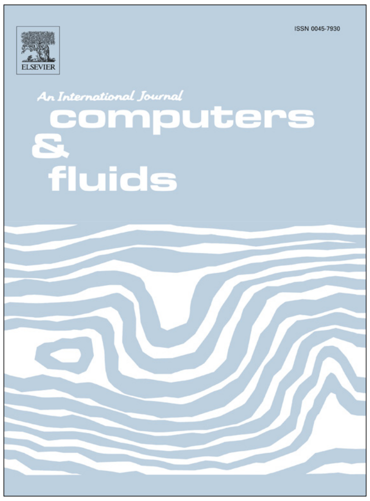


Taylor series expansion using matrices: an implementation in MATLAB®

Carlos Pantaleón, Amitabha Ghosh

PII: S0045-7930(15)00018-3
DOI: http://dx.doi.org/10.1016/j.compfluid.2015.01.009
Reference: CAF 2788

To appear in: *Computers & Fluids*

Received Date: 4 November 2013
Revised Date: 22 August 2014
Accepted Date: 22 January 2015

Taylor series expansion using matrices: an implementation in MATLAB®

Carlos Pantaleón[*], Amitabha Ghosh
Department of Mechanical Engineering, Rochester Institute of Technology, Rochester, NY 14623, United States

Abstract

Taylor series expansions are widely used in engineering approximations, for instance, to develop finite differences schemes or numerical integration methods. This technical note presents a novel technique to generate, display and manipulate Taylor series expansion by using matrices. The resulting approach allows algebraic manipulation as well as differentiation in a very intuitive manner in order to experiment with different numerical schemes, their truncation errors and their structures, while avoiding manual calculation errors. A detailed explanation of the mathematical procedure to generate a matrix form of the Taylor series expansion for a function of two variables is presented along with the algorithm of an implementation in MATLAB®. Example cases of different orders are tabulated to illustrate the generation and manipulation capabilities of this technique. Additionally, an extended application is developed to determine the modified equations of finite difference schemes for partial differential equations, with one-dimensional examples of the wave equation and the heat equation using explicit and implicit schemes.

Keywords

Taylor series; finite differences; truncation error; modified equation; symbolic computation

1. Introduction

Taylor series are primarily presented in introductory calculus courses [1], but the subject is frequently revisited in numerical methods courses for STEM (Science, Technology, Engineering and Mathematics) undergraduates [2, 3] and used in graduate courses of numerical partial differential equations or computational methods in applied sciences [4, 5]. One of the most relevant uses of Taylor series expansions is to develop finite differences schemes or numerical integration methods as well as determining the consistency requirements, order of accuracy and the nature of the truncation errors.

The proper computation of the Taylor series expansion terms is critical to understand the development process of any finite difference scheme or linear multistep method [6]. It assists learning advanced applications such as the modified equation approach [7] in computational fluid dynamics. However, this procedure can become tedious since it requires extensive manual work with a meticulous focus in order to avoid any error while manipulating long analytical expressions. The entire process can be improved by automating the symbolic computation procedure to minimize the gap between the formulation phase and the numerical solution.

There are built-in functions in software packages that provide efficient manipulation of Taylor series expansions in one or several variables [8, 9], but these are only useful for analytical expressions. Other implementations are intended for the numerical methods area, such as a recursive generation of Taylor coefficients for numerical integration of ordinary differential equations [10]. Furthermore, the connection of Taylor series with calculus of divided differences using recursively generated tables is well known in the mathematical and engineering sciences [11].

The present work attempts to deliver, in a more detailed manner, the symbolic manipulation technique for Taylor series expansions using matrices, originally introduced by the authors for teaching a graduate course in the computational fluid dynamics [12]. The same approach can be used in applications of numerical partial differential equation where manipulating Taylor series expansions is required.

