Al-Mustansiriyah Journal of Science
Vol. 27, No 1, 2016

journal homepage:www.mjs-mu.com

Visualization of The Motion of Singularities for The Solutions of Nonlinear Partial Differential Equation via Computing Their Locations in Complex Plane: Algorithmic Approach

Inaam A. Malloki * , Hussein Jameel Mutashar **
* Department of Mathematics, College of Science, University of Mustansiriyah
** Department of Bio-Medical Engineering, University of Technology

A r t i c l e i n f o

Received 23/4/2015
Accepted 18/5/2015

ABSTRACT

Singularities often occur in solutions of partial differential equations, In this paper, two new algorithms have been presented here, by which one can visualize the motion of poles of solution of partial differential equation, through detecting the position of the poles as time varies. We start by solving the given partial differential equation by spectral method, then continue this solution into complex plane through Padè approximation, and then compute the singularity of the resulting solution. Subsequently, we apply both algorithms to some Cauchy problems of Constantin-Lax-Majda and Burgers and Sharma – Tasso – Olver equations.

الخلاصة

النقاط المنفردة غالبا ما تتكون في حلول المعادلات التفاضلية الجزئية، في هذا البحث، قدمنا خوارزميات جديدة، بواسطتهما نستطيع ان نصور حركة اقطاب حل المعادلة من خلال تحديد موقع تلك الاقطاب كلما تغيير الزمن، نبدأ بحل المعادلة بالطريقة الطيفية، ثم نوسع الحل الى الفضاء العقدي من خلال تقريبات باديبه واخيرا حساب اقطاب هذا الحل، طبقنا هذه الخوارزميات على مسالة كوشي لمعادلة قسطنطين – لاكس – ماجدة ومعادلة بركرس ومعادلة شارما – تاسو – اولفر .

INTRODUCTION

Many nonlinear differential equations feature local solutions, which develop singularities. The occurrence of singularities in mathematical models often has a physical interpretation (e.g., ignition in combustion, focusing in optics, cusps in free-surface flows, etc.) [1]. Proving all time regularity or occurring of singularity for solution of nonlinear partial differential equation is often a mathematical challenge [2]. It becomes important to determine the location and nature of the singularities, as well as their dependence on the initial data [1]. This paper concerns with Burgers hierarchy, especially for the first and second levels which correspond to Burgers and Sharma-Tasso-Olver (STO) equation respectively; our strategy is to visualize the dynamic of poles by computing the location of poles numerically, our scheme in this strategy differ from the other techniques that found in literature. There are several works for computing the nature and location of the singularities of solutions of partial differential equations numerically. Sulem et al. in 1983 [2] divided the methods which have been used for the detection and decipherment of singularities into two broad classes: those based upon expansion the solution in a Taylor series using some initial data; and those based upon discretizing the governing equation. Hussaini et al. in 1988 [3] introduced shock-capturing and shock-fitting techniques, also there exist various adaptive techniques based on scale invariance and moving grid, suggested by Berger in 1988 [4] and Budd et.al in 1996 [5] respectively.

Wiedeman in 2003 [6] describes a technique, for computing the singularities of solution to partial differential equation, that is to detect the complex singularities. Wiedeman utilized the Padè approximation to gain information on the nature and position of singularities, of the solution of the partial differential equation, to locate the pole he applies a numerical maximization search to the objective function $f(z) = \log|u(z)|$. where $u(z)$ is the Padè approximation of the solution, and to compute the order of pole, he used the principle argument of computing poles order, after treated it numerically.

Trefethen and Tee in 2006 [7], have presented an adaptive spectral method to problems whose solution have a singularities in complex domain close to [-1,1]. Here our strategy introduced by two algorithms, by them, one can trace complex singularities of the solution of P.D.E. in complex domain, in order to visualize the dynamics of singularities as time varies. When the initial data for the P.D.E. are analytic (and periodic), it is possible to trace the temporal behavior of the width of the analyticity strip [8], [2], in order to obtain evidence for motion of singularities or blowup. This takes advantage of the position of complex singularities in discrete space. In our algorithms, there are three steps the successive steps are:

A) Find the solution by spectral method in the truncated series form.

B) Continue the solution into complex plane via Padè approximation.
C) Compute the singularities of the approximated solution.

In next sections, we describe these tools briefly, then we use these algorithms for some examples.

Spectral Methods

Spectral methods are a class of spatial discretizations for differential equations; they provide a way of translating an equation expressed in continuous space and time into a discrete equation, which can be solve numerically. They provide very low error approximations, and they are usually global methods. Due to this fact, spectral methods usually have a very high order of approximation. They may use in its procedure the Galerkin and Collocation methods.

Truncated Fourier Series [3], and [9]

Fourier series are particularly suited for the discretization of periodic function $u(x) = u(x + L)$. For such a periodic domain with periodicity L , and the Fourier functions are

$u_N(x) = \sum_{|k|\le K} c_k e^{ik\alpha x} =$
$\sum_{|k|\le K} c_k \Phi_k \ , \ with\ c_k \epsilon\ \mathbb{C}$ (1)

where the coefficients c_k are the complex Fourier coefficients, for the Fourier mode $\Phi_k(x) = \exp(i\alpha kx)$. Note that the summation limits are sometimes also denoted as $|k| \le N/2$ with $N = 2K+1$, where N is number of grid points . Additionally, a Fourier-transformed quantity is often denoted by a hat, $\hat{u}_k = c_k$. The transformation from the space of the discrete representation of u_N (physical space) to the space of the Fourier components c_k. (Spectral space) is called the (forward) discrete Fourier transform $\mathcal{F}(u_N)$. Correspondingly, the reverse transform is the inverse Fourier transform $\mathcal{F}^{-1}(c_k)$. An efficient way to compute this is via the fast Fourier transforms (FFT), [10]. The relation between physical and spectral space is shown in straightforward way to be

$u_N(x_j) = \sum_{|k|\le K} c_k \Phi_k(x_j), \qquad c_k =$
$N \sum_{j=1}^{N} u_N(x_j) \Phi_{-k}$. (2)

These relations can be used to transform between the physical and spectral space, and are called a "discrete Fourier transformation", implemented via a MATLAB code "FFT" and "IFFT".

