U = \begin{pmatrix} 5.2500 & 1.3333 & 3.4000 \\ 0 & -5.5899 & -1.0794 \\ 0 & 0 & 3.2342 \end{pmatrix}$, $L = \begin{pmatrix} 1 & 0 & 0 \\ 0.3175 & 1 & 0 \\ 0.2857 & -0.5043 & 1 \end{pmatrix}$, $\tilde{U} = U + E_U$, $\tilde{L} = L + E_L$ *where E_U and E_L are the matrices obtained from accumulation of rounding errors. Also* $P_r = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, *and* $P_c = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$. *Then* $E_{A'} = \begin{pmatrix} 0 & 6\times10^{-5} & 6\times10^{-5} \\ 0 & 0 & 0 \\ 0 & 0 & 3\times10^{-5} \end{pmatrix}$ *is a perturbation in A, and* $\|E_{A'}\|_\infty = 1.2\times10^{-4}$. *We now compute the upper bound* e_+ *of* (20). *Therefore,* $e' \le e_+ \le 62.9618$, *where* $a = 5.25$ *is the maximum of* $|a_{i,j}|$ *of A, l = 1 is the maximum of* $|l_{i,j}|$ *of L,* $\epsilon = 2^{-\beta}$, $\beta = 4$, *and* $n = 3$ *is the size of the input matrix* A. *Hence we obtain the following upper bound* e_d^+ *on* $|e_d|$ *by Lemma* 5. *That is,* $|e_d| \le \left(\min\left(\|A\|_1, \|A\|_\infty, (\|A\|_1\|A\|_\infty)^{\frac{1}{2}}\right) + ne_+\right)^{n-1} n^2 e_+ = 2.2347\times10^7$, *where* $\|A\|_1 = 9.7$, $\|A\|_\infty = 9.9833$, *and* $\sqrt{\|A\|_1\|A\|_\infty} = 9.8406$. *Hence,* $e_d^+ = 2.2347\times10^7$ *is an upper bound of* $|e_d|$.

$d_- = \det\tilde{U} - e_d^+ = -2.2347\times10^7$ *and* $d_+ = \det\tilde{U} + e_d^+ = 2.2347\times10^7$. *Since* $\det(A) = -94.91597$, *we have*

$\det\tilde{U} - e_d^+ \le \det(A) \le \tilde{U} + e_d^+$. *On the other hand* $e_d = \det(A + EA) - \det(A) = 0.00123$ *which is less than the upper bound* 2.2347×10^7 *we have computed*.

Acknowledgements

We thank the editor and the referees for their valuable comments.

References

[1] Muir, T. (1960) The Theory of Determinants in Historical Order of Development. Vol. I-IV, Dover, New York.

[2] Fortune, S. (1992) Voronoi Diagrams and Delaunay Triangulations. In: Du, D.A. and Hwang, F.K., Eds., *Computing in Euclidean Geometry*, World Scientific Publishing Co., Singapore City, 193-233.

[3] Brönnimann, H., Emiris, I.Z., Pan, V.Y. and Pion, S. (1997) Computing Exact Geometric Predicates Using Modular Arithmetic. *Proceedings of the* 13*th Annual ACM Symposium on Computational Geometry*, 174-182.

[4] Brönnimann, H. and Yvinec, M. (2000) Efficient Exact Evaluation of Signs of Determinant. *Algorithmica*, **27**, 21-56. http://dx.doi.org/10.1007/s004530010003

[5] Pan, V.Y. and Yu, Y. (2001) Certification of Numerical Computation of the Sign of the Determinant of a Matrix. *Algorithmica*, **30**, 708-724. http://dx.doi.org/10.1007/s00453-001-0032-8

[6] Clarkson, K.L. (1992) Safe and Effective Determinant Evaluation. *Proceedings of the* 33*rd Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 387-395.