[*] Corresponding author.
E-mail address: cjp9444@rit.edu; Telephone: +1 (849) 248-7599

2. Methodology

An effective way to compute and display the coefficients of the Taylor series expansion (TSE) was created by distributing the factorial terms and the binomial terms into two separate matrices. The results are then combined on a Hadamard product [13], also known as entrywise product, whose operator is denoted by the symbol $\odot$. A detailed explanation of the process is shown below:

i. For a function of two variables $u(x, y)$, the TSE of order k and about an origin $(x_o,\ y_o)$, evaluated at a shifted position using step sizes $\{\Delta x,\ \Delta y\}$ and step factors $\{\alpha,\ \beta\}$, is written as:

$$u(x_o + \alpha\Delta x,\ y_o + \beta\Delta y) \cong \sum_{n=0}^{k} \frac{1}{n!}\left(\alpha\Delta x\frac{\partial}{\partial x} + \beta\Delta y\frac{\partial}{\partial y}\right)^{n} u(x_o, y_o)$$

ii. In this approach, the TSE is expressed as a sum in terms of a matrix $\mathbf{A}\ (m\times m)$, where $m = k+1$. Each element $a_{i\,j}$ is a coefficient of the differential operator $(\partial/\partial x)^{i-1}(\partial/\partial y)^{j-1}$, such that:

$$u(x_o + \alpha\Delta x,\ y_o + \beta\Delta y) \cong \sum_{i=1}^{m}\sum_{j=1}^{m} a_{i\,j}\,(\partial/\partial x)^{i-1}(\partial/\partial y)^{j-1}\,u(x_o, y_o)$$

iii. The factorial term in the TSE is arranged into a matrix $\mathbf{B}\ (m\times m)$, with elements defined as: $b_{i\,j} = 1/(i+j-2)!$

iv. Using the binomial theorem [14], the power term is expanded into a Pascal matrix [15], and the coefficients of the differential operators are stored in a matrix $\mathbf{C}\ (m\times m)$, with elements given as:

$$c_{i\,j} = \frac{(i+j-2)!}{(i-1)!\,(j-1)!}(\alpha\Delta x)^{i-1}(\beta\Delta y)^{j-1}$$

v. The matrix representation of the TSE is obtained as the Hadamard product $\mathbf{A} = (\mathbf{B}\odot\mathbf{C})$, and higher order elements are truncated by setting $a_{i\,j} = 0\ \ \forall\ \ i+j > m+1$.

vi. The result for each non-zero element of the matrix $\mathbf{A}$ can be simplified to:

$$a_{i\,j} = \frac{1}{(i-1)!\,(j-1)!}(\alpha\Delta x)^{i-1}(\beta\Delta y)^{j-1}\ \ \forall\ \ i+j \le m+1$$

vii. For an arbitrary number of dimensions, d, the TSE of order k can be expressed using column vectors for the variables $\mathbf{x}$, the origin $\mathbf{x_0}$, step sizes $\Delta\mathbf{x}$, and step factors $\boldsymbol{\alpha}$, such that:

$$u(\mathbf{x_0} + \boldsymbol{\alpha}\odot\Delta\mathbf{x}) \cong \sum_{n_1=0}^{k}\sum_{n_2=0}^{k}\cdots\sum_{n_d=0}^{k}\frac{(\alpha_1\Delta x_1)^{n_1}(\alpha_2\Delta x_2)^{n_2}\cdots(\alpha_d\Delta x_d)^{n_d}}{n_1!\,n_2!\cdots n_d!}\left(\frac{\partial^{n_1+n_2+\cdots+n_d}}{\partial x_1^{n_1}\,\partial x_2^{n_2}\cdots\partial x_d^{n_d}}\right)u(\mathbf{x_0})$$

3. Implementation

A serial implementation of this procedure is very simple, as illustrated in Fig. 1. Some code optimization can be performed by recursively generating the factorial terms or by computing the elements in a parallel manner. Furthermore, matrices obtained for a default order can be previously stored to omit the computing process in future uses. As a limitation of the Taylor series, a variable step size cannot be adopted, instead an adaptive step size selection is suggested [1].

```
for i=1:k+1
  for j=1:k+1
    if (i+j-2) < (k+1)
      $a_{ij} = \frac{1}{(i-1)!\,(j-1)!}(\alpha\Delta x)^{i-1}(\beta\Delta y)^{j-1}$
    end
  end
end
```

Fig. 1: Pseudo code to generate matrix TSE.

A MATLAB® function was created to obtain the matrix **A** as output, given the step factors $\{\alpha,\beta\}$ and the order k. Symbolic step sizes are used to keep track of the differential operators which are omitted for displaying purposes. The generalization of this method into an arbitrary number of dimensions would utilize the multinomial theorem [14]. The current implementation allows the generation, manipulation and combination TSEs of any desired order, to experiment with different numerical schemes, their truncation error and their structure while avoiding manual calculation errors. The source code for this function, as well as the other examples shown in this contribution, is available at http://1drv.ms/1qsQaAL.

