Accuracy of the simplified finite difference method via LU- factorization for system of non-linear ODEs

by

Tousif Iqra

A thesis
Submitted for the Degree of Master of Science
in
Mathematics

School of Natural Sciences,
National University of Sciences and Technology,
H-12, Islamabad, Pakistan

FORM TH-4

National University of Sciences & Technology

MS THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by: Ms. Tousif Iqra, Regn No. 00000203235 Titled: ""Accuracy of the simplified finite difference method via LU-factorization for system of non-linear ODEs" be accepted in partial fulfillment of the requirements for the award of MS degree.

Examination Committee Members

1. Name: Dr. Mujeeb Ur Rehman Signature: ____________________

2. Name: Dr. Meraj Mustafa Hashmi Signature: ____________________

External Examiner: Prof. Masood Khan Signature: ____________________

Supervisor's Name: Dr. Muhammad Asif Farooq Signature: ____________________

Head of Department

28-9-2020

Date

COUNTERSINGED

Date: 28/9/2020

Dean/Principal

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS thesis written by **Ms. Tousif Iqra,** (Registration No. **00000203235**), of **School of Natural Sciences** has been vetted by undersigned, found complete in all respects as per NUST statutes/regulations, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS/M.Phil degree. It is further certified that necessary amendments as pointed out by GEC members and external examiner of the scholar have also been incorporated in the said thesis.

Signature: MAsif

Name of Supervisor: Dr. Muhammad Asif Farooq

Date: 28/09/2020

Signature (HoD): ______

Date: 28-09-2020.

Signature (Dean/Principal): ______

Date: 28-09-2020

Dedicated

to

My Beloved Parents

Acknowledgements

Fa Inna ma'al 'usri yusra (Verily, with hardship comes ease).

[94:6], Qur'an - Surah Ash-Sharh **(The Relief)**

First and foremost, praises and thanks to the God, the Almighty, for his showers of blessings and mercy bestowed upon me through the difficulties of life. I seek His guidance, and pray for ease throughout this life and the life hereafter. Secondly, I would like to express my deepest gratitude to my research supervisor Dr. Muhammad Asif Farooq for sincere guidance, vision, dynamism, motivation and constant encouragement throughout this research. I would cordially like to pay my earnest and honest gratitude to my family especially my parents, siblings (Javeria Tabassum and Dr. Abdul Rehman), and my friends specially Naila Mahreen, Amna Sadiq, Mamoona Shahid, Sidra, Muhammad Irfan and who were there for me every single hour when I needed them the most and entrusting me their unconditional support, prayers and patience. It wouldn't be possible for me to make out this far without their support.

Abstract

The main purpose of this thesis is to introduce a refined and reliable numerical method. The method introduced in this work for the solution of non-linear ODEs with boundary value problem is called simplified finite difference method (SFDM) and can be defined as the extension of finite difference method which is used for the solution of linear ODEs and PDEs. Quasilinearlization technique is used to transfer the non-linear coupled ODEs into linear ODEs. We write a general introduction of well-known methods for the approximate solution of differential equations and to find an error between the exact solution and approximate solution. We use SFDM for the different number of coupled ODEs and the result gives the reliability of this method. These problems are taken from fluid mechanics. The results of theses ODEs are compared with the other methods to check the accuracy, efficiency and reliability that we expected.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we give an introduction of the finite difference method(FDM) and Taylor series. We introduce some approximate methods for the solution of ODEs and PDEs like FEM, FVM and spectral methods. Some Techniques are introduced for coupled ODEs such as Quasi-linearlization, bvp4c and numerical procedure for the simplified finite difference method(SFDM).

1.1 Finite difference method (FDM)

The finite difference is one of the oldest and simplest methods for the solution of differential equations that approximate the derivative. In 2000 Ditkowskti [1] work on the error bound of finite difference to explore the rate of convergence to approximate the PDEs. They observed error bound to depend on the time and mesh size and for their purpose the use of the parabolic and hyperbolic partial differential equation. In 2001 Mickens [2] for the construction of differential equations introduced a non-structural finite difference with its applications and rules. In 2006 Farjadpour et al. [3] work on the increase of accuracy of finite difference method which reduces due to the discretization so the use subpixel smoothing to fix to improve the accuracy and they designed it properly. In (2009) McGee et al. [4] work with the coupled flow models for transport with the finite difference method. They work with the blood flow in the vessel and for this, they use the Navier-stokes equation and also plasma flow in the vessel. They form the coupled equations and solved these by the finite difference method and Schwartz

Method is also used for it. In (2010) Dolicani et al. [5] work with finite difference methods in a thin plate. In the process, the differential equation is replaced with the difference equation and also use to solve the thin plate bending. It can find solutions for stress, momentum, strain and plate deflection. Mehra et al. [6] work n the comparison of the finite difference, Wavelet Galerkin and spectral methods. The plot the graphs and compare the results using MATLAB. In (2011) Chambolle et al.[7] they work with the variation problem by the approximation of the finite difference method. For an upwind, they give the dual formulation of the finite difference method. They showed the effect of the multiscale method. They also give an example of numerical solutions to prove their quantitative and qualitative behavior. In(2012) Sungu et al.[8] introduced the hybrid method for the non-linear partial differential equation. Because different methods are used for different subdomains like finite difference and differential transformation this method is hybrid. The main objective of this method was to achieve the accuracy of finite difference and the flexibility of differential transform. The finite difference method is used for the discretization and time operator is obtained by the differential transformation. This method showed faster results and iterative procedures for the calculation of accurate solutions. In (2013) Izadian et al. [9] worked on the application of the finite difference method and solved the elliptic equation on the irregular mesh points. Its application is in Dirichlet boundary condition for 3 dimensional Poisson's equation on irregular grids. Taylor series expansion is the use and approximation of finite differences. Results are also given which shows the efficiency of this method. In (2013) Lakshmi et al. [10] worked with the ODEs with the linear boundary conditions and solve it by finite difference method. The hyperbolic and elliptic partial differential equations are changed into an ordinary differential equation with boundary and initial boundary value problems. The central difference is used for this replacement. Then it is solved by the Numerov-type method. This method can be used for many hyperbolic and elliptic PDEs which shows that the method is flexible. In (2015) Gulkac [11] worked on the implicit finite difference method. The heat equation is solved with the moving boundary problems. The accuracy and efficiency are checked by the Fourier series and two-dimensional heat equation. The application of this method is easier than the other methods like finite element and spectral methods. Leonhard Euler (1707-1783) already know about it for one-dimensional space and this is extended to two dimensional by Carl David Tolme Runge (1856-1927) in 1908. The work on the finite difference method began in the 1950s in numerical applications and their development on the computer is done for the solution of complex problems and simulation of complex problems in technology and science. Results of partial differential equations are obtained during the last five decades regarding the stability, convergence and accuracy of finite difference method.

In FDM, the derivatives in the original equation is replaced by the finite differences. To get higher accuracy, it can increase the order of an element. The use of a regular grid can help to fit the simulation in a box-shaped geometry. Large scale simulation can be solved by the regular grid on the supercomputer.

The main purpose of FDM is to find an approximation of differential equations. The boundary value problems in which the conditions are given on the edge of their domains which relate with the derivative on some time or space give the required function. In FDM the derivatives are replaced by the approximation which leads to an algebraic system of equations that can be solved instead of the original differential equations. Before applying the FDM we should know how to use the derivative approximation on the function.

1.2 Taylor Series for FDM

1.2.1 Taylor series in 1D

A Taylor series is a function expansion about some point. Taylor series for one-dimension is real function $w(q)$ expansion about $q = b$(point) is given by

$$w(q+\Delta q) = w(q)+\Delta q w'(q)+\frac{(\Delta q)^2}{2!}w''(q)+\frac{(\Delta q)^3}{3!}w'''(q)+\frac{(\Delta q)^4}{4!}w^{(4)}(\xi_1), \quad \xi_1 \epsilon (q, q+\Delta q) \tag{1.1}$$

where ξ_1 is some number between q and $q + \Delta q$

$$w(q-\Delta q) = w(q)-\Delta q w'(v)+\frac{(\Delta q)^2}{2!}w''(q)-\frac{(\Delta q)^3}{3!}w'''(v)+\frac{(\Delta q)^4}{4!}w^{(4)}(\xi_2),\ \ \xi_2\epsilon(q-\Delta q, q) \tag{1.2}$$

$$w(q+2\Delta q) = w(q)+2\Delta q w'(q)+4\frac{(\Delta q)^2}{2!}w''(q)+8\frac{(\Delta q)^3}{3!}w'''(q)+16\frac{(\Delta q)^4}{4!}w^{(4)}(\xi_3),\ \xi_3\epsilon(q, q+2\Delta q) \tag{1.3}$$

$$w(q-2\Delta q) = w(q)-2\Delta q w'(q)+4\frac{(\Delta q)^2}{2!}w''(q)-8\frac{(\Delta q)^3}{3!}w'''(q)+16\frac{(\Delta q)^4}{4!}w^{(4)}(\xi_4),\ \xi_4\epsilon(q-2\Delta q, q) \tag{1.4}$$

For $b = 0$, the series expansion is Maclaurin series

1.2.2 Forward difference formulas

Here we derive forward difference formula. Let us consider

$$w(q + \Delta q) = w(q) + \Delta q w'(q) + \frac{(\Delta q)^2}{2!}w''(\xi), \quad \xi\epsilon(q, q + \Delta q). \tag{1.5}$$

Rearranging the equation (1.5) gives

$$\frac{w(q + \Delta q) - w(q)}{\Delta q} - w'(q) = \frac{\Delta q}{2!}w''(\xi), \quad \xi\epsilon(q, q + \Delta v), \tag{1.6}$$

where

$$w'(q) = \frac{w(q + \Delta q) - w(q)}{\Delta q} + O(\Delta q) \tag{1.7}$$

is called first order forward difference approximation.