[7] Avnaim, F., Boissonnat, J., Devillers, O., Preparata, F.P. and Yvinec, M. (1997) Evaluating Signs of Determinants Using Single-Precision Arithmetic. *Algorithmica*, **17**, 111-132. http://dx.doi.org/10.1007/BF02522822

[8] Pan, V., Yu, Y. and Stewart, C. (1997) Algebraic and Numerical Techniques for Computation of Matrix Determinants. *Computers & Mathematics with Applications*, **34**, 43-70. http://dx.doi.org/10.1016/S0898-1221(97)00097-7

[9] Golub, G.H. and Van Loan, C.F. (1996) Matrix Computations. 3rd Edition, John Hopkins University Press, Baltimore.

[10] Edmonds, J. (1967) Systems of Distinct Representatives and Linear Algebra. *Journal of Research of the National Bureau of Standards*, **71**, 241-245. http://dx.doi.org/10.6028/jres.071B.033

[11] Bareiss, E.H. (1969) Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination. *Mathematics of Computation*, **22**, 565-578.

[12] Bini, D. and Pan, V.Y. (1994) Polynomial and Matrix Computations, Fundamental Algorithms. Vol. 1, Birkhäuser, Boston. http://dx.doi.org/10.1007/978-1-4612-0265-3

[13] Conte, S.D. and de Boor, C. (1980) Elementary Numerical Analysis: An Algorithm Approach. McGraw-Hill, New York.

American Journal of Computational Mathematics, 2014, 4, 474-481
Published Online December 2014 in SciRes. http://www.scirp.org/journal/ajcm
http://dx.doi.org/10.4236/ajcm.2014.45040

Mean Square Heun's Method Convergent for Solving Random Differential Initial Value Problems of First Order

M. A. Sohaly

Department of Mathematics, Faculty of Science, Mansoura University, Mansoura, Egypt
Email: m_stat2000@yahoo.com

Received 23 September 2014; revised 20 October 2014; accepted 12 November 2014

Abstract

This paper deals with the construction of Heun's method of random initial value problems. Sufficient conditions for their mean square convergence are established. Main statistical properties of the approximations processes are computed in several illustrative examples.

Keywords

Stochastic Partial Differential Equations, Mean Square Sense, Second Order Random Variable, Finite Difference Scheme

1. Introduction

Random differential equation (RDE), is an ordinary differential equation (ODE) with random inputs that can model unpredictable real-life behavior of any continuous system and they are important tools in modeling complex phenomena. They arise in many physics and engineering applications such as wave propagation, diffusion through heterogeneous random media. Additional examples can be found in materials science, chemistry, biology, and other areas. However, reaching a solution of these equations in a closed form is not always possible or even easy. Due to the complex nature of RDEs, numerical simulations play an important role in studying this class of DEs. For this reason, few numerical and analytical methods have been developed for simulating RDEs. Random solutions are always studied in terms of their statistical measures.

This paper is interested in studying the following random differential initial value problem (RIVP) of the form:

$$\frac{\mathrm{d}X(t)}{\mathrm{d}t} = f\left(t, X(t)\right), \quad X(t_0) = X_0. \tag{1}$$

Randomness may exist in the initial value or in the differential operator or both. In [1] [2], the authors discussed the general order conditions and a global convergence proof was given for stochastic Runge-Kutta methods applied to stochastic ordinary differential equations (SODEs) of Stratonovich type. In [3] [4], the authors discussed the random Euler method and the conditions for the mean square convergence of this problem. In [5], the authors considered a very simple adaptive algorithm, based on controlling only the drift component of a time step but if the drift was not linearly bounded, then explicit fixed time step approximations, such as the Euler-Maruyama scheme, may fail to be ergodic. Platen, E. [6] discussed discrete time strong and weak approximation methods that were suitable for different applications. [12] discussed the mean square convergent Euler and Runge Kutta methods for random initial value problem. Other numerical methods are discussed in [7]-[12].

In this paper the random Heun's method is used to obtain an approximate solution for Equation (1). This paper is organized as follows. In Section 2, some important preliminaries are discussed. In Section 3, random differential equations are discussed. In Section 4, the convergence of random Heun's method is discussed. Section 5 presents the solution of numerical example of first order random differential equation using random Heun's method showing the convergence of the numerical solutions to the exact ones (if possible). The general conclusions are presented in Section 6.