Chebyshev Polynomials [3] , and [9]

Fourier series are only a good choice for periodic function. For problems with non-periodic boundary conditions, ansatz functions based on orthogonal polynomials are preferred. One popular choice are the Chebyshev polynomials, defined on a domain $|x| \le 1$ as

$T_k(x) = cos(k \cos^{-1} x), \quad k =$
$0, 1, 2, \ldots.$ (3)

A function $u(x)$ is approximated via a finite series of Chebyshev polynomials as:

$u_N(x) =$
$\sum_{k=0}^{N} a_k \mathrm{T}_k(x)$

with the a_k being the Chebyshev coefficients. A common distribution of points in particular for Chebyshev polynomials are the "Gauss-Lobatto" points.

$x_j = \cos\left(\frac{\pi j}{N}\right), \quad j =$
$0,1, \ldots, N$ (5)

The $N + 1$ points x_j correspond to the locations of the extremes of $T_N = \pm 1$.

The transformation between the physical space u_N and spectral (Chebyshev) space a_k is done via the so-called Chebyshev transform. Since the Chebyshev polynomials are essentially cosine functions on a transformed coordinate, as in Fourier series the fast transform based on the FFT can be used here.

If a collocation method on the Gauss-Lobatto grid (5) is employed, the derivative of a discretized function u_N can be written as a matrix multiplication,

$u'_N(x_i) =$
$\sum_{j=0}^{N} D_{ij} u_N(x_j)$. (6)

For the P^{th} order of derivative there is a general form:

$u_N^{(p)}(x_i) =$
$\sum_{j=0}^{N} D_{ij}^{(p)} u_N(x_j)$. (7)

Or in the vector form:

$u_N^{(p)} =$
$D^{(p)}.u_N$ (8)

Convolution Sum of Nonlinear Terms [3], and [11]

The fairly complex summation in the nonlinear term resembling a convolution is a consequence of the nonlinearity. To do this convolution consider the product $w_j = w(x_j),\ j = 1, \ldots, N$ of two grid functions $u_j = u(x_j)$, $and\ v_j = u(x_j)$ in physical space

$w_j =$
$u_j . v_j$, (9)

we proceed by transforming to spectral space with using

$u_j = \sum_{|k|\le K} \hat{u}_k e^{ikx_j}, \qquad \hat{u}_k =$
$\frac{1}{N} \sum_{j=1}^{N} u_j \ e^{-ikx_j}$. (10)

And the discrete form of the orthogonality relation

$\frac{1}{N} \sum_{j=1}^{N} e^{ikx_j} \ e^{imx_j} =$
$\delta_{k,-m+n.N}$, (11)

in the last equation, $n \in \mathbb{Z}$ corresponds to an arbitrary multiple of N which is closely related to aliasing errors (see further about aliasing errors in [9]). Finally one obtains:

$\widehat{w}_l = \frac{1}{N} \sum_{|k|\le K} \sum_{|m|\le K} \hat{u}_k \hat{v}_k \sum_{j=1}^{N} e^{ikx_j} \ e^{i(m-l)x_j} =$
$\sum_{\substack{k=-K \\ l=k+m \\ |m|\le K}}^{K} \hat{u}_k \hat{v}_m$. (12)

i.e. the well-known result that a multiplication in physical space corresponds to a convolution in spectral space. In other words, the fairly expensive evaluation of a convolution in spectral space, equation (12), is equivalent to the direct evaluation of a point-wise multiplication in physical space, and it is done via the following steps:

1. Transform $\hat{u}_k, \hat{v}_m$ to physical space using FFT: $u_j = \mathcal{F}^{-1}(\hat{u}_k)$, $v_j = \mathcal{F}^{-1}(\hat{v}_k)$ (13)

2. Multiplication in physical space: $w_j = u_j . v_j$
3. Transform back to spectral space: $\widehat{w}_l = \mathcal{F}(w_j)$;
This final results now reads as follows:

$$\widehat{w}_l = \sum_{\substack{k=-K \\ l=k+m+n.N \\ |m|\leq K}}^{K} \widehat{u}_k \widehat{v}_m \ . \quad (13)$$

Such an evaluation of the spectral convolution in physical space is usually termed ***pseudo-spectral*** evaluation of the nonlinear terms.

Padè Approximation

The Padè approximant of a function expanded by given power series with the radius of convergence r is a rational function of numerator degree n and denominator degree m whose power series agrees with the given one up to degree $n + m$ inclusively. A collection of Padè approximants formed by using suitable sets of values of n and m often provides a means of obtaining information about the function outside its circle of convergence, and of more rapidly evaluating the function within its circle of convergence. Padè approximation is ill-posed conditions since it is related to analytic continuation, since the aim is typically to gain information about a function in a region of the complex plane based on information at a single point [12]. Padè approximation can be defined as follows: Let $f(x)$ denote a function having a power series expression

$$f(x) = \sum_{k=0}^{\infty} c_k x^k \quad (14)$$

for any positive integers $n \; and \; m \; with \; n \geq m$, the Padè approximation of order (n, m) of the series $f(x)$, denoted by "$[n, m]$ Padè approximation" is defined to be a rational function $R_{n,m}(x)$ expressed in a fractional form:

$$R_{n,m}(x) = \frac{P_n(x)}{Q_m(x)} = \frac{p_0 + px + \cdots + p_n x^n}{q_0 + q_1 x + \cdots + q_m x^m} \quad (15)$$

With the property that:

$$f(x) - R_{n,m}(x) = O(x^{n+m+1}) \quad (16)$$

The unknown coefficients $p_0 \ldots p_n \; and \; q_0 \ldots q_m \; of \; R_{n,m}(x)$ can be determined from the above condition.

Proposition 1 [13]

Padè approximation of type $[n, m]$ for some $n, m \in \mathbb{N}$, always exist.

Defects (Froissart Doublets) of Padè Approximation

Before going to define and detect the defect or calculate the Padè approximation, we have to distinguish between the coefficient problem and value problem. *Coefficient problem*, is the problem of calculating the coefficients $p_0 , p_1, \ldots p_n$ and $q_0 , q_1 \ldots q_m$ of $R_{n,m}(z)$, and then evaluate the approximants at given z. *Value problem* is the problem, if point-wise evaluation of whole sequence is required at pre-specific values of z [12].

By defect we mean: there is an extraneous pole and a nearby zero. More precisely, in any particular calculations the paired roots in the numerators and the denominators will non-rigorously equal. This phenomenon of "pairing" of such zeros and poles got the name of *Froissart phenomenon* and the pairs are known as "*Froissart doublets"* [14], Froissart doublets and spurious poles are a synonymous names of defects. There are real difficulties in the numerical detection of defects [15], while, they easily recognized by their transient nature. They tend to appear and disappear as one looks at one approximant and then the next.