1.2.3 Second order forward difference method

Similarly, the second order forward difference are

$$w'(q) = \frac{-3w(q) + 4w(q + \Delta q) - w(q + 2\Delta q)}{2\Delta q} + O(\Delta q^2), \tag{1.8}$$

$$w''(q) = \frac{2w(q) - 5w(q + \Delta q) + 4w(q + 2\Delta q) - w(q + 3\Delta q)}{\Delta q^3} + O(\Delta q^2). \tag{1.9}$$

1.2.4 Third order forward difference method

The third order forward difference is

$$w'(q) = \frac{-w(q+2\Delta q) + 6w(q+\Delta q) - 3w(q) - 2w(q-\Delta q)}{6\Delta q} + O(\Delta q^3). \tag{1.10}$$

1.2.5 Backward difference formulas

If $h < 0$, say $h = -\Delta q$ where $\Delta q > 0$ then

$$w'(q) = \frac{w(q) - w(q-\Delta q)}{\Delta q} + O(\Delta q). \tag{1.11}$$

1.2.6 Second order backward difference method

The second order backward difference formulas are

$$w'(q) = \frac{-3w(q) - 4w(q-\Delta q) + w(q-2\Delta q)}{2\Delta q} + O(\Delta q^2) \tag{1.12}$$

$$w''(q) = \frac{2w(q) - 5w(q-\Delta q) + 4w(q-2\Delta q) - w(q-3\Delta q)}{\Delta q^3} + O(\Delta q^2) \tag{1.13}$$

1.2.7 Third order backward difference method

Similarly the third order backward difference formula is

$$w'(q) = \frac{2w(q+\Delta q) + 3w(q) - 6w(q-\Delta q) + w(q-2\Delta q)}{6\Delta q} + O(\Delta q^3) \tag{1.14}$$

1.2.8 Central difference formulas

By subtracting equation (1.3) from equation (1.4) gives

$$w(q+\Delta q) - w(q-\Delta q) = 2\Delta q w'(q) + \Delta q^3 \frac{w'''(\xi_1) + w'''(\xi_2)}{12} \tag{1.15}$$

$$\frac{w(q+\Delta q) - w(q-\Delta q)}{2\Delta q} - w'(q) = \Delta q^2 \frac{w'''(\xi_1) + w'''(\xi_2)}{12} \tag{1.16}$$

The second order central difference formula is

$$w'(q) = \frac{w(q+\Delta q) - w(q-\Delta q)}{2\Delta q} + O(\Delta q^2) \tag{1.17}$$

$$w''(q) = \frac{w(q+\Delta q) - 2w(q) + w(q-\Delta q)}{\Delta q^2} + O(\Delta q^2) \tag{1.18}$$

1.2.9 Fourth order backward difference method

Similarly, 4th order backward difference formulas are

$$w'(q) = \frac{-w(q+2\Delta q) + 8w(q+\Delta q) - 8w(q-\Delta q) + w(q-2\Delta q)}{12\Delta q} + O(\Delta q^4) \quad (1.19)$$

$$w''(q) = \frac{-w(q+2\Delta q) + 16w(q+\Delta q) - 30w(q) + 16w(q-\Delta q) - w(q-2\Delta q)}{12\Delta q} + O(\Delta q^2) \quad (1.20)$$

1.2.10 Taylor expansions in 2D

The Taylor expansion in 2D is given by

$$w(z_0 + \triangle z, v_0 + \triangle v) = w(z_0, v_0) + w_z(z_0, v_0) \triangle z + w_v(z_0, v_0) \triangle v + \frac{1}{2}[w_{zz}(z_0, v_0) \triangle z^2 + 2w_{zv}(z_0, v_0) \triangle z \triangle v + w_{vv}(z_0, v_0) \triangle v^2] + O(\triangle z^3 + \triangle v^3) \quad (1.21)$$

1.3 Finite element method (FEM)

In FEM the complex problems are changed into simple problems and then obtain the solution of such complex problems. The solution to such a problem is approximated because the real complex problem is replaced by a simple problem. For the exact solution of the practical problem the mathematical tools which are mostly use will not be useful.

FEM is preferred to find the solution to such a given problem over other methods. Approximate solutions can be refined for working on computational methods. FEM consists of small subregions which are interconnected and known as finite elements from which the solution region is build up. The stresses and displacement of complex geometry structures are difficult to find exactly for this purpose, it is divided into small parts and it is used to approximate by these several parts which are finite elements. The condition for which the structure is equilibrium and the assumption of solution which is approximate is done in each piece. For the stresses and displacement when the conditions are satisfied they give an approximate solution.

1.3.1 Procedural Steps in FEM

1. Discretization (Selection of element Geometry)
2. Selection of the appropriate shape function. Determining the pattern for unknown variable distribution across the continuum.
3. Development of the finite element equation. The application of appropriate principles to form the equation governing the continuum and rewriting is presented in the form of an equation by incorporating an appropriate form function.
4. Assemble the equations of the elements to obtain the global equation
5. Solution for unknowns.

1.4 Finite volume method (FVM)

Conservation properties are an important feature of FVM. The conservation principle is to apply for each part of the control volume. Global conservation is also applicable to it. It is applied to the rectangular Cartesian grids, non-orthogonal and also for unstructured grids.

1.4.1 Procedural steps in FVM

1. The flow domain is divided into small control volumes.
2. In each control volume, the variables at the grid points are stored and defined at the center of it.
3. For the next process extra nodes on the boundary are often added
4. Over each control volume the equations which transport are integrated.

1.5 Spectral Methods

Spectral methods give extremely accurate results. These methods have been studied intensively. The methods depend on the application and identify the nodes by collocation, Galerkin and tau.

1.5.1 Collocation methods

Collocation methods are applicable for the non-linear problems and also for the coefficients which are complex.

1.5.2 Galerkin methods

Galerkin methods can give more convenient analysis and also its advantage is to optimal error estimates.

1.5.3 Tau method

The tau method is used when both methods collocations and Galerkin cannot give results.

In all of these methods, the disadvantage is that the condition number increase due to the discretization of matrices, the rounding error reduces the exponential accuracy that we expected theoretically. The discretization also makes difficult to use algebraic solvers. The solution of the fourth-order equation is difficult due to these disadvantages because approximation is applied for the higher-order and stability and accuracy are not guaranteed. The methods such as Hermite, Legendre polynomials, Chebyshev sinc and Fourier functions are observed and used to build a trivial function. These methods sometimes transfer the self-adjoint problems into discrete algebraic, non-symmetric problems. But all of these disadvantages can be reduced by a proper choice of trial and test functions. MATLAB is used for numerical experiments and can give an accurate result.

1.6 Quasi-linearlization for linear scalar second order ODE

Let us consider the non-linear second-order differential equation [21]

$$z^{''} = g(x, z, z^{'}), \tag{1.22}$$

with boundary conditions

$$z(0) = 0, z(L) = A. \tag{1.23}$$

Let

$$\chi(x, z, z^{'}, z^{''}) = z^{''} - g(x, z, z^{'}). \tag{1.24}$$

To derive the recurrence equation, note the n^{th} and $(n+1)^{th}$ iterations by z_n and z_{n+1}, respectively, and required that, for the two iterations, $\chi = 0$. For the n^{th} iteration, this gives

$$z_n^{''} - g(x, z_n, z_n^{'}) = 0. \tag{1.25}$$

For the $(n+1)^{th}$ iteration, we get

$$\begin{aligned} \chi(x, z_n, z_{n+1}^{'}, z_{n+1}^{''}) = \chi(x, z_n, z_n^{'}, z_n^{''}) + (\frac{\partial \chi}{\partial z})_n (z_{n+1} - z_n) + \\ (\frac{\partial \chi}{\partial z^{'}})_n (z_{n+1}^{'} - z_n^{'}) + (\frac{\partial \chi}{\partial z^{''}})_n (z_{n+1}^{''} - z_n^{''}) + ..., \end{aligned} \tag{1.26}$$

or

$$-(\frac{\partial g}{\partial z})_n (z_{n+1} - z_n) - (\frac{\partial g}{\partial z^{'}})_n (z_{n+1}^{'} - z_n^{'}) + (z_{n+1}^{''} - z_n^{''}) = 0. \tag{1.27}$$

Substituting $z_n^{''}$ from equation (1.25) into equation (1.27), we get

$$z_{n+1}^{''} - (\frac{\partial g}{\partial z^{'}})_n z_{n+1}^{'} - (\frac{\partial g}{\partial z})_n z_{n+1} = g(x, z_n, z_n^{'}) - (\frac{\partial g}{\partial z})_n z_n - (\frac{\partial g}{\partial z^{'}})_n z_n^{'}. \tag{1.28}$$

The boundary conditions are

$$z_{n+1} = 0, z_L = A. \tag{1.29}$$

1.7 CPU Time

The time of execution between the start and end of a given program is defined as CPU time or CPU Execution time. This is the time CPU is taking to compute the program, the routine of operating system included on program behalf execution and the other running programs and time waiting for I/O does not include in it.