2. Preliminaries

Definition 1 [13]. Let us consider the properties of a class of real r.v.'s $X_1, X_2, \cdots, X_n$ whose second moments, $E\left\{X_1^2\right\}, E\left\{X_2^2\right\}, \cdots$ are finite. In this case, they are called "second order random variables", (2.r.v's).

The Convergence in Mean Square [13]

A sequence of r.v's $\{X_n\}$ converges in mean square (m.s) to a random variable X if: $\lim_{n\to\infty}\|X_n - X\| = 0$ *i.e.* $X_n \xrightarrow{\text{m.s}} X$ or $\lim_{n\to\infty} X_n = X$ where l.i.m is the limit in mean square sense.

3. Random Initial Value Problem (RIVP)

If we have the random differential equation

$$\dot{X}(t) = f(X(t), t), \quad t \in T = [t_0, t_1], \qquad X(t_0) = X_0 \tag{2}$$

where X_0 is a random variable, and the unknown $X(t)$ as well as the right-hand side $f(X, t)$ are stochastic processes defined on the same probability space (Ω, F, P), are powerful tools to model real problems with uncertainty.

Definition 2 [6] [7].

- Let $g : T \to L_2$ is an m.s. bounded function and let $h > 0$ then The "m.s. modulus of continuity of *g*" is the function

$$W(g, h) = \sup_{|t - t^*| \le h} \left\| g(t) - g(t^*) \right\|, \qquad t, t^* \in T.$$

- The function *g* is said to be m.s. uniformly continuous in *T* if:

$$\lim_{h\to 0} W(g, h) = 0.$$

Note that: (The limit depend on *h* because *g* is defined at every *t* so we can write $W(g, h) = W(h)$).

In the problem (2) that we discusses we find that the convergence of this problem is depend on the right hand side (*i.e.* $f(X(t), t)$) then we want to apply the previous definition on $f(X(t), t)$ hence:

Let $f(X(t), t)$ be defined on *S* × *T* where *S* is bounded set in L_2 Then we say that *f* is "randomly bounded uniformly continuous" in *S*, if $\lim_{h\to 0} W(f(x, .), h) = 0$ (note that: $W(f(X(.), h)) = W(h)$).

A Random Mean Value Theorem for Stochastic Processes

The aim of this section is to establish a relationship between the increment $X(t) - X(t_0)$ of a 2-s.p. and its m.s.

derivative $\dot{X}(\xi)$ for some ξ lying in the interval $[t_0,t]$ for $t>t_0$. This result will be used in the next section to prove the convergence of the random Heun's method discussed.

Lemma (3.2) [6] [7].

Let $Y(t)$ is a 2-s.p., m.s. continuous on interval $T=[t_0,t_1]$. Then, there exists $\xi\in[t_0,t_1]$ such that:

$$\int_{t_0}^{t} Y(s)\,\mathrm{d}s = Y(\xi)(t-t_0), \quad t_0<t<t_1. \tag{3}$$

Theorem (3.3) [6] [7].

Let $X(s)$ be a m.s. differentiable 2-s.p. in $]t_0,t_1[$ and m.s. continuous in $T=[t_0,t_1]$. Then, there exists $\xi\in[t_0,t_1]$ such that:

$$X(t)-X(t_0)=\dot{X}(\xi)(t-t_0).$$

4. The Convergence of Random Heun's Scheme

In this section we are interested in the mean square convergence, in the fixed station sense, of the random Heun's method defined by:

$$X_{n+1}=X_n+\frac{h}{2}\left[f(X_n,t_n)+f(X_n+hf(X_n,t_n),t_{n+1})\right], \quad X(t_0)=X_0, \ n\ge 0 \tag{4}$$

where X_n and $f(X_n,t_n)$ are 2-r.v.'s, $h=t_n-t_{n-1}$, $t_n=t_0+nh$ and $f:S\times T\to L_2$, $S\subset L_2$ satisfies the following conditions:

C1: $f(X,t)$ is randomly bounded uniformly continuous.

C2: $f(X,t)$ satisfies the m.s. Lipschitz condition:

$$\left\|f(x,t)-f(y,t)\right\|\le k(t)\left\|x-y\right\| \quad \text{where}: \int_{t_0}^{t_1} k(t)\le\infty. \tag{5}$$

Note that under hypothesis C1 and C2, we are interested in the m.s. convergence to zero of the error:

$$e_n=X_n-X(t) \tag{6}$$

where $X(t)$ is the theoretical solution 2-s.p. of the problem (2), $t=t_n=t_0+nh$.