In Table 1, the MATLAB® function tay(α,β,k) takes the steps factors and the order as input to compute the TSE in a matrix form as output. The variables dx and dt denote the symbolic step size in space and time, respectively. Thus, for any given element of the TSE, the column number denotes the order of the spatial derivative whereas the row number denotes the order of the time derivative. The discretized terms are written using a simplified notation with a subscript linked to the spatial index i, and a superscript linked to the time level n. The first expression being evaluated, u_{i+1}^{n}, is only shifted in one dimension, whereas the second expression, u_{i-1}^{n+1}, is shifted in space and time. Finally, the third expression is a linear combination of four terms, showing how simple and intuitive it is to combine and manipulate TSEs using matrices.

Table 1: Examples of matrix TSE generation and manipulation.

Expression	Order	Input	**Output:** TSE matrix in MATLAB®
u_{i+1}^{n}	5th	tay(1,0,5)	[1, dx, dx^2/2, dx^3/6, dx^4/24, dx^5/120] [0, 0, 0, 0, 0, 0] [0, 0, 0, 0, 0, 0] [0, 0, 0, 0, 0, 0] [0, 0, 0, 0, 0, 0] [0, 0, 0, 0, 0, 0]
u_{i-1}^{n+1}	3th	tay(-1,1,3)	[1, -dx, dx^2/2, -dx^3/6] [dt, -dt*dx, (dt*dx^2)/2, 0] [dt^2/2, -(dt^2*dx)/2, 0, 0] [dt^3/6, 0, 0, 0]
$u_{i+1}^{n+1} - u_{i+1}^{n-1} - u_{i}^{n+1} + u_{i}^{n-1}$	4th	tay(1,1,4) -tay(1,-1,4) -tay(0,1,4)	[0, 0, 0, 0, 0] [0, 2*dt*dx, dt*dx^2, (dt*dx^3)/3, 0] [0, 0, 0, 0, 0]

+tay(0,-1,4)	[0, (dt^3*dx)/3, 0, 0, 0]
	[0, 0, 0, 0, 0]

Some of the characteristics and benefits of this approach are:

- The resulting matrix form is perfectly suitable for algebraic manipulation (as ordinary operations) as well as differentiation (as row and column shifting), and it serves as a computationally inexpensive way of generating symbolic TSE.

- The same procedure can be extended to additional dimensions by using multidimensional arrays. For instance, the generated structures are Pascal's triangle in 1-D, Pascal's pyramid in 2-D and Pascal's simplex in the general case [16].

- The code is easy to modify, according to specific needs, and its operation is user-friendly.

4. Applications

A useful application of the matrix form TSE is to determine the modified equation of a finite difference discretization, which is the actual partial differential equation (PDE) whose exact solution satisfies the finite difference equation (FDE). Although the procedure for determining the modified equation is extremely laborious, the final results contains highly valued information regarding the nature of the truncation error for numerical schemes involving two or less time levels. Li and Yang [17] showed that the von Neumann analysis and the modified equation approach are fully equivalent, yet the latter is able to handle non-linear problems.

The modified equation approach, originally developed by Warming and Hyett [18], starts by substituting the TSE of each term (to desired order) in the FDE, then a table is constructed to eliminate the higher time derivatives by means of a linear combination of derivatives of the same expanded FDE. The process is continued until every time derivative is eliminated. The final equation shows the nature of the scheme where even-order terms produce dissipation and odd-order terms yield dispersion.