1.8 FLOPS

FLOPS stands for floating-point operations per second. It is used to calculate the number of floating-point operations that process, device or a core is capable of performing within a second.

1.9 Band matrix

1.9.1 Bandwidth

Consider a matrix $T = (t_{i,j})$ of $n \times n$ order. If the elements except the diagonal are zero and the range of diagonally bordered band is determined by a_1 and a_2 which are constants

$$\left\{t_{i,j} = 0 \text{ if } j < i - a_1 \;\; or \;\; j > i + a_2; \;\; a_1, a_2 \geq 0\right\} \tag{1.30}$$

and the quantity a_1 is known as lower bandwidth and a_2 is upper bandwidth. The maximum of a_1 and a_2 is the matrix bandwidth.

Definition

Band matrix is defined as a matrix that has reasonably small bandwidth.

1.9.2 Examples

Examples of matrices are

1. Diagonal matrix
2. Tridiagonal matrix
3. Pentadiagonal matrix
4. Triangular matrices

1.9.3 Diagonal matrix

In bandwidth definition if $a_1 = a_2 = 0$ then the matrix is diagonal.
In MATLAB $A = \text{diag}(w)$ gives the square diagonal matrix which have the vector elements of w on the main diagonal.

1.9.4 Tridiagonal matrix:

If $a_1 = a_2 = 1$ then the matrix is triangular.
In MATLAB we can write tridiagonal matrix as

$$P = 5;$$

$$x = -1;$$

$$y = 4;$$

$$z = 2;$$

$$G = \text{diag(x} \times \text{ones(1,P))} + \text{diag(y} \times \text{ones(1,P-1),1)} + \text{diag(z} \times \text{ones(1,P-1),-1)} \quad (1.31)$$

1.9.5 Pentadiagonal matrix

In bandwidth definition if $a_1 = a_2 = 2$ and so on then the matrix is pentadiagonal matrix. In MATLAB we can write pentadiagonal matrix for $a_1 = a_2 = 2$ as

$$P = 5;$$

$$x = -1;$$

$$y = 4;$$

$$z = 2;$$

$$h = 4;$$

$$g = 9;$$

$$G = \text{diag(h*ones(1,P-2),2)} + \text{diag(a*ones(1,P))}+$$
$$\text{diag(y*ones(1,P-1),1)} + \text{diag(z*ones(1,P-1),-1)}+\text{diag(g*ones(1,P-2),-2)} \quad (1.32)$$

1.9.6 Triangular matrices

The upper and lower triangular matrices are define by

Upper triangular matrix

In bandwidth definition if $a_1 = 0, a_2 = n - 1$ is define as upper triangular matrix. In MATLAB it can be code as $b = triu(ones(5))$

Lower triangular matrix

In bandwidth definition if $a_1 = n - 1, a_2 = 0$ is define as lower triangular matrix. In MATLAB it can be code as $b = tril(ones(5))$

1.10 Sparse Matrix

A sparse matrix is defined as a matrix that contains very few non-zero components. In other words, in a $n \times n$ matrix where m is column and n is a row matrix the number of non-zero values is less than zero values such matrix is defined as a sparse matrix.

For example:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{1.33}$$

In 50x50 matrix contains only 5 nonzero components and all the other entries are zero.

1.11 LU Factorization

The LU-factorization of a nonsingular matrix A if it has upper-triangular U and lower-triangular L.

$A = LU$

For this form we can say A has LU decomposition. The uniqueness of this factorization (when it exists) does not exist.

1.11.1 Doolittle factorization

If diagonal elements of L are 1 then it is a Doolittle factorization.

$$\begin{bmatrix} \zeta_{11} & \zeta_{12} & \zeta_{13} & \cdots & \zeta_{1n} \\ \zeta_{21} & \zeta_{22} & \zeta_{23} & \cdots & \zeta_{2n} \\ \vdots \\ \zeta_{n1} & \zeta_{n2} & \zeta_{n3} & \cdots & \zeta_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

1.11.2 Crout factorization

If diagonal elements of U are 1 then it is a Crout factorization.

$$\begin{bmatrix} \zeta_{11} & \zeta_{12} & \zeta_{13} & \cdots & \zeta_{1n} \\ \zeta_{21} & \zeta_{22} & \zeta_{23} & \cdots & \zeta_{2n} \\ \vdots \\ \zeta_{n1} & \zeta_{n2} & \zeta_{n3} & \cdots & \zeta_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & 1 & u_{23} & \cdots & u_{2n} \\ \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

1.11.3 Cholesky Factorization

Cholesky factorization of a matrix A is defined as if the matrix is real, symmetric and also positive definite $A = U^T U$ where U represents the upper-triangular$(L = U^T)$

$$\begin{bmatrix} \zeta_{11} & \zeta_{12} & \zeta_{13} & \cdots & \zeta_{1n} \\ \zeta_{21} & \zeta_{22} & \zeta_{23} & \cdots & \zeta_{2n} \\ \vdots \\ \zeta_{n1} & \zeta_{n2} & \zeta_{n3} & \cdots & \zeta_{nn} \end{bmatrix} = \begin{bmatrix} u_{11} & 0 & 0 & 0 & 0 \\ u_{21} & u_{22} & 0 & \cdots & 0 \\ \vdots \\ u_{n1} & u_{n2} & u_{n3} & \cdots & u_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

LU factorization of a matrix A which is non-singular.

1. Change the linear system of equations into matrices form with A, B and X, where A represents augmented matrix, B and x shows constants and variable vectors respectively.

2. Consider $A = LU$, where U and L are upper triangular matrix and lower triangular matrix respectively also suppose that the diagonal entries are equal to 1.

3. To solve $y's$ consider $Ly = B$.

4. To solve variable vector x consider $Ux = y$, solve for the variable vectors x.

1.12 Thomas Algorithm

For the solution of a tridiagonal system of equation, we use the Thomas Algorithm or Tridiagonal Matrix Algorithm (TDMA) that is the simplified form of Gaussian elimination. For the system of n×n unknowns of the triangular system can be written as:

$$x_i v_{i1} + y_i v_i + z_i v_{i+1} = w_i \tag{1.34}$$

with the boundary conditions

$$v_0 = D_0 \tag{1.35}$$

$$v_{ns-1} = D_{ns-1} \tag{1.36}$$

The vector v and the coefficients x_i, y_i, z_i, w_i are unknown and the matrix can be express as

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ x_1 & y_1 & z_1 & & & \\ 0 & x_2 & y_2 & z_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \\ & & & x_{ns-2} & y_{ns-2} & z_{ns-2} \\ 0 & & & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{ns-3} \\ v_{ns-2} \\ v_{ns-1} \end{bmatrix} = \begin{bmatrix} D_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{ns-3} \\ w_{ns-2} \\ D_{ns-1} \end{bmatrix}$$

By applying boundary condition

$$\begin{bmatrix} y_1 & z_1 & & & \\ x_2 & y_2 & z_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \\ & & x_{ns-2} & y_{ns-2} & z_{ns-2} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{ns-3} \\ v_{ns-2} \end{bmatrix} = \begin{bmatrix} w_1 - a_1 D_0 \\ w_2 \\ \vdots \\ w_{ns-3} \\ w_{ns-2} z_{ns-2} D_{ns-1} \end{bmatrix}$$

The method eliminates the lower diagonal by the forward elimination and adjusts the RHS and upper diagonal.

Generally, the TDMA is written as

$$z_1^* = \begin{cases} \dfrac{z_1}{y_1} & ; i = 1 \\ \dfrac{z_1^*}{y_i - z_{i1}^* x_i} & ; i = 2, 3, ...ns - 3 \end{cases} \tag{1.37}$$

$$w_1^* = \begin{cases} \dfrac{w_1}{y_1} & ; i = 1 \\ \dfrac{w_i - w_{i1}^* x_i}{y_i - z_{i1}^* x_i} & ; i = 2, 3, ...ns - 2 \end{cases} \tag{1.38}$$

Now by the back substitution we obtain the solution

$$v_{ns-2} = w_{ns-2}^*, \qquad i = 2, 3, ...ns - 2 \tag{1.39}$$

$$v_i = w_i^* - z_i^* v_{i+1}, \qquad i = ns - 3, ns - 4, ...1 \tag{1.40}$$

1.13 bvp4c

bvp4c [22] is MATLAB built-in function which is an effective solver. It is used to solve the boundary values problems in MATLAB. To obtain the required accuracy it begins with the solution of a system of equations with the initial guess provided at initial mesh point and the step size is changed. It can help to reduce the error that comes in poor guessing for the BVPs solution.