Taking into account (2), and Th (3.3), one gets, since from (2) we have at $t=t_\xi$ then: $\dot{X}(t_\xi)=f(x(t_\xi),t_\xi)$. (Note: $\xi\in[t_0,t_1]$ and we can use ξ instead of t_ξ.)

And from Th (3.3) at $t=t_\xi$ then we obtain:

$$X(t_\xi)-X(t_0)=\dot{X}(t)(t_\xi-t_0)\Rightarrow X(t_\xi)-X(t_0)=f(X(t_\xi,t_\xi))(t_\xi-t_0).$$

Note that we deal with the interval (t_n,t_{n+1}) $\ni t_\xi\in(t_n,t_{n+1})$ and hence t_0 was the starting in the problem (2) and here t_n is the starting and since Heun's method deal with solution depend on previous solution and if we have $X(t_n)$ instead of $X(t_0)$ then we can use $X(t_{n+1})$ instead of $X(t_\xi)$ hence the final form of the problem (2) is:

$$X(t_{n+1})=X(t_n)+hf(X(t_\xi,t_\xi)), \ \text{for some } t_\xi\in(t_n,t_{n+1}). \tag{7}$$

Now we have the solution of problem (2) is: $X(t_n)$.

At $t=t_n$ then $X(t_n)=X(t)$ and the solution of Heun's method (4) is: X_n.

Then we can define the error:

$$e_n=X_n-X(t).$$

By (4) and (7) it follows that:

$$e_{n+1}=X_{n+1}-X(t_{n+1})=X_n+\frac{h}{2}\left[f(X_n,t_n)+f(X_n+hf(X_n,t_n),t_{n+1})\right]$$
$$-X(t_n)-\frac{h}{2}f(X(t_\xi),t_\xi)-\frac{h}{2}f(X(t_\xi),t_\xi).$$

Then we obtain:

$$e_{n+1} = X_n - X(t_n) + \frac{h}{2}\left\{ f(X_n, t_n) - f\left(X\left(t_\xi\right), t_\xi\right)\right\} + \frac{h}{2}\left\{ f\left(X_n + hf(X_n, t_n), t_{n+1}\right) - f\left(X\left(t_\xi\right), t_\xi\right)\right\}.$$

By taking the norm for the two sides:

$$\begin{aligned}\|e_{n+1}\| &= \left\| X_n - X(t_n) + \frac{h}{2}\left\{ f(X_n, t_n) - f\left(X\left(t_\xi\right), t_\xi\right)\right\} + \frac{h}{2}\left\{ f\left(X_n + hf(X_n, t_n), t_{n+1}\right) - f\left(X\left(t_\xi\right), t_\xi\right)\right\}\right\| \\ &\le \|X_n - X(t_n)\| + \frac{h}{2}\left\| f\left(X\left(t_\xi\right), t_\xi\right) - f(X_n, t_n)\right\| + \frac{h}{2}\left\| f\left(X_n + hf(X_n, t_n), t_{n+1}\right) - f\left(X\left(t_\xi\right), t_\xi\right)\right\|.\end{aligned} \tag{8}$$

Since:

$$\begin{aligned}&\left\| f\left(X\left(t_\xi\right), t_\xi\right) - f(X_n, t_n)\right\| \\ &= \left\| f\left(X\left(t_\xi\right), t_\xi\right) - f\left(X\left(t_\xi\right), t_n\right) + f\left(X\left(t_\xi\right), t_n\right) + f\left(X(t_n), t_n\right) - f\left(X(t_n), t_n\right) - f(X_n, t_n)\right\| \\ &\le \left\| f\left(X\left(t_\xi\right), t_\xi\right) - f\left(X\left(t_\xi\right), t_n\right)\right\| + \left\| f\left(X\left(t_\xi\right), t_n\right) - f\left(X(t_n), t_n\right)\right\| + \left\| f\left(X(t_n), t_n\right) - f(X_n, t_n)\right\|.\end{aligned} \tag{9}$$