The preliminary step could be done already by using the previously discussed script for matrix form TSE. The most extensive part was developed in a set of MATLAB® scripts that exploits the manipulation properties of the TSE matrix, reducing the differentiation to row or column shifting. The general steps required to obtain the modified equation using TSE in matrix form are as follows:

i. Given the PDE, choose a discretization scheme and obtain the TSE matrix up to the desired order, as detailed in Table 2 with the output shown in Figs. 2 and 3. Notice that multiple contiguous lines in the input columns of Tables 1 and 2 are actually single continuous terms.

ii. The resultant TSE matrix is restructured based on the derivative terms to form the first row of the modified equation table whose elements represent the coefficients of the derivatives.

iii. In every row, the lowest time derivative term (which is not included in the original PDE) is eliminated multiplying by the appropriate coefficient and applying the associated derivative operator to the first row, such that the coefficient sum in the time derivative column is zero.

iv. After the sum of every time derivative column reaches zero, the remaining sum of columns multiplied by the associated derivative terms yields the modified equation. The remaining undesired coefficients (those that don't appear in the original PDE) may be removed by choosing a proper combination of step sizes based on the system properties.

Modified equation examples were created for the first order upwind scheme (FTBS) on the 1-D wave PDE (Table 3) as well as the simple implicit scheme for the 1-D unsteady heat PDE (Table 4). The obtained tables were compiled from the main matrices shown in Figs. 4 and 5, which can be compared with available literature [19]. Other schemes could be easily implemented by the user.

Table 2: Example of required input to obtain TSE matrix of a given discretization.

Governing PDE	**Discretization**	**Input**
1-D wave propagation $\frac{\partial u}{\partial t}+c\frac{\partial u}{\partial x}=0$	FTBS scheme $\frac{u_i^{n+1}-u_i^n}{\Delta t}+c\frac{u_i^n-u_{i-1}^n}{\Delta x}=0$	(tay(0,1,4)-tay(0,0,4))/dt +c*(tay(0,0,4)-tay(-1,0,4))/dx
1-D heat conduction $\frac{\partial u}{\partial t}=\alpha\frac{\partial^2 u}{\partial x^2}$	Simple implicit scheme $\frac{u_i^{n+1}-u_i^n}{\Delta t}=\alpha\frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{(\Delta x)^2}$	(tay(0,1,4)-tay(0,0,4))/dt -a*(tay(1,1,4)-2*tay(0,1,4)+tay(1,-1,4))/dx^2

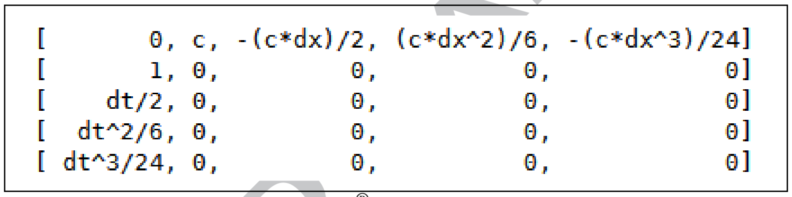

```
[       0, c, -(c*dx)/2, (c*dx^2)/6, -(c*dx^3)/24]
[       1, 0,         0,          0,            0]
[    dt/2, 0,         0,          0,            0]
[  dt^2/6, 0,         0,          0,            0]
[ dt^3/24, 0,         0,          0,            0]
```

Fig. 2: TSE matrix generated in MATLAB® for the wave propagation using a FTBS scheme.

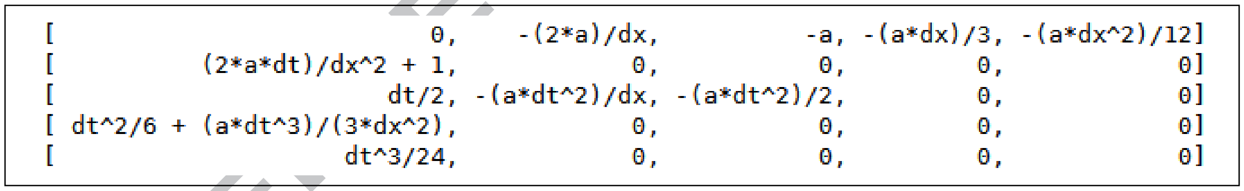

```
[                          0,    -(2*a)/dx,          -a, -(a*dx)/3, -(a*dx^2)/12]
[          (2*a*dt)/dx^2 + 1,            0,           0,         0,            0]
[                       dt/2, -(a*dt^2)/dx, -(a*dt^2)/2,         0,            0]
[ dt^2/6 + (a*dt^3)/(3*dx^2),            0,           0,         0,            0]
[                    dt^3/24,            0,           0,         0,            0]
```

Fig. 3: TSE matrix generated in MATLAB® for the heat conduction using a simple implicit scheme.