Syntax

$$sol = bvp4c(odefun, bcfun, solinit)$$

$$solinit = bvpinit(x, yinit, params)$$

where $odefun$ is a function that deals with the differential equations f(x,y). Its form

$$\begin{aligned} dydx &= odefun(x, y) \\ dydx &= odefun(x, y, parameters) \end{aligned} \tag{1.41}$$

For a column vector y and scalar x, $odefun$ return a column vector, f(x,y) is represented by dydx. Unknown parameters are represented by parameters. The function $bcfun$ deals with the residual in boundary conditions. To deal with the two-point in boundary value condition which has the form bc(y(a),y(b)), form of $bcfun$ is

$$\begin{aligned} res &= bcfun(ya, yb) \\ res &= bcfun(ya, yb, parameters) \end{aligned} \tag{1.42}$$

where column vectors are ya and yb corresponding to y(a) and y(b). Unknown parameters are represented by parameters. The output res also represents the column vector. For a solution, the initial guess is contained in the form of structure solinit. The function bvpinit makes it manageable to form the guess structure.

$$solinit = bvpinit(x, yinit, parameters) \tag{1.43}$$

where x is the interval, yinit is initial guess.

Chapter 2

Finite difference method (FDM) for scalar ODEs

In this chapter, we solve the linear differential equation of second and third orders. Further steps are also discussed in this chapter after the simplification of differential equations by FDM. We will present the algorithm that is used for the solution of coupled ODEs.

2.1 Linear differential equation of second order:

FDM is used for the solution of the second-order differential equation. This method is reliable and gives accuracy to the results.

2.1.1 General second order ODE

Consider a linear differential equation of second-order [21]

$$\frac{d^2u}{dv^2} + A(v)\frac{du}{dv} + B(v)u = C(v). \tag{2.1}$$

and the boundary conditions for equation (2.1) is

$$u(0) = \alpha, \;\; u(L) = \delta.$$

The grid points for this equation as in finite-difference form are defined as

$$v_k = v_{k-1} + h, \;\; k = 1, 2, ..., M,$$

where $v_k = L$. and total number of intervals is represented by M.
Now u be the variable and derivatives of u at v_m are given by

$$u = u_k, \tag{2.2}$$

$$\frac{du}{dv} = \frac{u_{k+1} - u_{k-1}}{2h}, \tag{2.3}$$

$$\frac{d^2u}{dv^2} = \frac{u_{k+1} - 2u_k + u_{k-1}}{h^2}. \tag{2.4}$$

Now the equation (2.1) and boundary conditions of it become

$$\frac{u_{k+1} - 2u_k + u_{k-1}}{h^2} + A(v)\frac{u_{k+1} - u_{k-1}}{2h} + B(v)u_k = C(v), \tag{2.5}$$

or

$$a_k u_{k-1} + b_k u_n + c_k u_{k+1} = r_k, \tag{2.6}$$

$$a_k = 2 - hA(v_k), \qquad b_k = 2h^2B(v_k) - 4, \qquad c_k = 2 + hA(v_k)\ , \qquad r_k = 2h^2C(v_k). \tag{2.7}$$

It is written in matrix-vector form in compact form as

$$Au = s. \tag{2.8}$$

where

$$u = \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ u_{M-1} \end{bmatrix} \qquad s = \begin{bmatrix} s_1 \\ s_2 \\ . \\ . \\ s_{M-1} \end{bmatrix} = \begin{bmatrix} r_1 - \alpha a_1 \\ r_2 \\ . \\ . \\ r_{M-1} - \delta c_{M-1} \end{bmatrix} \tag{2.9}$$

The matrix A is tridiagonal matrix and is written in LU-Factorization as

$$A = LU. \tag{2.10}$$

where

$$L = \begin{bmatrix} \xi_1 & & & & & \\ a_2 & \xi_2 & & & & \\ & & .... & & & \\ & & & a_{M-2} & \xi_{M-2} & \\ & & & & a_{M-1} & \xi_{M-1} \end{bmatrix} \tag{2.11}$$

and

$$U = \begin{bmatrix} 1 & \gamma_1 & & & & \\ & 1 & \gamma_2 & & & \\ & & .... & & & \\ & & & & 1 & \gamma_{M-2} \\ & & & & & 1 \end{bmatrix} \tag{2.12}$$

where U is upper and L is lower triangular matrix. Here the unknowns (ξ_i, γ_i), $k = 1, 2, ..., M-1$ are to be related as

$$\xi_1 = -1 - \frac{\lambda}{h}, \quad \gamma_1 = \frac{\lambda}{\xi_1 h} \tag{2.13}$$

$$\xi_k = b_k - a_k\gamma_{k-1}, \quad k = 2, 3, ..., M-1 \tag{2.14}$$

$$\xi_k\gamma_k = c_k, \quad k = 2, 3, ..., M-2 \tag{2.15}$$

After defining these relations equation (2.8) becomes

$$LUu = s, \qquad Uu = z, \qquad \text{and} \quad Lz = s. \tag{2.16}$$

we have

$$\begin{bmatrix} \xi_1 & & & & \\ a_2 & \xi_2 & & & \\ & & .... & & \\ & & & a_{M-2} & \xi_{M-2} \\ & & & & a_{M-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ . \\ . \\ . \\ z_{M-2} \\ z_{M-1} \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ . \\ . \\ . \\ s_{M-2} \\ s_{M-1} \end{bmatrix} \tag{2.17}$$

we can find the unknown elements of z

$$z_1 = s_1/\xi_1, z_k = \frac{s_k - a_k z_{k-1}}{\xi_k}, k = 2, 3, ..., M-1, \tag{2.18}$$

and

$$\begin{bmatrix} 1 & \gamma_1 & & & \\ & 1 & \gamma_2 & & \\ & & .... & & \\ & & & 1 & \gamma_{M-2} \\ & & & & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ . \\ u_{M-2} \\ u_{M-1} \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ . \\ . \\ . \\ z_{M-2} \\ z_{M-1} \end{bmatrix}. \qquad (2.19)$$

We then get

$$u_{k-1} = z_{k-1}, \qquad u_k = z_k - \gamma_k u_{k+1}, \qquad k = M-2, M-3, ..., 3, 2, 1, \qquad (2.20)$$

which is a solution of equation (2.8).
As a summary,the solution of equation (2.1) involves the following steps.

1. Reduce the given differential equation to its corresponding finite difference form.
2. Compare with equation (2.6) to identify a_k,b_k,c_k, and r_k.
3. Calculate ξ_k and γ_k from equation (2.15)
4. Calculate z_k from equation (2.18)
5. Calculate y_k from equation (2.20) which is the solution we required.

2.2 Linear differential equations of third order:

The third order linear differential equation can be solved by FDM. For this purpose, the third order differential equation is changed into second order. The boundary condition also changes when we change the equation order.

2.2.1 General third order ODE

Consider a third order linear differential equation

$$\frac{d^3u}{dv^3} + A(v)\frac{d^2u}{dv^2} + B(v)\frac{du}{dv} + C(v)u = D(v). \qquad (2.21)$$

The boundary conditions of equation (2.21) are

$$u(0) = \alpha, \quad \frac{du(0)}{dv} = \epsilon, \quad \frac{du(L)}{dv} = \lambda. \tag{2.22}$$

These equation replace into two equation

$$\frac{du}{dv} = e, \frac{d^2e}{dv^2} = -A(v)\frac{de}{dv} - B(v)e - C(v)u + D(v). \tag{2.23}$$

The boundary conditions of equation (2.23) are

$$u(0) = \alpha, \;\; e(0) = \epsilon, \;\; e(L) = \lambda. \tag{2.24}$$

2.3 Thomas Algorithm for SFDM

Thomas algorithm is implemented in MATLAB to compute the solution G.
The tridiagonal matrix X can be written in LU-Factorization as

$$X = LU \tag{2.25}$$

where

$$L = \begin{bmatrix} \xi_1 & & & & & \\ a_2 & \xi_2 & & & & \\ & & .... & & & \\ & & & a_{M-2} & \xi_{M-2} & \\ & & & & a_{M-1} & \xi_{M-1} \end{bmatrix} \tag{2.26}$$

and

$$U = \begin{bmatrix} 1 & \gamma_1 & & & \\ & 1 & \gamma_2 & & \\ & & .... & & \\ & & & 1 & \gamma_{M-2} \\ & & & & 1 \end{bmatrix} \tag{2.27}$$

where U upper and L is lower triangular matrix, respectively. Here the unknowns $(\xi_k, \gamma_i), \quad k = 1, 2, ..., M-1$ are to be related as

$$\xi_1 = -1 - \frac{\lambda}{h}, \qquad \gamma_1 = \frac{\lambda}{\xi_1 h} \tag{2.28}$$

$$\xi_k = y_k - x_k\gamma_{k-1}, \qquad k = 2, 3, ..., M-1 \tag{2.29}$$

$$\xi_k\gamma_k = z_k, \qquad k = 2, 3, ..., M-2 \tag{2.30}$$

After defining these relations equation (2.25) becomes

$$LUG = s, \qquad UG = z, \qquad \text{and} \qquad Lz = s. \tag{2.31}$$

we have

$$\begin{bmatrix} \xi_1 & & & & \\ a_2 & \xi_2 & & & \\ & & .... & & \\ & & & a_{M-2} & \xi_{M-2} \\ & & & & a_{M-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ . \\ . \\ . \\ z_{M-2} \\ z_{M-1} \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ . \\ . \\ . \\ s_{M-2} \\ s_{M-1} \end{bmatrix} \tag{2.32}$$

we can find the unknown elements of z

$$z_1 = s_1/\xi_1, z_k = \frac{s_k - x_k z_{k-1}}{\xi_k}, k = 2, 3, ..., M-1, \tag{2.33}$$

and

$$\begin{bmatrix} 1 & \gamma_1 & & & \\ & 1 & \gamma_2 & & \\ & & .... & & \\ & & & 1 & \gamma_{N-2} \\ & & & & 1 \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ . \\ . \\ . \\ G_{M-2} \\ G_{M-1} \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ . \\ . \\ . \\ z_{M-2} \\ z_{M-1} \end{bmatrix}. \tag{2.34}$$