Methods of Computing Padè Approximants

There are many methods to compute Padè approximants. Some of them are implementing in computer algebra system such as *Maple* and *Mathematica* and their built-in utilities are frequently used in applied problems [16]. Some of them depend on the acceleration of convergence for sequences or series, such as epsilon algorithm [17], eata algorithm [12] and roh algorithm [18]. Epsilon algorithm is suggested by baker in [12] as the best algorithm for value problem since it canceled all (or reduce the number) of defects. For coefficients problem, the methods that used to compute the Padè approximation have a common problem. Basically due to the fact that Padè approximants has as many singularities as the denominator has roots. Only exceptional methods do not produce unwanted singularities, even in exact arithmetic. However, there have been developments in algorithms for rational interpolation that are claimed to overcome this issue. There is an efficient method singular value decomposition (SVD algorithm) suggested by Trefethen in 2013 [19]. In this paper we will use both of epsilon algorithm and SVD-algorithm.

Procedure of Visualize the Motion of Singularities for The Partial Differential Equation's Solution

Our scheme starts by solving the given PDE. By spectral method, with efficient method for time integration with respect to time variable (Euler or Runge-Kutta Methods), this method give us an approximated solution in series form; "truncated Fourier series". As soon as the spectral solution is available, we continue the solution analytically into complex plane by using Padè approximation, since the truncated Fourier series is an entire function, so that; it has no singularities in its domain; while Padè approximation may have some. The analytic continuation is done here numerically at each time step. The final step of our scheme is to locate the singular point (points) in the complex plane at each time step. By plotting these locations in the complex plane, one can see the behavior and motion of the singularities as time varies. All of the computation tools, spectral methods and Padè approximations and algorithms that compute the numerical analytic continuation, are combined, and used here to reach our goal. Next subsections give brief details that explain how these computation tools have been combined, in our work, we introduce our scheme in algorithmic form by two

independent algorithms; ***algorithm A*** and ***algorithm B***. Algorithm A treat the Padè approximation as value problem while algorithm B treated it as coefficients problem. We writes the procedure of our scheme which devoted to visualize the motion of singularities for the solution of nonlinear PDE in algorithmic form, algorithm A and algorithm B are presented here.

Algorithm A

The following steps (step1-step3) describe the algorithm that picturing the motion of one pole along its possible domain D, by compute the $[\frac{K}{2},\frac{K}{2}]$Padè approximation at each value of all $(N+1)\times(N+1)$ points in the sub-region $D_i \subset D$, where N is the number of grid point and $N=2k$, and $i=1,\dots,N$ and chose the candidate point to be the wanted pole, the point which make Padè approximation rich its maximum. This means that this algorithm treated Padè approximation as a value problem.

Step 1:

Solve the given PDE (initial value problem) by pseudo-spectral method, for $n, j=0,1,\dots,N,$ and $x_j \in I=[0,2\pi]$ and $t^n \in [T^{intial},T^{end}]$

$u(x_j,t^n)=$

$\sum_{k=-K}^{K}\hat{u}_k(t^n)\Phi_k(x_j)$. (17)

Step 2:

- Initialize the value of (n) to be zero. Find $x^* \in I$ at which the solution (17) has its maximum value, and for the time iteration (n) do the following to the solution (17) in step 1.
- While $n \le N$ do
- Continue the (17) to complex plane through Padè approximation and formulate it as in the approximation

$u(z,t^n) \cong R_{\frac{K}{2},\frac{K}{2}}(w)+S_{\frac{K}{2},\frac{K}{2}}(v)-\hat{u}_0$, $z \in \mathcal{D}$ (18)

where

$$\mathcal{D}=\begin{cases}\mathcal{D}_{x^*}\times I\,, & \mathcal{D}_{x^*}\ is\ a\ neighbors\ of\ x^*\,, \quad if\ n=0\\ \mathcal{D}_{z^*_{n-1}}\,, & \mathcal{D}_{z^*_{n-1}}\ is\ a\ neighbors\ of\ z^*_{n-1}\,, \quad if\ n>0\end{cases}$$

- Compute the approximation (18) using epsilon algorithm for each $z \in \mathcal{D}$, and then find z_n^*, at which $|u(z,t^n)|$ rich its maximum value.
- Plot z_n^* in complex plane (snapshot of points), and assign it as a function of the time t^n, i.e. $z_n^*=z^*(t^n)$.
- Increase n by 1, and return.

Step 3:

Collect all snapshots of z^* in one graph, this trace all complex singularities in the given time interval $[T^{intial},T^{end}]$, which appear as one singularity moving as time varies from T^0 to T^1, and then to T^2, and so on until T^{end}.

Remark 1

In the step 2 of algorithm A, the radius of the neighbors $'r'$ is consider as a parameter for this algorithm, and the rectangle region D, is divide as mesh grid $N\times N$ points.

Algorithm B

The following steps (step1-step3) describe the algorithm that picturing the motion of one pole along its possible domain D, by computing the $[n,m]$ Padè approximation through SVD-algorithm, which gives all coefficients of the polynomials in nominator and denominator of Padè approximation, and then compute the roots of both polynomials excluding the defect. The roots of denominator are the poles at each time step. This means that this algorithm treated Padè approximation as a coefficients problem.

Step 1:

Solve the given PDE (initial value problem) by pseudo-spectral method, and write the solution in the form (19), Where $n=0,1,\dots,N$ and $t^n \in [T^{intial},T^{end}]$

$u(x,t^n)=$

$\sum_{k=-K}^{K}\hat{u}_k(t^n)\Phi_k(x)$. (19)

Step 2:

Initialize the value of (n) to be zero, and for the time iteration (n) do the following to the solution (19) in step 1,

- While $n \le N$ do
- Continue the (19) to complex plane through Padè approximation and formulate it as

$u(x,t^n)=\sum_{m=0}^{2K+1}\hat{u}_{k(m)}(t^n)\phi(x)\,,\ where\ k(m)=-K+m$ (20)

$u(z,t^n) \cong R_{p,q}(z)\ ,where \quad z\in\mathbb{C},\ p,q\in\mathbb{Z}^+,$ (21)

- Compute the approximation (21) in previous step using SDV-algorithm, this algorithm give us the "q" roots of denominator of rational Padè approximation $R_{p,q}(z)$,

if we chose $q=2$ in (21), then we have just two poles z^* and its conjugates $\bar{z}^*$.

- Plot z^*, and $\bar{z}^*$in complex plane (snapshot of points), and assign them as a function of the current t^n, i.e. $z^*=z^*(t^n)$, and $\overline{z}^*=\bar{z}^*(t^n)$. Here z^*, and $\bar{z}^*$ may be z_q^*, and $\bar{z}_q^*$, $q=1,2,\dots,m$.
- Increase n by 1, and return.