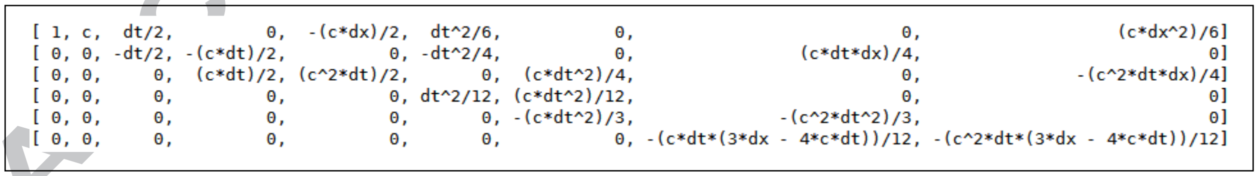
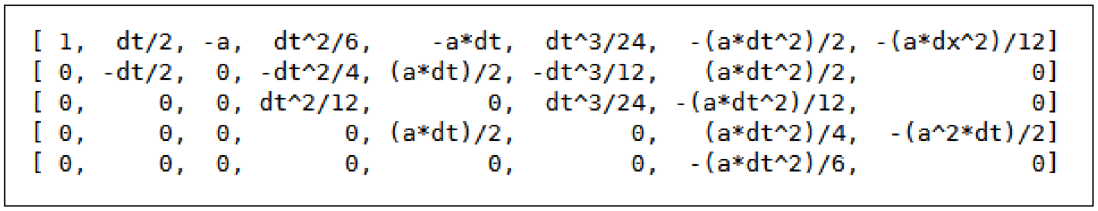

```
[ 1, c,  dt/2,         0,  -(c*dx)/2,  dt^2/6,          0,                         0,                         (c*dx^2)/6]
[ 0, 0, -dt/2, -(c*dt)/2,          0, -dt^2/4,          0,               (c*dt*dx)/4,                                  0]
[ 0, 0,     0,  (c*dt)/2, (c^2*dt)/2,       0,  (c*dt^2)/4,                        0,                    -(c^2*dt*dx)/4]
[ 0, 0,     0,         0,          0, dt^2/12, (c*dt^2)/12,                        0,                                  0]
[ 0, 0,     0,         0,          0,       0, -(c*dt^2)/3,            -(c^2*dt^2)/3,                                  0]
[ 0, 0,     0,         0,          0,       0,           0, -(c*dt*(3*dx - 4*c*dt))/12, -(c^2*dt*(3*dx - 4*c*dt))/12]
```

Fig. 4: Main matrix of modified equation in MATLAB® for wave propagation using a FTBS scheme.


```
[ 1,  dt/2, -a,  dt^2/6,     -a*dt,  dt^3/24,  -(a*dt^2)/2, -(a*dx^2)/12]
[ 0, -dt/2,  0, -dt^2/4, (a*dt)/2, -dt^3/12,    (a*dt^2)/2,            0]
[ 0,     0,  0, dt^2/12,         0,  dt^3/24, -(a*dt^2)/12,            0]
[ 0,     0,  0,       0, (a*dt)/2,         0,   (a*dt^2)/4,  -(a^2*dt)/2]
[ 0,     0,  0,       0,         0,        0,  -(a*dt^2)/6,            0]
```

Derivatives	u_t	u_x	u_{tt}	u_{tx}	u_{xx}	u_{ttt}	u_{ttx}	u_{txx}	u_{xxx}
FTBS 1-D wave PDE	1	c	$\frac{\Delta t}{2}$	0	$-c\frac{\Delta x}{2}$	$\frac{\Delta t^2}{6}$	0	0	$c\frac{\Delta x^2}{6}$
$-\frac{\Delta t}{2}\frac{\partial}{\partial t}$			$-\frac{\Delta t}{2}$	$-c\frac{\Delta t}{2}$	0	$-\frac{\Delta t^2}{4}$	0	$c\frac{\Delta t\Delta x}{4}$	0
$c\frac{\Delta t}{2}\frac{\partial}{\partial x}$				$c\frac{\Delta t}{2}$	$c^2\frac{\Delta t}{2}$	0	$c\frac{\Delta t^2}{4}$	0	$-c^2\frac{\Delta t\Delta x}{4}$
$\frac{\Delta t^2}{12}\frac{\partial^2}{\partial t^2}$						$\frac{\Delta t^2}{12}$	$c\frac{\Delta t^2}{12}$	0	0
$-c\frac{\Delta t^2}{3}\frac{\partial^2}{\partial x\partial t}$							$-c\frac{\Delta t^2}{3}$	$-c^2\frac{\Delta t^2}{3}$	0
$c\frac{\Delta t}{12}(4c\Delta t-3\Delta x)\frac{\partial^2}{\partial x^2}$								$\frac{c\Delta t}{12}(4c\Delta t-3\Delta x)$	$\frac{c^2\Delta t}{12}(4c\Delta t-3\Delta x)$
Sum of coefficients	1	c	0	0	$c\frac{\Delta x}{2}(\nu-1)$	0	0	0	$c\frac{\Delta x^2}{6}(2\nu^2-3\nu+1)$