We then get

$$G_{k-1} = z_{k-1}, \qquad G_k = z_k - \gamma_k G_{k+1}, \qquad k = M-2, M-3, ..., 3, 2, 1, \tag{2.35}$$

Which gives solution of equation (2.25). $g' = G$ is in discretization form and we can find g easily

$$\frac{g_{k+1} - g_k}{h} = G_k \tag{2.36}$$

2.4 Numerical procedure of the simplified finite difference (SFDM) method

For the solution of coupled non-linear ODEs with the boundary conditions first, transfer the non-linear ODEs into linear ODEs of the first order. This purpose is fulfilled by the SFDM. The algorithm for the SFDM with the necessary details are as follows:

1. First we reduce the ODEs in the third order. The reduction forms second and first-order group ODEs.

2. For further process, the system of non-linear ODEs is linearized by the Taylor series.

3. Finite differences formulas are used to replace the derivatives in ODEs

4. Finally, we attain algebraic equations that Thomas algorithm can effectively solve.

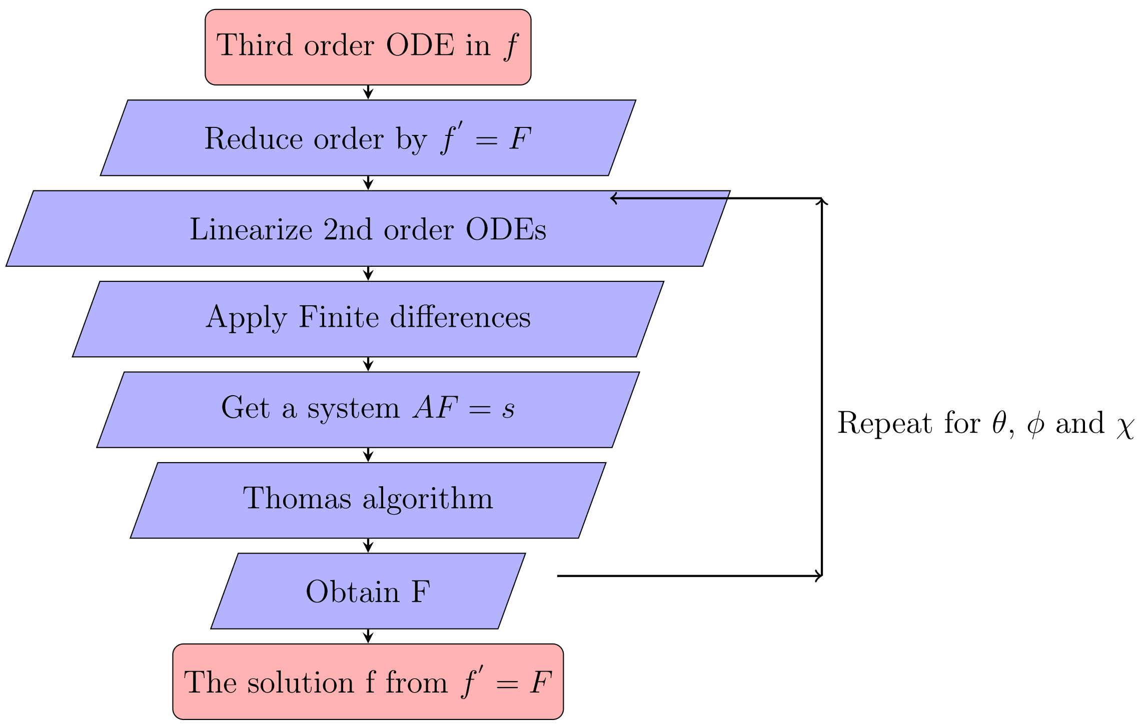

Figure 2.1: Flow chart of SFDM.

Chapter 3

Simplified finite difference method (SFDM) for two Coupled ODEs

SFDM is used to solve the coupled ODEs. In this chapter, we solve the two coupled ODEs and compare their results with other methods. Here we present some problems that appear in fluid mechanics.

Example 3.0.1.

Consider a two coupled ODEs [13]

$$g''' - g'^2 + gg'' + \epsilon^2 + M(\epsilon - g') = 0, \tag{3.1}$$

$$\theta'' - Pr_o(n\theta g' - g\theta') = 0. \tag{3.2}$$

along with the boundary conditions

$$\begin{gathered} g(\eta) = 0, g'(\eta) = 1, \theta(\eta) = 1, \quad as \quad \eta = 0 \\ g'(\eta) \to \epsilon, \theta(\eta) \to 0, \quad as \quad \eta \to \infty \end{gathered} \tag{3.3}$$

To initiate we assume $g^{'} = G$ in equation (3.1) then we get

$$\frac{d^2G}{d\eta^2} = G^2 - g\frac{dG}{d\eta} - \epsilon^2 - M(\epsilon - G) = 0. \tag{3.4}$$

This expression can be written for the function g as

$$\phi_1(\eta, G, G^{'}) = G^2 - g\frac{dG}{d\eta} - \epsilon^2 - M(\epsilon - G). \tag{3.5}$$

Let us approximate $\frac{dG}{d\eta}$ in the above equation by an approximation of the forward difference

$$\phi_1(\eta, G, G^{'}) = G^2 - g_k(\frac{G_{k+1} - G_k}{h}) - \epsilon^2 - M(\epsilon - G). \tag{3.6}$$

The second-order coefficients ODE read as

$$X_n = -\frac{\partial \phi_1}{\partial G'} = -(-g) = g = g_k, \tag{3.7}$$

$$Y_n = -\frac{\partial \phi_1}{\partial G} = -2G - M, \tag{3.8}$$

$$Y_n = -2G_k - M, \tag{3.9}$$

$$Z_n = \phi_1(\eta, G, G^{'}) + Y_n G_k + X_n \frac{G_{k+1} - G_k}{h}. \tag{3.10}$$

After some manipulation equation (3.10) becomes

$$x_k G_{k-1} + y_k G_k + z_k G_{k+1} = w_k, \qquad k = M, ...3, 2, 1, \tag{3.11}$$

where

$$x_k = -hX_n + 2, \qquad y_k = -4 + 2h^2 Y_n, \qquad z_k = hX_n + 2\ , \qquad w_k = 2h^2 Z_n. \tag{3.12}$$

The expression written in matrix-vector form

$$XG = p. \tag{3.13}$$

where

$$X = \begin{bmatrix} y_1 & z_1 & & & & \\ x_2 & y_2 & z_2 & & & \\ & & .... & & & \\ & & & x_{M-2} & y_{M-2} & z_{M-2} \\ & & & & x_{M-1} & y_{M-1} \end{bmatrix}, \tag{3.14}$$

$$G = \begin{bmatrix} G_1 \\ G_2 \\ . \\ . \\ G_{M-1} \end{bmatrix} \qquad s = \begin{bmatrix} p_1 \\ p_2 \\ . \\ . \\ p_{M-1} \end{bmatrix}. \tag{3.15}$$

The tridiagonal matrix X can be written in LU-Factorization as

$$X = LU. \tag{3.16}$$

by Thomas Algorithm we get

$$G_{k-1} = z_{k-1}, \qquad G_k = z_k - \gamma_k G_{k+1}, \qquad k = 1, 2, 3, ...M-3, M-2, \tag{3.17}$$

which is a solution of equation (4.31). From the discretization form of $f^{'} = F$ we can find f .

$$\frac{g_{k+1} - g_k}{h} = G_k \tag{3.18}$$

which gives a required solution of equation (3.1). A similar procedure may also be opting for θ.