Step 3:

Collect all snapshots of z^* and $\bar{z}^*$ in one graph, this trace all complex singularities in the given time interval $[T^{intial},T^{end}]$, which appear as two singularities moving as time varies from T^0 to T^1, and then to T^2, and so on until T^{end}.

Application of Algorithm A and Algorithm B.

The details of our implementation are as follows: All of spectral methods, time integral, and epsilon algorithm, are coded in MATLAB 2013. The time integral is done via Runge-Kutta Method, with adaptive time step, in this work we set $N=2^5$, in both of algorithm A and algorithm B, the convolution sum in nonlinear terms is computed with MATLAB function (conv.), in all of our applications, we used the default tolerance parameters.

Application I (Constantin-Lax-Majda equation)

Consider the problem

$u_t + H(u)u_x - \nu u_{xx} = 0\ ,\ x \in (-\pi,\pi)\ ,\quad and\ \nu \in \mathbb{R},\quad t > 0 \quad (22)$

Initialed by the condition: $u(x,0) = 1 + \nu \frac{(1-e^{-2})}{1+e^{-2}-2e^{-1}cosx}$.

Where the operator H is the Hilbert transformation,

$H(u(x,t)) = \frac{1}{\pi} F \int_{-\infty}^{\infty} \frac{u(\gamma)}{\gamma - x} d\gamma \quad (23)$

With the property $H(e^{ikx}) = i\ sign(k)\ e^{ikx}$

the exact solution for this problem is [20]:

$u(x,t) = 1 + \nu \frac{(1-e^{2t-2})}{1+e^{2t-2}-2e^{t-1}cosx} \quad (24)$

which periodic solution of period 2π, for this example we use $\nu = 0.01$.

If x in (24) is complex then all singularities of (24) are simple pole, located at

$z = \pm i(1-t) + 2n\pi \quad (25)$

(25) is computed with aid of *maple*. Now we have to solve (22), due to the periodicity of initial condition, a Fourier Galerkin scheme shall be used for the spatial discretization; the approximated solution is:

$u(x_j, t^n) = \sum_{k=-K}^{K} \hat{u}_k(t^n) e^{ikx_j}. \quad (26a)$

or in a vector form:

$u_n(x,t^n) = \sum_{k=-K}^{K} \hat{u}_k(t^n) e^{ikx}\ . \quad (26b)$

Due to the property of Hilbert transformation [21]

$H(u_n(x,t^n)) = \sum_{k=-K}^{K} i\ sgn(k)\ \hat{u}_k(t^n) e^{ikx},\ sgn(k) = 0\ if\ k = 0. \quad (27)$

In the Fourier space equation (22) become

$\hat{u}_t - sgn\ (k) k \hat{w} + \nu k^2 \hat{u} = 0 \quad (28)$

where $\hat{w}$ is the convolution term.

In the first step of our two algorithms is to integrate (28) to get the solution in the forms of (26). The resulting solution, is continued in to the complex plane, in step 2 of both algorithms A and B, by convert it to Padè rational form and then using epsilon algorithm and SDV algorithm in algorithm A and B algorithm respectively as describe above. and then compute the locations of singularities as describe in algorithm A and algorithm B, in the following figures (Figure 1-2), we illustrate how our algorithms can detect the motion of poles as a compare with the exact motion of poles.

The first row of figure (1) shows $|u(x,t)|$, for varies ($t = 0.1, 0.2, 0.5$ and 0.9), with $\nu = 0.01$ and the second row shows the computed value of $|u(z,t)|$, For the same values of t that in the first row in complex domain $z = x + iy$, $x \in (-\pi, \pi)$, $y \in (0,1)$. In this domain, there exist one pole $z = (1-t)i$ and it's conjugate $\bar{z}$. It is easily seen that, the motion of maximum value of u, along the positive imaginary part toward real axis. The same motion is detected in the negative imaginary part (not shown here).

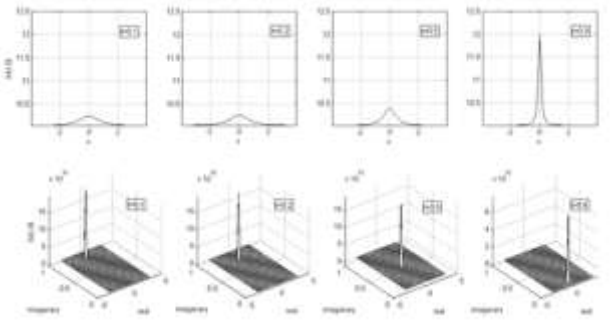

Figure 1: For various times $\mathbf{t = 0.1, 0.2, 0.5, 0.9}$ The first row shows $|\mathbf{u(x,t)}|$, and the second row shows the computed value of $|\mathbf{u(z,t)}|$, in complex domain $\boldsymbol{D = (-\pi, \pi) \times (0,1)}$ for the same values of t that in the first row.

Figure (2) shows the locations of the pole ($z = (1-t)i$). Explains algorithm A and compares the pole motion with the exact one. In this figure, the exact pole and the approximated poles by algorithm A, are aligned from left to right, and then collected in last graph of the figure; In all cases the pole is moving downward , (its conjugate moving upward direction), and the two poles (z and $\bar{z}$) coalesce on the real axis at $t = 1$. This is when the solution $u(x,t)$ blows up on the real axis. In the figure (5.3) we list four cases of (p,q) in algorithm B, namely a, b, c and d, corresponding to (8,7), (10,10), (12,12), and (6,24), respectively. In each case, the motion of q poles are computed by algorithm B; every pole are started from position ∇ at time $t^0 = 0$, and end in position $*$ at time $t^{end} = 1$. In each case one can see at least one of the q poles has a close path (trajectory) to the exact trajectory that appear in figure (2).