Fig. 5: Main matrix of modified equation in MATLAB® for heat equation with a simple implicit scheme.

Table 3: Modified equation generation for the 1-D wave propagation using a FTBS scheme.

Table 4: Modified equation generation for the 1-D heat conduction with a simple implicit scheme.

From Tables 3-4 the respective modified equations can be obtained from the last rows:

- Wave propagation: $\frac{\partial u}{\partial t}+c\frac{\partial u}{\partial x}+c\frac{\Delta x}{2}(\nu-1)\frac{\partial^2 u}{\partial x^2}+c\frac{(\Delta x)^2}{6}\left(2\nu^2-3\nu+1\right)\frac{\partial^3 u}{\partial x^3}=0,\quad \nu\equiv c\frac{\Delta t}{\Delta x}$
- Heat conduction: $\frac{\partial u}{\partial t}-\alpha\frac{\partial^2 u}{\partial x^2}-\frac{\alpha}{2}\left(\alpha\Delta t+\tfrac{1}{6}(\Delta x)^2\right)\frac{\partial^4 u}{\partial x^4}=0$

5. Conclusions

The current work presents a detailed documentation of a novel technique to generate, display and manipulate Taylor series expansion by using matrices. An implementation in MATLAB® shows several examples and is used to determine the modified equation in computational fluid dynamics. This approach addresses the rigorous analysis required to understand some of the core concepts of finite differences schemes and linear multistep methods. The integration of this tool allows one to perform very extensive algebraic manipulations and differentiations while reducing the required time and the risks of manual computation errors. The combination of all these active learning features will stimulate students and researchers to appreciate the analytical theory and the tools for abstract thinking.

References

[1] MIT OpenCourseWare. (2005). *Manipulating Taylor Series*. Retrieved from http://ocw.mit.edu/high-school/calculus/taylor-series/manipulating-taylor-series/

Derivatives	u_t	u_{tt}	u_{xx}	u_{ttt}	u_{txx}	u_{tttt}	u_{ttxx}	u_{xxxx}
Simple implicit 1-D heat PDE	1	$\frac{\Delta t}{2}$	$-\alpha$	$\frac{\Delta t^2}{6}$	$-\alpha\Delta t$	$\frac{\Delta t^3}{24}$	$-\frac{\alpha}{2}\Delta t^2$	$-\frac{\alpha}{12}\Delta x^2$
$-\frac{\Delta t}{2}\frac{\partial}{\partial t}$		$-\frac{\Delta t}{2}$	0	$-\frac{\Delta t^2}{4}$	$\alpha\frac{\Delta t}{2}$	$-\frac{\Delta t^3}{12}$	$\frac{\alpha}{2}\Delta t^2$	0
$\frac{\Delta t^2}{12}\frac{\partial^2}{\partial t^2}$				$\frac{\Delta t^2}{12}$	0	$\frac{\Delta t^3}{24}$	$-\frac{\alpha}{12}\Delta t^2$	0
$\alpha\frac{\Delta t}{2}\frac{\partial^3}{\partial t\,\partial x^2}$					$\alpha\frac{\Delta t}{2}$	0	$\frac{\alpha}{4}\Delta t^2$	$-\frac{\alpha^2}{2}\Delta t$
$-\alpha\frac{\Delta t^2}{6}\frac{\partial^4}{\partial t^2\partial x^2}$							$-\frac{\alpha}{6}\Delta t^2$	0
Sum of coefficients	1	0	$-\alpha$	0	0	0	0	$-\frac{\alpha}{2}\left(\alpha\Delta t+\frac{\Delta x^2}{6}\right)$