Since the theoretical solution $X(t)$ is m.s. bounded in $[t_0, t_1]$, $\sup\limits_{t_0 \le t \le t_1} \|X(t)\| \le M < \infty$ and under hypothesis C1, C2, we have:

- $\left\| f\left(X\left(t_\xi\right), t_\xi\right) - f\left(X\left(t_\xi\right), t_n\right)\right\| = w(h).$
- $\left\| f\left(X\left(t_\xi\right), t_n\right) - f\left(X(t_n), t_n\right)\right\| \le k(t_n) Mh \qquad (10)$

where $k(t_n)$ is Lipschitz constant (from C2) and:

From Th (3.3) we have $X(t) - X(t_0) = \dot{X}(\xi)(t - t_0)$ and note that the two points are $X\left(t_\xi\right)$ and $X(t_n)$ in (10) then:

$$\left\| X\left(t_\xi\right) - X(t_n)\right\| = \left\|\dot{X}(\xi)\right\| \left| t_\xi - t_n\right| \le Mh$$

where: $\left| t_\xi - t_n\right| = h$ and $M = \sup\limits_{t_0 \le t \le t_1} \left\|\dot{X}(t)\right\|.$

- $\left\| f\left(X(t_n), t_n\right) - f(X_n, t_n)\right\| \le k(t_n)\|X(t_n) - X_n\| = k(t_n)\|e_n\|.$

Then by substituting in (9) we have:

$$\left\| f\left(X\left(t_\xi\right), t_\xi\right) - f(X_n, t_n)\right\| \le w(h) + k(t_n) Mh + k(t_n)\|e_n\|. \tag{11}$$

And another term:

$$\begin{aligned}&\left\| f\left(X_n + hf(X_n, t_n), t_{n+1}\right) - f\left(X\left(t_\xi\right), t_\xi\right)\right\| = \left\| f\left(X\left(t_\xi\right), t_\xi\right) - f\left(X_n + hf(X_n, t_n), t_{n+1}\right)\right\| \\ &= \Big\| f\left(X\left(t_\xi\right), t_\xi\right) - f\left(X\left(t_\xi\right), t_{n+1}\right) + f\left(X\left(t_\xi\right), t_{n+1}\right) + f\left(X(t_n), t_{n+1}\right) \\ &\quad - f\left(X(t_n), t_{n+1}\right) - f\left(X_n + hf(X_n, t_n), t_{n+1}\right)\Big\| \\ &\le \left\| f\left(X\left(t_\xi\right), t_\xi\right) - f\left(X\left(t_\xi\right), t_{n+1}\right)\right\| + \left\| f\left(X\left(t_\xi\right), t_{n+1}\right) - f\left(X(t_n), t_{n+1}\right)\right\| \\ &\quad + \left\| f\left(X(t_n), t_{n+1}\right) - f\left(X_n + hf(X_n, t_n), t_{n+1}\right)\right\| \\ &\le w(h) + k(t_{n+1}) Mh + k(t_{n+1})\left[\|e_n\| - hM\right].\end{aligned}$$

Since:

- $\left\| f\left(X\left(t_\xi\right), t_\xi\right) - f\left(X\left(t_\xi\right), t_{n+1}\right)\right\| \le w(h).$
- $\left\| f\left(X\left(t_\xi\right), t_{n+1}\right) - f\left(X(t_n), t_{n+1}\right)\right\| \le k(t_{n+1}) Mh$

where $k(t_n)$ is Lipschitz constant (from C2) and:

From Th (3.3) we have $X(t)-X(t_0)=\dot{X}(\xi)(t-t_0)$ and note that the two points are $X(t_\xi)$ and $X(t_n)$ in (10) then we have:

$$\left\|X(t_\xi)-X(t_n)\right\|=\left\|\dot{X}(\xi)\right\|\left|t_\xi-t_n\right|\le Mh$$

where $\left|t_\xi-t_n\right|=h$ and $M=\sup_{t_0\le t\le t_1}\left\|\dot{X}(t)\right\|$.