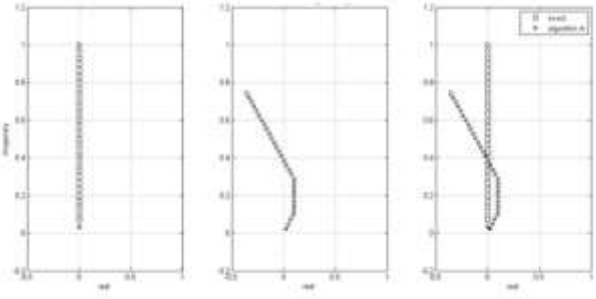

Figure (2) :along all-time $\boldsymbol{t = 0:1}$, the motion of the exact pole, and the approximated poles that computed by algorithm A, in this figure, their motions are aligned from left to right, and then collected in last graph of this figure; all of them are moving downward , (its conjugate moving upward direction-not shown here)

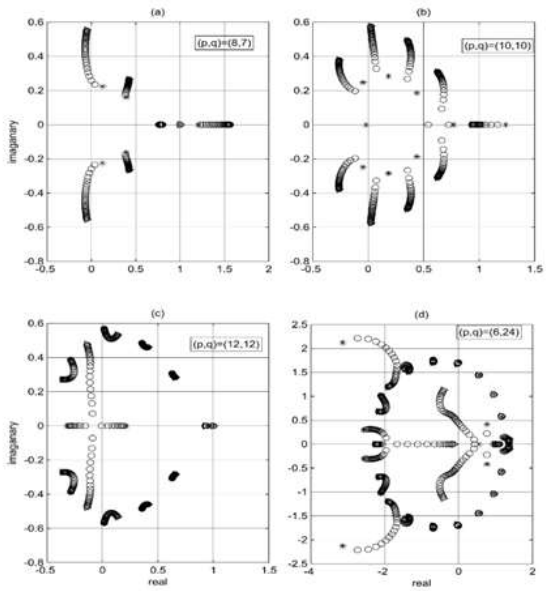

Figure (3) : four cases of algorithm B, , case a, b, c and d for (p, q) is (8,7), (10,10), (12,12), and (6,24), respectively. In each case the motion of q poles are computed by algorithm B, every pole are started from position ∇ at time $t_0 = 0$, and end in position $*$ at time $t_{N=end} = 1$.

Figure (4) compare algorithm A and B with the exact locations of poles along all time from $t = 0$ untile $t = 1$ which corresponding to the motion of this pole. For algorithm B we consider the case of $(p, q) = (12, 12)$ and from the 12 poles we chose the pole number 11.

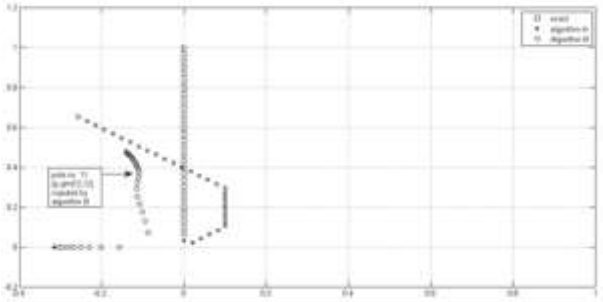

Figure (4) : the motion of pole exact and approximated computed by algorithm A and B; for algorithm B, (p, q)= (12,12), and pole No. 11. In each case the motion of pole are computed along time started from position ∇ at time $t_0 = 0$, and end in position $*$ at time $t_{N=end} = 1$.

Application II (Viscose Burgers Equation)

The second problem is the Burgers equation, Consider viscose Burgers equation

$$u_t + uu_x + \nu u_{xx} = 0 \qquad \nu \in R^+ \qquad (29)$$

$$u(x, 0) = \sin 2\pi x + \frac{1}{2}\sin \pi x \qquad (30)$$

Initial condition (30) is periodic with period '2'. Trefethen [7] studies this problem, with boundary conditions $u(0, t) = u(1, t) = 0$.

Due to step 1 of our algorithms, we have to find spectral solution of the Burgers equation (29), subject to the initial condition (30), after we get this solution. Then we start by applying step 2 and step 3 of our algorithms A and B, to picturing the approximated dynamical system that covered the motion of poles of solution. The figure (5), show the approximation solution of (29) subject to initial condition (30), for some specific value of time t, $t \in (0,1)$ and for $x \in (0,1)$, which is computed by pseudo-spectral methods, with the parameters that specified in the application of algorithms section. Then we applying step 2 of algorithms A and B, to get the approximated locations of pole or (q poles in algorithm B) at each time step. By step 3, we collect all of these locations in one figure, to view the dynamic of pole as time various from $t_0 = 0$ to $t_{end} = 1$. Figures (6) and (7) is used to collect all of locations in one figure; for algorithm A and B respectively. That make us can see the motion of pole in the available domain. Which shows how this pole (q poles) is moving from position ∇ at time $t_0 = 0$, and end in position $*$ at time $t_{end} = 1$.

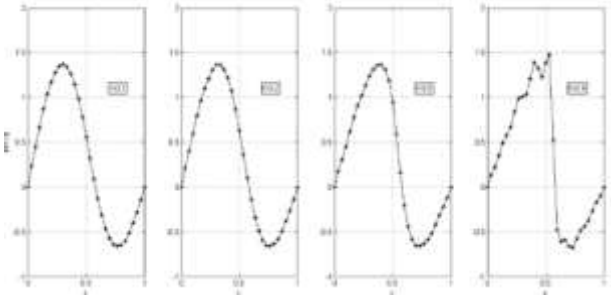

Figure (5) : spectral solution of equation (29) at time t=0.1, 0.2, 0.5, 0.9.

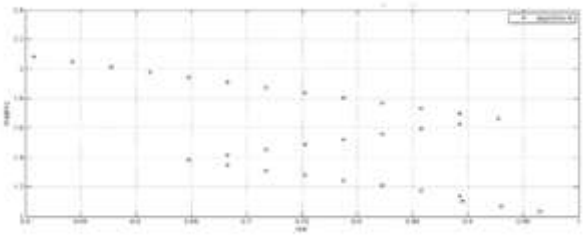

Figure (6): motion of pole from ∇ to * , that computed by algorithm A

The Figures (6) and (7) used to collect all of locations in one figure; for algorithms A and B respectively. which make us can see the motion of pole in the available domain. which shows how this pole (q poles) is moving from position ∇ at time $t^0 = 0$, and ends in position $*$ at time $t^{end} = 1$