$$\frac{d^2\theta}{d\eta^2} = Pr_o(n\theta G - g\frac{d\theta}{d\eta}).$$

$$\phi_2(\eta, \theta, \theta^{'}) = Pr_o(n\theta G - g\frac{d\theta}{d\eta}).$$

$$X_{nn} = \frac{\partial\phi_2}{\partial\theta'} = Pr_0 g, \tag{3.19}$$

$$X_{nn} = -\frac{\partial\phi_2}{\partial\theta'} = Pr_o g_k, \tag{3.20}$$

$$Y_{nn} = -\frac{\partial\phi_2}{\partial\theta} = -nPr_o G, \tag{3.21}$$

$$Y_{nn} = -\frac{\partial \phi_2}{\partial \theta} = -nPr_o G_k. \tag{3.22}$$

Table 3.1. $-f''(0)$, $-\theta'(0)$ comparison with bvp4c and SFDM for n=1 and $\epsilon = 0.1$

		bvp4c		CPU Time(sec)	SFDM		CPU Time(sec)	Absolute Error	
Pr	M	$-f''(0)$	$-\theta'(0)$	(bvp4c)	$-f''(0)$	$-\theta'(0)$	(SFDM)	$-f''(0)$	$-\theta'(0)$
0.7	0.1	1.009892	0.812049	2.476289	1.009868	0.827645	6.800476	0.000024	0.015596
1	-	1.009892	1.01274	2.351210	1.009868	1.016543	6.591892	0.000024	0.003803
3	-	1.009893	1.926745	2.338525	1.009868	1.919854	6.762179	0.000025	0.006891
7	-	1.009893	3.072613	2.586294	1.009868	3.058422	6.707079	0.000025	0.014191
10	-	1.009893	3.720453	2.392673	1.009868	3.700721	6.732499	0.000025	0.019732
0.7	0.2	1.048905	0.804591	2.297988	1.04823	0.8213916	7.076160	0.000675	0.016801
-	0.3	1.086569	0.797513	2.545248	1.08314	0.7996568	6.297520	0.00255	0.002014
-	0.4	1.12301	0.790780	2.391195	1.119332	0.7932026	6.207300	0.003678	0.002423
-	0.5	1.158335	0.784366	2.374771	1.154416	0.787055	6.077611	0.003919	0.002689

3.1 Results and discussion

In the attempt to check the accuracy of our purposed method SFDM, we have compared the time of execution of SFDM with bvp4c using built-in MATLAB routine tic-toc and also check the absolute error between SFDM and bvp4c. bvp4c is a built-in function and it takes less time than SFDM. The absolute error shows the error in SFDM as compare to bvp4c because SFDM is manual solver and bvp4c is a built-in function.

Chapter 4

Simplified finite difference method (SFDM) for three and four Coupled ODEs

In this chapter, we solve three and four coupled ODEs by SFDM and compare the results with other methods and results. This chapter shows that the SFDM is reliable for even three and four coupled ODEs.

4.1 SFDM for three coupled ODEs

Example 4.1.1.

Consider a third order coupled ODEs [14]

$$g^{'''} + gg^{''} - \frac{2n}{n+1}g^{'2} - K_p g^{'} + M(E_1 - g^{'}) = 0, \quad (4.1)$$

$$(1 + \frac{4}{3}R_d)p^{''} + Pr_o(N_b p^{'}\phi^{'} + gp^{'} + N_t(k^{'})^2 + \frac{2}{n+1}sp + ME_c(g^{'} - E_1)^2) = 0, \quad (4.2)$$

$$\chi^{''} + \frac{N_t}{N_b}p^{''} + Pr_o L_e g\chi^{'} = 0. \quad (4.3)$$

along with boundary conditions

$$g^{'}(\infty) = 0, \quad g'(0) = 1, \quad g(0) = \alpha(\frac{1-n}{1+n}),$$
$$k(\infty) = 0, \quad k^{'}(0) = B_i(\theta(0) - 1), \quad \chi(\infty) = 0, \quad N_b\chi^{'}(0) + N_t k^{'}(0) = 0 \quad (4.4)$$

Assume $g^{'} = G$ in equation (4.1), we may write

$$\frac{d^2G}{d\eta^2} = -g\frac{dG}{d\eta} + \frac{2n}{n+1}G^2 + K_pG - M(E_1 - G). \tag{4.5}$$

The function g can be expressed as

$$\phi_1(\eta, G, G^{'}) = -g\frac{dG}{d\eta} + \frac{2n}{n+1}G^2 + K_pG - M(E_1 - G). \tag{4.6}$$

In the above equation we can approximate $\frac{dG}{d\eta}$ using forward difference approximation

$$\phi_1(\eta, G, G^{'}) = -g_k(\frac{G_{k+1} - G_k}{h}) + \frac{2n}{n+1}G_k^2 + K_pG_k - M(E_1 - G_k). \tag{4.7}$$

The second order ODE coefficients can be read as

$$X_n = -\frac{\partial \phi_1}{\partial G^{'}} = -(-g) = g = g_k, \tag{4.8}$$

$$Y_n = -\frac{\partial \phi_1}{\partial G} = -(\frac{4n}{n+1}G + K_p + M) = -(\frac{4n}{n+1}G_k + K_p + M), \tag{4.9}$$

$$Z_n = \phi_1(\eta, G, G^{'}) + Y_nG_k + X_n\frac{G_{k+1} - G_k}{h}. \tag{4.10}$$

After some manipulation equation (4.10) becomes

$$x_kG_{k-1} + y_kG_k + z_kG_{k+1} = w_k, \qquad k = M, ...3, 2, 1, \tag{4.11}$$

where

$$x_k = -hX_n + 2, \quad y_k = -4 + 2h^2Y_n, \quad z_k = hX_n + 2\ , \quad w_k = 2h^2Z_n. \tag{4.12}$$

The expression written in matrix-vector form

$$XG = p. \tag{4.13}$$

where

$$X = \begin{bmatrix} y_1 & z_1 & & & & \\ x_2 & y_2 & z_2 & & & \\ & & .... & & & \\ & & & x_{M-2} & y_{M-2} & z_{M-2} \\ & & & & x_{M-1} & y_{M-1} \end{bmatrix}, \tag{4.14}$$

$$G = \begin{bmatrix} G_1 \\ G_2 \\ . \\ . \\ G_{M-1} \end{bmatrix} \qquad s = \begin{bmatrix} p_1 \\ p_2 \\ . \\ . \\ p_{M-1} \end{bmatrix}. \tag{4.15}$$

The tridiagonal matrix X can be written in LU-Factorization as

$$X = LU. \tag{4.16}$$

by Thomas Algorithm we get

$$G_{k-1} = z_{k-1}, \qquad G_k = z_k - \gamma_k G_{k+1}, \qquad k = 1, 2, 3, ...M-3, M-2, \tag{4.17}$$

which is a solution of equation (4.5). From the discretization form of $g^{'} = G$ we can find g .

$$\frac{g_{k+1} - g_k}{h} = G_k \tag{4.18}$$

gives a required solution of equation (4.1). A similar procedure may also be opting for p and χ solutions.

$$\frac{d^2p}{d\eta^2} = -(\frac{Pr_o}{(1+\frac{4}{3}R_d)}(N_b\frac{dp}{d\eta}\frac{d\chi}{d\eta} + g\frac{dp}{d\eta} + N_t(\frac{dp}{d\eta})^2 + \frac{2}{n+1}sk + ME_c(\frac{dg}{d\eta} - E_1)^2)) \tag{4.19}$$

$$\begin{aligned} \phi_2(\eta,\theta,\theta^{'}) = -(\frac{Pr_o}{(1+\frac{4}{3}R_d)}(N_b(\frac{p_k - p_{k-1}}{h})(\frac{\chi_k - \chi_{k-1}}{h}) + g_k(\frac{p_k - p_{k-1}}{h}) + \\ N_t(\frac{p_k - p_{k-1}}{h})^2 + \frac{2}{n+1}sp_k + ME_c(G_k - E_1)^2)), \end{aligned} \tag{4.20}$$

$$X_{nn} = -\frac{\partial \phi_2}{\partial p^{'}} = -(-\frac{Pr_o}{(1+\frac{4}{3}R_d)}(N_b\chi^{'} + g + 2N_t k^{'})), \tag{4.21}$$

$$X_{nn} = \frac{Pr_o}{(1+\frac{4}{3}R_d)}(N_b(\frac{\chi_k - \chi_{k-1}}{h}) + g_k + 2N_t\frac{p_k - p_{k-1}}{h})), \tag{4.22}$$

$$Y_{nn} = \frac{2sPr_o}{(1+4/3R_d)(n+1)}, \tag{4.23}$$

$$\frac{d^2\chi}{d\eta^2} = \frac{-N_t}{N_b}\frac{d^2p}{d\eta^2} - L_e Pr_o g\chi^{'} \tag{4.24}$$

$$Q(\eta, \chi, \chi^{'}) = \frac{-N_t}{N_b}\frac{p_{k-1} - 2p_k + p_{k+1}}{h^2} - L_e Pr_o(g_k\frac{\chi_k - \chi_{k-1}}{h}) \tag{4.25}$$

Similarly, the coefficients for equation (4.3) are written as

$$X_{nnn} = Pr_o L_e g_k, \qquad Y_{nnn} = 0. \tag{4.26}$$

Table 4.1. $-g^{''}(0)$ value comparison for various n values from literature and $\alpha = 0.25$

n	Fang et al. [12]	Khader and Ahmed [17]	(bvp4c)	CPU Time(sec)	(SFDM)	CPU Time(sec)
10	1.1433	1.1433	1.1433	3.243484	1.1433	5.489294
9	1.1404	1.1404	1.1404	3.238109	1.1404	4.421134
7	1.1323	1.1322	1.1323	1.244996	1.1323	4.901705
5	1.1186	1.1186	1.1186	3.2041311	1.1186	4.560193
3	1.0905	1.0904	1.0905	3.176654	1.0905	4.434601
1	1.0000	1.0000	1.0000	3.241500	1.0000	3.912580
0.5	0.9338	0.9337	0.9338	3.227510	0.9338	3.227510
0	0.7843	0.7843	0.7843	6.427043	0.7843	3.815771
-1/3	0.5000	0.5000	0.5000	3.227962	0.5024	4.916691

4.2 Results and discussion

For table 4.1 in the attempt to check the accuracy of our purposed method SFDM, we have compared the time of execution of SFDM with bvp4c using built-in MATLAB routine tic-toc and the results showed the accuracy of SFDM. SFDM took more time than bvp4c because it is a manual solver and bvp4c is a built-in function. The accuracy of SFDM results is good as compared with bvp4c.