And the last term:

$$\left\|f\left(X(t_n),t_{n+1}\right)-f\left(X_n+hf(X_n,t_n),t_{n+1}\right)\right\|\le k(t_{n+1})\left\|X(t_n)-X_n-hf(X_n,t_n)\right\|$$
$$\le k(t_{n+1})\left[\left\|X(t_n)-X_n\right\|-\left\|hf(X_n,t_n)\right\|\right]=k(t_{n+1})\left[\left\|e_n\right\|-hM\right].$$

Then by substituting in (8) we have:

$$\begin{aligned}
\|e_{n+1}\| &\le \|e_n\|+\frac{h}{2}\left[w(h)+k(t_n)Mh+k(t_n)\|e_n\|\right]+\frac{h}{2}\left[w(h)+k(t_{n+1})Mh+k(t_{n+1})\left[\|e_n\|-hM\right]\right]\\
&=\|e_n\|\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\frac{h}{2}\left[2w(h)+hk(t_n)M+hk(t_{n+1})M-K(t_{n+1})hM\right]\\
&\le\left\{\|e_{n-1}\|\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right]\right\}\\
&\quad\times\left\{\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)\right\}+\frac{h}{2}\left[2w(h)+hk(t_n)M+hk(t_{n+1})M-K(t_{n+1})hM\right]\\
&=\|e_{n-1}\|\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right]\\
&\quad\times\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\frac{h}{2}\left[2w(h)+hk(t_n)M+hk(t_{n+1})M-K(t_{n+1})hM\right]\\
&\le\left[\left\{\|e_{n-2}\|\left(1+\frac{h}{2}k(t_{n-2})+k(t_{n-1})\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-2})M+hk(t_{n-1})M-K(t_{n-1})hM\right]\right\}\right.\\
&\quad\left.\times\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right]\right]\\
&\quad\times\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\frac{h}{2}\left[2w(h)+hk(t_n)M+hk(t_{n+1})M-K(t_{n+1})hM\right]\\
&=\|e_{n-2}\|\left(1+\frac{h}{2}k(t_{n-2})+k(t_{n-1})\right)\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)\\
&\quad+\frac{h}{2}\left[2w(h)+hk(t_{n-2})M+hk(t_{n-1})M-K(t_{n-1})hM\right]\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)\\
&\quad+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right]\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)\\
&\quad+\frac{h}{2}\left[2w(h)+hk(t_n)M+hk(t_{n+1})M-K(t_{n+1})hM\right]\\
&\le\left[\left[\left\{\|e_{n-3}\|\left(1+\frac{h}{2}k(t_{n-3})+k(t_{n-2})\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-3})M+hk(t_{n-2})M-K(t_{n-2})hM\right]\right\}\right.\right.\\
&\quad\left.\times\left(1+\frac{h}{2}k(t_{n-2})+k(t_{n-1})\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-2})M+hk(t_{n-1})M-K(t_{n-1})hM\right]\right]\\
&\quad\left.\times\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right]\right]
\end{aligned}$$

$$\times\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\frac{h}{2}\left[2w(h)+hk(t_n)M+hk(t_{n+1})M-K(t_{n+1})hM\right]$$

$$=\|e_{n-3}\|\left(1+\frac{h}{2}k(t_{n-3})+k(t_{n-2})\right)\left(1+\frac{h}{2}k(t_{n-2})+k(t_{n-1})\right)\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)$$

$$+\frac{h}{2}\left[2w(h)+hk(t_{n-2})M+hk(t_{n-1})M-K(t_{n-1})hM\right]\left(1+\frac{h}{2}k(t_{n-2})+k(t_{n-1})\right)$$

$$\times\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right]$$

$$\times\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\frac{h}{2}\left[2w(h)+hk(t_n)M+hk(t_{n+1})M-K(t_{n+1})hM\right]$$

$$\times\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right].$$

Finally, we have:

$$\|e_{n+1}\|\le\|e_0\|\left(1+\frac{h}{2}k(t_0)+k\left(t_{n-(n-1)}\right)\right)\left(1+\frac{h}{2}k\left(t_{n-(n-1)}\right)+k\left(t_{n-(n-2)}\right)\right)\left(1+\frac{h}{2}k\left(t_{n-(n-2)}\right)+k\left(t_{n-(n-3)}\right)\right)\cdots$$