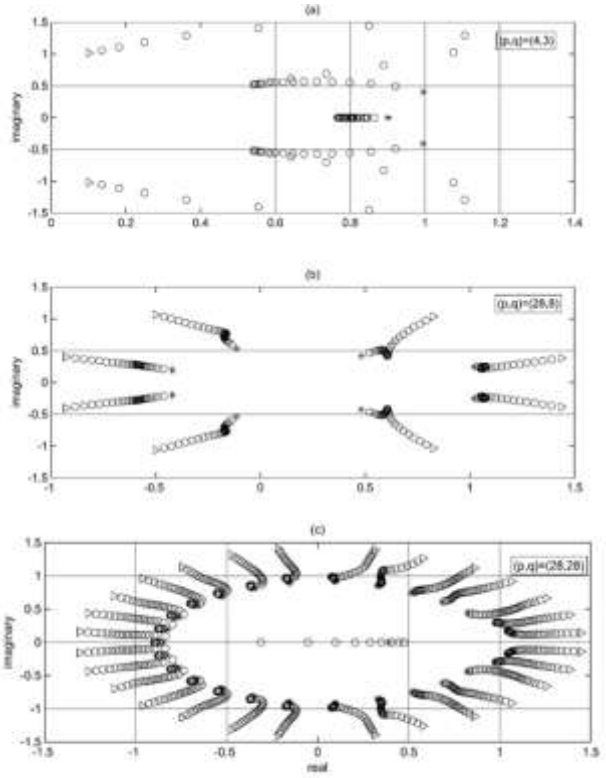

Figure (7) : the motion of q poles from ∇ to * , that computed by algorithm B for application II. for the three cases (a), (b), and (c) for which $(p, q) = (4, 3), (28, 8), and\ (28, 28)$. Respectively

Application III (STO Equation I)

As we mention before, there is no study for motion of poles for the rational solution of the STO equation, here we consider the STO equation

$$u_t + \alpha u_{xxx} + 3\alpha u_x^2 + 3\alpha u u_{xx} + 3\alpha u^2 u_x = 0 \quad (31)$$

The exact solution of (31) is [22]:

$$u(x,t) = \frac{ae^{z_1} + b\cos(z_2) - c\sin(z_3)}{d + e^{z_1} + \sin(z_2) + \cos(z_3)} \quad (32)$$

Which is periodic for $x \in (0,1)$, and $t \in (0,0.25)$, and $z_1 = ax - \alpha a^3 t - x_1$, $z_2 = bx + \alpha b^3 t - x_2$, $z_3 = cx + \alpha c^3 t - x_3$ and $a, b, c, d, x_1, x_2,$ and $x_3 \in \mathbb{R}$.

where the initial condition of (31) is $u_0 = u(x, 0)$. And u is that given in (32). Here we consider $(a, b, c, d, x_1, x_2, x_3) = (2.2, 3, 6, 8, 0.3, 6, 2)$. As in application I, we use (32) to compare the exact locations of pole with the approximated one. The exact location of complex pole can be detected from the equation (32) in complex domain $D = (0,1) \times (0,1)$, when the complex solution that appear in the figure (8)

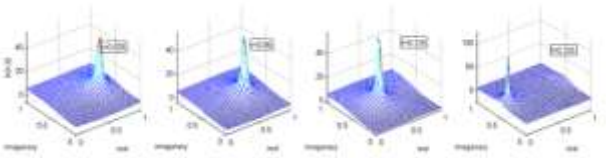

Figure (8): Complex solution of equation (31) for values of $t = 0.025, 0.05, 0.125, 0.225$

Now we have to apply our algorithms, for step 1, the solve STO equation (31) by pseudo-spectral methods, due to periodic of the initial condition, Fourier spectral method is performed, and it can be compute by

$$\hat{u}_t - ik^3\alpha\hat{u} - 3\alpha k^2(\hat{u} * \hat{u}) - 3\alpha k^2(\hat{u} * \hat{u}) + 3ik\alpha(\hat{u} * \hat{u} * \hat{u}) = 0 \quad (33)$$

$$\hat{u}_t = ik^3\alpha\hat{u} + 6\alpha k^2(\hat{w}) - 3ik\alpha(\hat{v}) \quad (34)$$

for $x \in (0,1)$, and $t \in (0,0.25)$, in step1, and then in step 2 we determine the locations of pole approximately at each time step by continue the solution into the complex domain $z = x + iy$, $x, y \in (0,1)$. By step3, we collect all position of pole in one graph, so we get the dynamics of pole, which is moving from right to lift. The figure (9) shows this step, the three columns shows the exact locations and the approximated (by algorithm A) locations respectively.

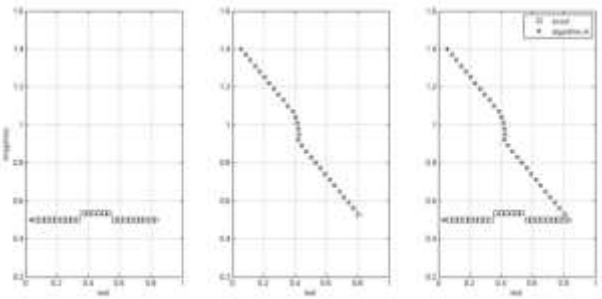

Figure (9) :along all-time $t = 0$ until $t = 0.25$, the motion of the exact pole, and the approximated poles that computed by algorithm A, the motion of pole from ∇ to * , in this figure, their motions are aligned from left to right, and then collected in last graph of this figure.

In Figure (10) we list three cases of algorithm B, for the degree of approximate (p, q), case a, b and c for (p, q) is (14,13), (24,5), and (25,21), respectively. In each case the motion of q poles are computed by algorithm B, every pole are started from position ∇ at time $t_0 = 0$, and end in position $*$ at time $t_{end} = 0.25$. In each case one can see that at least one of the q poles have a close trajectory to the exact trajectory that computed exactly, which appear in previous figure.

Figure (11) compare algorithm A and B with the exact locations of poles along all-time from $t = 0$ untile $t = 0.25$ which corresponding to the motion of this pole. For algorithm B we consider the case of $(p, q) = (24,5)$ and from the 5 poles we chose the pole number 1. Figure (11) also, shows that approximated dynamics by algorithm A and B, are close to the exact dynamic. In addition, all of their motion are from right to lift, while the locations is not coincided with the exact location.

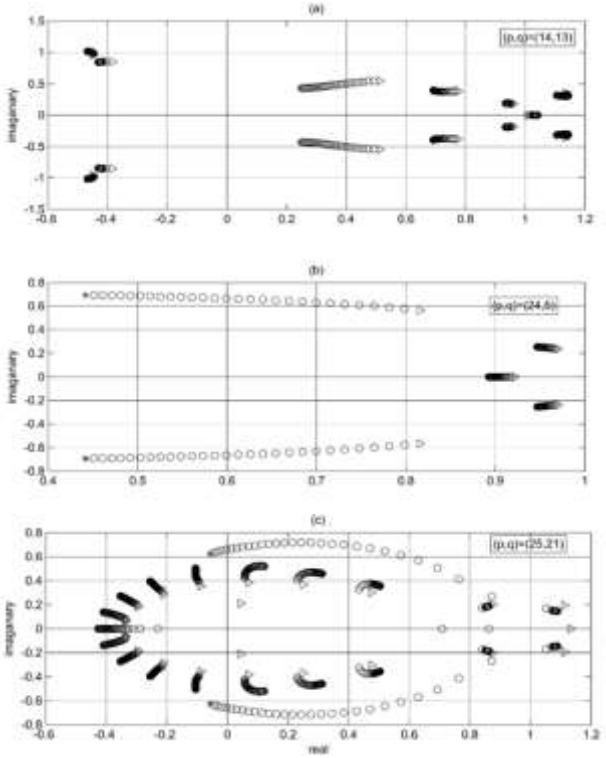

Figure (10) : motion of q poles from ∇ to * , that computed by algorithm B. for the three cases (a), (b), and (c) for which $(p,q) = (14,13), (24,5), and\ (25,21)$. Respectivly .