4.3 Results and discussion

For table 4.2 in the attempt to check the accuracy of our purposed method SFDM, we have compared the results of bvp4c with SFDM. The results showed good accuracy and also we have compared the time of execution of SFDM with bvp4c using built-in MATLAB routine tic-toc. SFDM took more time than bvp4c because it is a manual solver and bvp4c is a built-in function. The accuracy of SFDM results is good as compared with bvp4c.

The results are computed for $N = 1000$ grid points in the η direction. However, the number of grid points varied in some calculations to achieve better accuracy.

Table 4.2. $-f^{''}(0)$ value comparison with bvb4c for different parameters M, n, α, E_1 and K_p.

M	n	α	E_1	K_p	$(-f^{''}(0))(bvp4c)$	CPU Time(sec)	$(-f^{''}(0))(SFDM)$	CPU Time(sec)
0	0.5	0.3	0.1	0.1	0.996308	1.708002	0.996308	4.532807
0.3					1.097247	1.681953	1.097247	2.762483
0.7					1.236298	1.593069	1.236298	3.458474
0.1	0				0.907889	1.796501	0.907889	3.583938
	0.5				1.025923	1.641161	1.025923	2.462961
	1				1.078835	1.675789	1.078835	3.275369
	0.5	0.4			1.043448	2.579608	1.043448	3.233970
		0.7			1.097515	2.332225	1.097515	3.347940
		1			1.153791	2.328705	1.153791	3.406247
		0.3	0.5		0.954581	2.292361	0.954581	3.308675
			1		0.877466	2.2321733	0.877466	3.308496
			1.5		0.807036	2.456861	0.807036	3.308675
			0.1	0.1	1.025923	2.248751	1.025923	3.879703
				0.3	1.12657	2.274022	1.12657	3.307774
				0.5	1.216757	2.241876	1.216757	3.256483

Example 4.3.1.

Consider a third order coupled ODEs [15]

$$f^{'''} + Nf^{'''} - N\lambda(\frac{n+1}{2})f^{''2}f^{'''} - \frac{2n}{n+1}f^{'2} + ff^{''} + M(E - f^{'}) - K_pf^{'} + G_r\theta = 0, \tag{4.27}$$

$$(1 + \frac{4}{3}R_d)\theta^{''} + P_{r_o}(N_b\theta^{'}\phi^{'} + f\theta^{'} + +N_t(\theta^{'})^2 + ME_c(f^{'} - E)^2 + \frac{2}{n+1}s\theta) = 0, \tag{4.28}$$

$$\phi^{''} + \frac{N_t}{N_b}\theta^{''} + L_ef\phi^{'} = 0. \tag{4.29}$$

along with boundary conditions

$$f(0) = \alpha(\frac{1-n}{1+n}), \;\; f'(0) = 1, \;\; f^{'}(\infty) = 0, \;\; \theta(0) = 1,$$
$$\theta(\infty) = 0, \;\; N_b\phi^{'}(0) + N_t\theta^{'}(0) = 0, \;\; \phi(\infty) = 0. \tag{4.30}$$

Assume $f^{'} = F$ in equation (4.27), we may write

$$\frac{d^2F}{d\eta^2} = (\frac{1}{1 + N - N\lambda(\frac{n+1}{2})(\frac{dF}{d\eta})^2})(\frac{2n}{n+1}F^2 - f\frac{dF}{d\eta} - M(E - F) + K_pF - G_r\theta). \tag{4.31}$$

Then writing this expression as

$$\chi_1(h,F,F^{'}) = (\frac{1}{1+N-N\lambda(\frac{n+1}{2})(\frac{dF}{d\eta})^2})(\frac{2n}{n+1}F^2 - f\frac{dF}{d\eta} - M(E-F) + K_pF - G_r\theta), \tag{4.32}$$

and replace $\frac{dF}{d\eta}$ by forward difference approximation

$$\begin{aligned}\chi_1(h,F,F^{'}) = (\frac{1}{1+N-N\lambda(\frac{n+1}{2})(\frac{F_{k+1}-F_k}{h})^2})(\frac{2n}{n+1}F_k^2 - f_k(\frac{F_{k+1}-F_k}{h}) - \\ M(E-F_k) + K_pF_k) - G_r\theta.\end{aligned} \tag{4.33}$$

The coefficients of second order ODE read

$$\begin{aligned} X_n = -\frac{\partial\chi_1}{\partial F'} = (\frac{1}{(1+N-N\lambda(\frac{n+1}{2})(\frac{dF}{d\eta})^2)^2})[(\lambda N(n+1)](\frac{-2n}{n+1}F^2 + \\ f\frac{dF}{d\eta} + M(E-F) - K_p\frac{\theta_r}{\theta_r-\theta}F + G_r\theta] - f(1+N-N\lambda(\frac{n+1}{2})(\frac{dF}{d\eta})^2)] = \\ ((\frac{1}{1+N-N\lambda(\frac{n+1}{2})(\frac{F_{k+1}-F_k}{h})^2)})[\lambda N(n+1)](\frac{-2n}{n+1}F^2 + \\ f\frac{F_{k+1}-F_k}{h} + M(E-F) - K_pF + G_r\theta) - f_k(1+N-N\lambda(\frac{n+1}{2})(\frac{F_{k+1}-F_k}{h}))], \end{aligned} \tag{4.34}$$

$$\begin{aligned} Y_n = -\frac{\partial\chi_1}{\partial F} &= -(\frac{1}{1+N-N\lambda(\frac{n+1}{2})(\frac{dF}{d\eta})^2})(\frac{4n}{n+1}F + M + K_p) \\ &= -(\frac{1}{1+N-N\lambda(\frac{n+1}{2})(\frac{F_{k+1}-F_k}{h})^2})(\frac{4n}{n+1}F_k + M + K_p), \end{aligned} \tag{4.35}$$

$$Z_n = \chi_1(h,F,F^{'}) + Y_nF_k + X_n\frac{F_{k+1}-F_k}{h}. \tag{4.36}$$

After manipulation in equations (4.34)-(4.36) the linear algebraic system in F are written as

$$x_kF_{k-1} + y_kF_k + z_kF_{k+1} = w_k, \qquad k = M, ...3, 2, 1, \tag{4.37}$$

where

$$x_k = -hX_n + 2, \quad y_k = -4 + 2h^2Y_n, \quad z_k = hX_n + 2\ , \quad w_k = 2h^2Z_n. \tag{4.38}$$

The expression written in matrix-vector form

$$XF = p. \tag{4.39}$$

where

$$X = \begin{bmatrix} y_1 & z_1 & & & & \\ x_2 & y_2 & z_2 & & & \\ & & .... & & & \\ & & & x_{M-2} & y_{M-2} & z_{M-2} \\ & & & & x_{M-1} & y_{M-1} \end{bmatrix}, \tag{4.40}$$

$$F = \begin{bmatrix} F_1 \\ F_2 \\ . \\ . \\ F_{M-1} \end{bmatrix} \qquad s = \begin{bmatrix} p_1 \\ p_2 \\ . \\ . \\ p_{M-1} \end{bmatrix}. \tag{4.41}$$

The tridiagonal matrix X can be written in LU-Factorization as

$$X = LU. \tag{4.42}$$

by Thomas Algorithm we get

$$F_{k-1} = z_{k-1}, \qquad F_k = z_k - \gamma_k F_{k+1}, \qquad k = 1, 2, 3, ...M-3, M-2, \tag{4.43}$$

which is a solution of equation (4.31). From the discretization form of $f' = F$ we can find f .

$$\frac{f_{k+1} - f_k}{h} = F_i \tag{4.44}$$

which gives a required solution of equation (4.27). A similar procedure may also be opting for θ and ϕ.