[2] Griffiths, D. F., & Higham, D. J. (2010). The Taylor Series Method. In *Numerical methods for ordinary differential equations: Initial value problems* (3rd ed., pp. 33-42). London, United Kingdom: Springer.
[3] University of South Florida. (2009). *Taylor Series Revisited: Numerical Methods for the STEM Undergraduate*. Retrieved from http://nm.mathforcollege.com/topics/taylor_series.html
[4] Strang, G. (2008). *Computational Science and Engineering I*. Retrieved from http://ocw.mit.edu/courses/mathematics/18-085-computational-science-and-engineering-i-fall-2008/
[5] Strang, G. (2006). *Mathematical Methods for Engineers II*. Retrieved from http://ocw.mit.edu/courses/mathematics/18-086-mathematical-methods-for-engineers-ii-spring-2006/

[6] Hairer, E., & Wanner, G. (2010). Linear multistep method. *Scholarpedia*, *5*(4), 4591. doi:10.4249/scholarpedia.4591

[7] Griffiths, D. F., & Sanz-Serna, J. M. (1986). On the Scope of the Method of Modified Equations. *SIAM Journal on Scientific and Statistical Computing*, *7*(3), 994-1008. doi:10.1137/0907067

[8] MathWorks®. (2014). *Taylor series expansion - MATLAB® documentation center*. Retrieved from http://www.mathworks.com/help/symbolic/taylor.html

[9] Schöpf, R. (1998). *A REDUCE package for manipulation of Taylor series*. Retrieved from http://www.reduce-algebra.com/docs/taylor.pdf

[10] Jorba, À., & Zou, M. (2005). A Software Package for the Numerical Integration of ODEs by Means of High-Order Taylor Methods. *Experimental Mathematics*, *14*, 99-117. doi:10.1080/10586458.2005.10128904

[11] Richardson, C. H. (2012). *An Introduction to the Calculus of Finite Differences* (2nd ed.). Whitefish, MT: Literary Licensing.

[12] Ghosh, A., & Pantaleón, C. (2013). Teaching Computational Fluid Dynamics Using MATLAB®. *Proceedings of ASME International Mechanical Engineering Congress and Exposition*, *5*. doi:10.1115/IMECE2013-66458

[13] Million, E. (2007). *The Hadamard Product*. Retrieved from University of Puget Sound website: http://buzzard.ups.edu/courses/2007spring/projects/million-paper.pdf

[14] Hildebrand, A. J. (2009). *Binomial and multinomial coefficients*. Retrieved from University of Illinois at Urbana-Champaign website: http://www.math.uiuc.edu/~hildebr/461/binomial.pdf

[15] Alan, E., & Strang, G. (2004, March). Pascal Matrices. *The American Mathematical Monthly*, *111*(3), 361-385. doi:10.2307/4145127

[16] Rowland, E. (2002). *Pascal's Simplices*. Retrieved from http://www.math.rutgers.edu/~erowland/pascalssimplices.html

[17] Li, J., & Yang, Z. (2013). The von Neumann analysis and modified equation approach for finite difference schemes. *Applied Mathematics and Computation*, *225*(1), 610-621. doi:10.1016/j.amc.2013.09.046

[18] Warming, R. F., & Hyett, B. J. (1974). The Modified Equation Approach to the Stability and Accuracy Analysis of Finite-Difference Methods. *Journal of Computational Physics*, *14*, 159-179. doi:10.1016/0021-9991(74)90011-4

[19] Pletcher, R. H., Tannehill, J. C., & Anderson, D. H. (2011). Applications of Numerical Methods to Selected Model Equations. In *Computational Fluid Mechanics and Heat Transfer* (3rd ed., pp. 103-113). Boca Raton, FL: CRC Press.

- We generated Taylor series expansions by using symbolic matrices in MATLAB®.
- The risk of manual computation error in Taylor series expansions was reduced.
- New numerical schemes may be developed and analyzed using this technique.
- Order, accuracy and stability of CFD schemes can be researched using matrices.
- Examples are given for the generation of the modified equations for PDE's.