$$\times\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-2})M+hk(t_{n-1})M-K(t_{n-1})hM\right]$$

$$\times\left(1+\frac{h}{2}k\left(t_{n-(n-1)}\right)+k\left(t_{n-(n-2)}\right)\right)\left(1+\frac{h}{2}k\left(t_{n-(n-2)}\right)+k\left(t_{n-(n-3)}\right)\right)\cdots\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)$$

$$\times\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)\left(1+\frac{h}{2}k(t_{n-2})+k(t_{n-1})\right)\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)$$

$$+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right]\left(1+\frac{h}{2}k\left(t_{n-(n-2)}\right)+k\left(t_{n-(n-3)}\right)\right)\cdots\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)$$

$$\times\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)\left(1+\frac{h}{2}k(t_{n-2})+k(t_{n-1})\right)\left(1+\frac{h}{2}k(t_{n-1})+k(t_n)\right)\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\cdots$$

$$+\frac{h}{2}\left[2w(h)+hk(t_n)M+hk(t_{n+1})M-K(t_{n+1})hM\right]\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)$$

$$+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right]$$

$$=\|e_0\|\prod_{i=0}^{n}\left(1+\frac{h}{2}k(t_{n-i})+k\left(t_{n-(i-1)}\right)\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-2})M+hk(t_{n-1})M-K(t_{n-1})hM\right]$$

$$\times\prod_{i=0}^{n-1}\left(1+\frac{h}{2}k(t_{n-i})+k\left(t_{n-(i-1)}\right)\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right]$$

$$\times\prod_{i=0}^{n-2}\left(1+\frac{h}{2}k(t_{n-i})+k\left(t_{n-(i-1)}\right)\right)+\cdots+\frac{h}{2}\left[2w(h)+hk(t_n)M+hk(t_{n+1})M-K(t_{n+1})hM\right]$$

$$\times\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-1})M+hk(t_n)M-K(t_n)hM\right]$$

$$=\|e_0\|\prod_{i=0}^{n}\left(1+\frac{h}{2}k(t_{n-i})+k\left(t_{n-(i-1)}\right)\right)+\frac{h}{2}\left[2w(h)+hk(t_{n-2})M+hk(t_{n-1})M-K(t_{n-1})hM\right]$$

$$\times\left[1+\prod_{i=0}^{n-1}\left(1+\frac{h}{2}k(t_{n-i})+k\left(t_{n-(i-1)}\right)\right)+\prod_{i=0}^{n-2}\left(1+\frac{h}{2}k(t_{n-i})+k\left(t_{n-(i-1)}\right)\right)+\cdots+\left(1+\frac{h}{2}k(t_n)+k(t_{n+1})\right)\right].$$

Taking into account that $e_0=0$ where $e_0=X_0-X(t_0)=0$ and by taking the limit as:

$$h \to 0 \quad \text{then we have:} \quad \lim_{h\to 0}\|e_{n+1}\| \to 0$$

i.e. $\{e_n\}$ converge in m.s. to zero as: $h \to 0$ hence $X_n \xrightarrow{\text{m.s}} X(t_n) = X(t)$.

5. Case Study

Example: Solve the problem

$$\frac{\mathrm{d}N}{\mathrm{d}t} = \alpha N, \quad N(0) = 1000, \quad t \in [0,1], \quad \alpha \sim \text{random variable.}$$

The theoretical solution is:

$$N(t) = 1000\mathrm{e}^{0.5\alpha}.$$

The approximations:

$\alpha \sim \text{Binomial}(5,0.2)$					
h (step size)	$E[y]$ for the exact solution	$E[y_n]$ for Heun's method	Error on Heun's	Error on Euler [12]	Error on Runge Kutta [12]
0.25	1318.179242	1306.250000	11.929242	68.179242	156.820758
0.125	1140.431227	1128.515625	1.368727	15.431227	97.068773
0.025	1025.572765	1025.562500	0.010265	0.572765	21.927235
0.0025	1002.505635	1002.505625	0.000010	0.005635	2.244365
$\alpha \sim \text{Erlang}(0.5,2)$					
0.25	1306.122449	1296.875000	9.247449	56.122449	131.377551
0.125	1137.777778	1136.718750	1.059028	12.777778	80.972222
0.025	1025.572765	1025.468750	0.007936	0.476686	18.273314
0.0025	1002.505635	1002.504688	0.000007	0.004695	1.870305
$\alpha \sim \text{Poisson}(2)$					
0.25	1764.823762	1687.500000	77.323762	264.823762	485.176238
0.125	1305.122500	1296.875000	8.247500	55.122500	319.877500
0.025	1051.933860	1051.875000	0.058860	1.933860	73.066140
0.0025	1005.018808	1005.018750	0.000058	0.018808	7.481192

In this results showed in the table we have the Heun's method gave better approximation as: $h \to 0$ than Euler and Rung-Kutta [12] for solving random variable, ınitial value problems.

6. Conclusion

The initially valued first order random differential equations can be solved numerically using the random Heun's methods in mean square sense. The convergence of the presented numerical techniques has been proven in mean square sense. The results of the paper have been illustrated through an example.

References

[1] Burrage, K. and Burrage, P.M. (1996) High Strong Order Explicit Runge-Kutta Methods for Stochastic Ordinary Differential Equations. *Applied Numerical Mathematics*, **22**, 81-101. http://dx.doi.org/10.1016/S0168-9274(96)00027-X

[2] Burrage, K. and Burrage, P.M. (1998) General Order Conditions for Stochastic Runge-Kutta Methods for Both Commuting and Non-Commuting Stochastic Ordinary Equations. *Applied Numerical Mathematics*, **28**, 161-177.

http://dx.doi.org/10.1016/S0168-9274(98)00042-7

[3] Cortes, J.C., Jodar, L. and Villafuerte, L. (2007) Numerical Solution of Random Differential Equations, a Mean Square Approach. *Mathematical and Computer Modeling*, **45**, 757-765. http://dx.doi.org/10.1016/j.mcm.2006.07.017

[4] Cortes, J.C., Jodar, L. and Villafuerte, L. (2006) A Random Euler Method for Solving Differential Equations with Uncertainties. Progress in Industrial Mathematics at ECMI.

[5] Lamba, H., Mattingly, J.C. and Stuart, A. (2007) An Adaptive Euler-Maruyama Scheme for SDEs, Convergence and Stability. *IMA Journal of Numerical Analysis*, **27**, 479-506. http://dx.doi.org/10.1093/imanum/drl032

[6] Platen, E. (1999) An Introduction to Numerical Methods for Stochastic Differential Equations. *Acta Numerica*, **8**, 197-246. http://dx.doi.org/10.1017/S0962492900002920

[7] Higham, D.J. (2001) An Algorithmic Introduction to Numerical Simulation of SDE. *SIAM Review*, **43**, 525-546. http://dx.doi.org/10.1137/S0036144500378302

[8] Talay, D. and Tubaro, L. (1990) Expansion of the Global Error for Numerical Schemes Solving Stochastic Differential Equation. *Stochastic Analysis and Applications*, **8**, 483-509. http://dx.doi.org/10.1080/07362999008809220

[9] Burrage, P.M. (1999) Numerical Methods for SDE. Ph.D. Thesis, University of Queensland, Brisbane.

[10] Kloeden, P.E., Platen, E. and Schurz, H. (1997) Numerical Solution of SDE through Computer Experiments. 2nd Edition, Springer, Berlin.

[11] El-Tawil, M.A. (2005) The Approximate Solutions of Some Stochastic Differential Equations Using Transformation. *Applied Mathematics and Computation*, **164**, 167-178. http://dx.doi.org/10.1016/j.amc.2004.04.062

[12] El-Tawil, M.A. and Sohaly, M.A. (2011) Mean Square Numerical Methods for Initial Value Random Differential Equations. *Open Journal of Discrete Mathematics*, **1**, 66-84. http://dx.doi.org/10.4236/ojdm.2011.12009

[13] Kloeden, P.E. and Platen, E. (1999) Numerical Solution of Stochastic Differential Equations. Springer, Berlin.