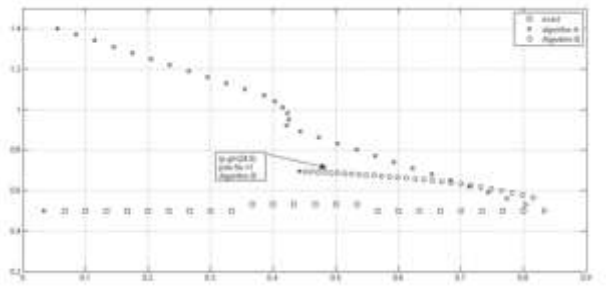

Figure (11) : motion of pole exact and approximated computed by algorithm A and B; for algorithm B, (p,q)= (24,5), and pole No. 1. In each case the motion of pole are computed along time started from position ∇ at time $t_0 = 0$, and end in position $*$ at time $t_{N=end} = 0.25$.

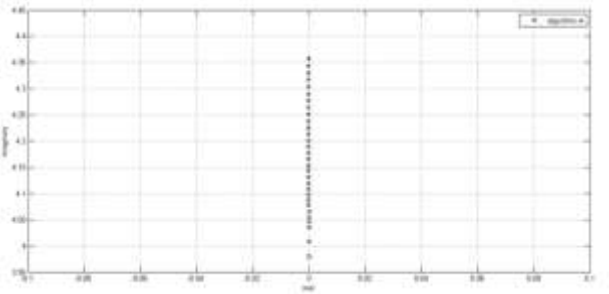

Figure (12): motion of pole from ∇ to * , that computed by algorithm A.

Application IV (STO Equation II)

Consider the STO equation (31) subject to the non-periodic initial condition,

$$u(x,0) = \frac{2ce^x}{ce^x + c} - 1 \qquad (35)$$

this equation has an exact solution [23]. For $c \in R, x \in (0,\pi), t \in (0,5)$.

Now we have to apply our algorithms, for step 1, the solve (31) by pseudo-spectral methods, can be compute by

$$\hat{u}_t + \alpha\, D^3\hat{u} + 3\alpha D(\hat{u}) * D(\hat{u}) + 3\alpha(\hat{u} * D\hat{u}) + 3\alpha(\hat{u} * D^2\hat{u}) = 0 \qquad (36)$$

By using Runge-Kutta method to integrated the equation (36), to get the solution $u(x,t)$ in the form of (26). By step 2, we continue the solution in (step 1) to complex domain $D = (0,\pi) \times (0,2\pi)$ through Padè approximation, and then find the location of pole at each time step. In figures (12) and (13), we list the result of step3 for algorithms A and B respectively. That shows how this pole (q poles) is moving from position ∇ at time $t_0 = 0$, and end in position $*$ at time $t_{end} = 5$.

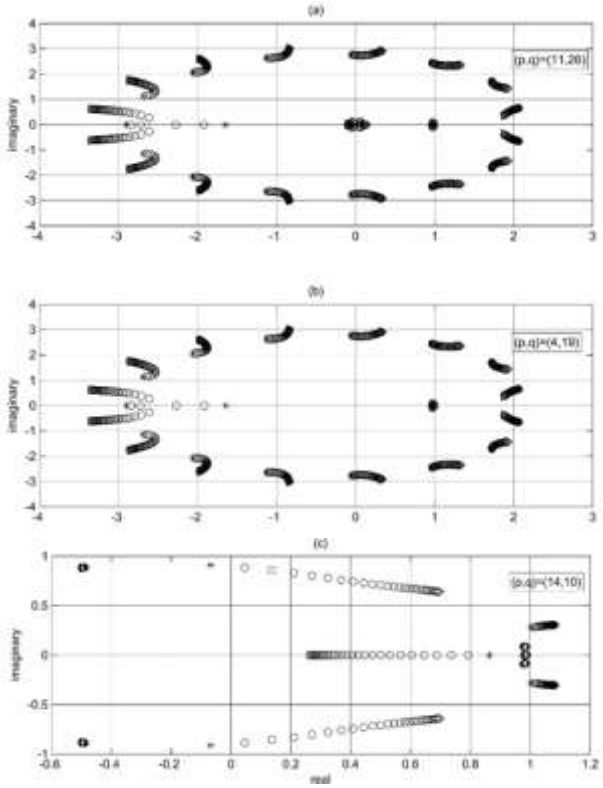

Figure (13): the motion of q poles from ∇ to * , that computed by algorithm B. for the three cases (a), (b), and (c) for which $(p,q) = (11,26), (4,19), and\ (14,10)$. Respectivly .

Discussions And Conclusions

In the implementation of our algorithm, there are some parameters that controlling the accuracy of both of them. The results from application I, show that the algorithm A is more accurate in detecting the position of a pole than

algorithm B, this accuracy is resulted from chose the radius of the neighborhood r, of the region $\mathcal{D}$, in step 2 of this algorithm. If r is as small as possible we get the best results. In all applications, we set ($r = 0.03$), and if we increase this value to ($r = 0.1$ for example) then we lose the accuracy, figure (14), shows this result by repeat the application I, with the new value of r, One can see how the locations of the pole lose its right exact locations. While the parameters of algorithm B, are the degrees of nominator and denominator of Padè approximation, p, and q respectively. Each of them can be vary from 1 to N. In addition, the resulting dynamics is as the number of the roots of denominator, which is equal to q for each value of p. As a result from this coefficients problem, these dynamics which computed by algorithm B, can lie in any region of complex plane. Therefore, we chose the dynamic, which is in our pre-specific region, and ignore the rest of unwanted dynamics.

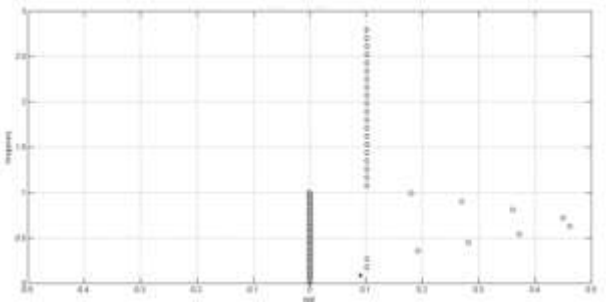

Figure (14): repeating the last column of figure (2) after change the value of r from 0.03 to 0.1

Figure (15), list some cases for the application I for deferent value of p and q. The algorithm A and algorithm B that presented in this paper are good tools to detect the position of poles and picturing their motion in complex plane. Algorithm A can approximated position of exact location better than algorithm B, since it start by initial location of the pole and then compute its next position. while algorithm B is compute all possible poles at each time step by this way it can find the motion of the wanted pole in very accuracy direction, which coincided with our goal; so that algorithm B is best than algorithm A.

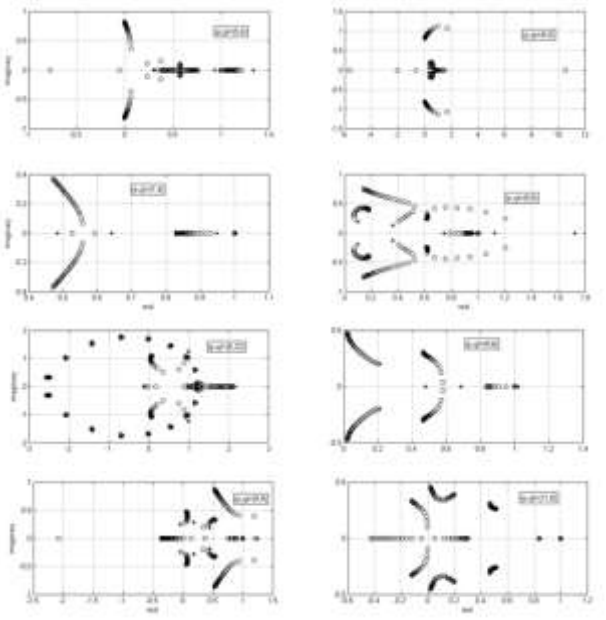

Figure (15): several cases of (p, q) for algorithm B, in application I

REFERENCES

1- Y. Tourigny, and M. Grinfeld, "Deciphering singularities by discrete methods", Math. Comp., 62, pp. 155–169. (1994)

2- C. Sulem, P. L. Sulem, and H. Frisch, "Tracing complex singularities with spectral methods", J. Comput. Phys., 50, pp. 138–161. (1983)

3- C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, "Spectral Methods in Fluid Dynamics", Springer-Verlag, Berlin, (1988).

4- M. Berger, and R.V. Kohn, "A rescaling algorithm for the numerical calculation of blowing-up solutions", Comm. Pure Appl. Math., 41, pp. 841–863. (1988)

5- C. J. Budd, W. Huang, and R. D. Russell, "Moving mesh methods for problems with blow-up", SIAM. J. Sci. Comput., 17, pp. 305–327. (1996)

6- J.A.C. Weideman, "Computing the dynamics of complex singularities of nonlinear PDEs ", SIAM Journal on Applied Dynamical Systems,171-186. (2003)

7- L.N. Trefethen and T. W. Tee, "A Rational Spectral Collocation Method with Adaptively Transformed Chebyshev Grid Points" , SIAM Journal on Scientific Computing, Volume 28 Issue 5. Pages 1798 – 1811. 2006

8- U.Frisch, "Fully developed turbulence and singularities", in Les Houches 1981, North-Holland, Amsterdam (1983).

9- P. Roger, “Spectral Method For Incompressible Viscous Flow”, Applied math. Science v.148. springer (2002).
10- J.W. Cooley, and J.W. Tukey, “An algorithm for the machine calculation of complex Fourier series”. Math. Comp. 19. pp 297-301. (1965).
11- J.W. Cooley, and J.W. Tukey, “An algorithm for the machine calculation of complex Fourier series”. Math. Comp. 19. pp 297-301. (1965).
12- J.Hesthaven, S.Gottlieb and D. Gottlieb "Spectral methods for time-dependent problems", Cambridge UP, Cambridge, UK. (2007)
13- G. A. Baker, and P. Graves-Morris, “Padè Approximants”, 2nd ed., Cambridge University Press, Cambridge, UK, 1996.
14- E.B. Saff “Introduction to Padè Approximants”, center for constructive Approximation, Vanderbilt university, 1976.
15- J.Gilewicz, Y. Kryakin, “Froissart doublets in Padè approximation in the case of polynomial noise” J. Comput. Appl. Math. 153 , 235 – 242. (2003)
16- L. F. Abd-Elall, L.M. Delves,; and J.K. Reid, “A numerical method for locating the zeros and poles of meromorphic function. In Numerical Methods for nonlinear algebraic Equations”, P. Rabinowitz (ed), pp. 47-59. Gordon and Breach, London. 1970
17- K.O. Geddes, “Symbolic computation of Padè approximants”, ACM Transactions on Mathematical Software. Vol. 5, No. 2. pp 218–233. 1979
18- P. Wynn, “On a device for computing the em(Sn) transformation”, MTAC 10, 91–96. (1956)
19- C. Brezinski, “Application du p- algorithms a la quadrature numerique”, C.R. Acad. Sc. Paris t. 270, pp 1252-1253 , (1970)
20- G. Pedro, S.Guttel, and L.N. Trefethen, “Robust Padè Approximation via SVD”, Society for Industrial and Applied Mathematics, SIAM review, Vol. 55, No. 1, pp. 101–117. (2013)
21- G. R. Baker, X. Li, and A. C. Morlet, “Analytic structure of two 1D-transport equations with nonlocal fluxes”, Phys. D, 91, pp. 349–375. (1996).
22- T. Sakajo, Blow-up solutions of Constantin-Lax-Majda equation with a generalized viscosity term, J. Math. Sci. Univ. Tokyo, vol. 10, pp. 187-207. (2003)
23- N.A. Kudryashov, D. I. Sinelshchikov, “Exact solutions of equations for the Burgers Hierarchy”. Applied Mathematics and Computation, Volume 215, Issue No. 3, , Pages 1293–1300, 1 October 2009
24- L. Jianming, D. Jie, and Y. Wenjun, “Backlund Transformation and New Exact Solutions of the Sharma-Tasso-Olver Equation”, Hindawi Publishing Corporation, Abstract and Applied Analysis, Volume 2011, Article ID 935710, 8 pages. , 2011