$$\frac{d^2\theta}{d\eta^2} = -\frac{Pr_o}{(1+\frac{4}{3}R_d)}(N_b\frac{d\theta}{d\eta}\frac{d\phi}{dh} + f\frac{d\theta}{d\eta} + N_t(\frac{d\theta}{d\eta})^2 + ME_c(\frac{df}{d\eta} - E)^2 + \frac{2}{n+1}s\theta).$$

$$\chi_2(h,\theta,\theta^{'}) = -\frac{Pr_o}{(1+\frac{4}{3}R_d)}(N_b(\frac{\theta_k-\theta_{k-1}}{h})(\frac{\phi_k-\phi_{k-1}}{h})+f_k(\frac{\theta_k-\theta_{k-1}}{h})+N_t(\frac{\theta_k-\theta_{k-1}}{h})^2+ ME_c(F_k-E)^2+\frac{2}{n+1}s\theta_k).$$

$$X_{nn} = -\frac{\partial\chi_2}{\partial\theta'} = -(-\frac{Pr_o}{(1+\frac{4}{3}R_d)}(N_b\phi^{'}+f+2N_t\theta^{'})), \tag{4.45}$$

$$X_{nn} = \frac{Pr_o}{(1+\frac{4}{3}R_d)}(N_b(\frac{\phi_k-\phi_{k-1}}{h})+f_k+2N_t\frac{\theta_k-\theta_{k-1}}{h})), \tag{4.46}$$

$$Y_{nn} = \frac{Pr_o}{(1+\frac{4}{3}R_d)}\frac{2}{n+1}s, \tag{4.47}$$

$$Y_{nn} = \frac{Pr_o}{(1+\frac{4}{3}R_d)}\frac{2}{n+1}s. \tag{4.48}$$

$$\frac{d^2\phi}{d\eta^2} = \frac{-N_t}{N_b}\frac{d^2\theta}{d\eta^2} - L_e Pr_o f\phi^{'}. \tag{4.49}$$

$$\chi_3(h,\phi,\phi^{'}) = \frac{-N_t}{N_b}\frac{\theta_{k-1}-2\theta_k+\theta_{k+1}}{h^2} - L_e Pr_o(f_k\frac{\phi_k-\phi_{k-1}}{h}). \tag{4.50}$$

Similarly, the coefficeints for equation (4.29) are

$$A_{nnn} = Pr_o L_e f_k, \qquad B_{nnn} = 0. \tag{4.51}$$

Table 4.3. $-f''(0)$ values comparison for various n values from literature and $\alpha = 0.25$

n	Fang et al. [12]	Daniel et al. [16]	Present result (SFDM)	CPU Time (sec)
10	1.1433	1.143316	1.143301	4.248827
9	1.1404	1.140388	1.140431	4.317216
7	1.1323	1.132281	1.132301	4.218688
5	1.1186	1.118587	1.118602	4.207821
3	1.0905	1.090490	1.090400	4.488553
1	1.0000	1.000001	1.000009	4.279683
0.5	0.9338	0.933828	0.933796	4.206902
0	0.7843	0.784284	0.784330	4.240561
-1/3	0.5000	0.500000	0.501889	4.497585
-0.5	0.0833	0.083289	0.086736	4.256302

4.4 Results and discussion

In the attempt to check the accuracy of our purposed method SFDM, we have checked the time of execution of SFDM using built-in MATLAB routine tic-toc. The results showed the accuracy and it took 4.3 second approximate CPU time.

4.5 SFDM for four coupled ODEs

SFDM for the solution of four coupled ODEs.

Example 4.5.1.

Consider a four coupled ODEs [20]

$$g''' + gg'' - (g')^2 + 1 + B[1 - (\frac{1}{2}g''\eta + g')] = 0, \tag{4.52}$$

$$p'' + Pr_o gp' + N_b\phi' p' + N_t p'^2 - Pr_o B\frac{1}{2}p'\eta = 0, \tag{4.53}$$

$$\psi'' + \frac{N_t}{N_b}p'' + L_e g Pr_o \psi' - Pr_o B\frac{1}{2}\psi'\eta = 0, \tag{4.54}$$

$$\chi'' + L_b Pr_o g\chi' - P_e[\chi\psi'' + \psi'\chi'] - L_b Pr_o B\frac{1}{2}\chi'\eta = 0. \tag{4.55}$$

along with the boundary conditions

$$\begin{gathered} g(\eta) = 0, g'(\eta) = \epsilon_1, p(\eta) = 1, \psi(\eta) = 1, \chi(\eta) = 1, \quad as \quad \eta = 0 \\ \\ g'(\eta) \to 0, p(\eta) \to 0, \psi(\eta) \to 0, \chi(\eta) \to 0, \quad as \quad \eta \to \infty \end{gathered} \tag{4.56}$$

Assume $g^{'} = G$ in equation (4.52), we may write

$$\frac{d^2G}{d\eta^2} = -g\frac{dG}{d\eta} + (G)^2 - 1 - B[1 - (\frac{1}{2}\frac{dG}{d\eta}\eta + G)]. \tag{4.57}$$

This expression can be written for the function g as

$$f(\eta, G, G^{'}) = -g\frac{dG}{d\eta} + (G)^2 - 1 - B[1 - (\frac{1}{2}\frac{dG}{d\eta}\eta + G)], \tag{4.58}$$

Let us approximate $\frac{dG}{d\eta}$ by forward difference approximation in above equation

$$f_1(\eta, G, G^{'}) = -g_k(\frac{G_{k+1} - G_k}{h}) + G_k^2 B[1 - (\frac{1}{2}(\frac{G_{k+1} - G_k}{h})\eta + G)]. \tag{4.59}$$

The second-order coefficients ODE read as

$$X_n = -\frac{\partial f_1}{\partial G'} = -(-g + \frac{B}{2}\eta) = g - \frac{B}{2}\eta = g_k - \frac{B}{2}\eta, \tag{4.60}$$

$$Y_n = -\frac{\partial f_1}{\partial G} = -2G - B, \tag{4.61}$$

$$Y_n = -2G_k - B, \tag{4.62}$$

$$Z_n = f_1(\eta, G, G^{'}) + Y_n G_k + X_n \frac{G_{k+1} - G_k}{h}. \tag{4.63}$$

After some manipulation equation (4.63) becomes

$$x_k G_{k-1} + y_k G_k + z_k G_{k+1} = w_k, \qquad k = M, ...3, 2, 1, \tag{4.64}$$

where

$$x_k = -hX_n + 2, \quad y_k = -4 + 2h^2 Y_n, \quad z_k = hX_n + 2\ , \quad w_k = 2h^2 Z_n. \tag{4.65}$$

The expression written in matrix-vector form

$$XG = p. \tag{4.66}$$

where

$$X = \begin{bmatrix} y_1 & z_1 & & & & \\ x_2 & y_2 & z_2 & & & \\ & & .... & & & \\ & & & x_{M-2} & y_{M-2} & z_{M-2} \\ & & & & x_{M-1} & y_{M-1} \end{bmatrix}, \tag{4.67}$$

$$G = \begin{bmatrix} G_1 \\ G_2 \\ . \\ . \\ G_{M-1} \end{bmatrix} \qquad s = \begin{bmatrix} p_1 \\ p_2 \\ . \\ . \\ p_{M-1} \end{bmatrix}. \tag{4.68}$$

The tridiagonal matrix X can be written in LU-Factorization as

$$X = LU. \tag{4.69}$$

Now solve by Thomas Algorithm we get

$$G_{k-1} = z_{k-1}, \qquad G_k = z_k - \gamma_k G_{k+1}, \qquad k = M-2, M-3, ..., 3, 2, 1, \tag{4.70}$$

which is a solution of (4.57). From the discretization form of $g^{'} = G$ we can find g .

$$\frac{g_{k+1} - g_k}{h} = G_k \tag{4.71}$$

gives a required solution of equation (4.52). A similar procedure may also be opting for p, χ and ψ.

$$\frac{d^2p}{d\eta^2} = -Pr_o g\frac{dp}{d\eta} - N_b\psi'\frac{dp}{d\eta} - N_t(\frac{dp}{d\eta})^2 + Pr_o B\frac{1}{2}\frac{dp}{d\eta}\eta$$

$$f_2(\eta, p, p^{'}) = -Pr_o g\frac{p_k - p_{k-1}}{h} - N_b\psi'\frac{p_k - p_{k-1}}{h} - N_t(\frac{p_k - p_{k-1}}{h})^2 + Pr_o B\frac{1}{2}\frac{p_k - p_{k-1}}{h}\eta$$

$$X_{nn} = -\frac{\partial f_2}{\partial p^{'}} = Pr_o g + N_b\frac{d\psi}{d\eta} + 2N_t\frac{dp}{d\eta} - Pr_o B\frac{1}{2}\eta, \tag{4.72}$$

$$X_{nn} = -\frac{\partial f_2}{\partial p^{'}} = Pr_o g + N_b\frac{\psi_k - \psi_{k-1}}{h} + 2N_t\frac{p_k - p_{k-1}}{h} - Pr_o B\frac{1}{2}\eta, \tag{4.73}$$

$$Y_{nn} = 0, \tag{4.74}$$

$$Y_{nn} = 0. \tag{4.75}$$

$$f_3(\eta, \psi, \psi^{'}) = -\frac{N_t}{N_b}p'' - L_e g Pr_o\psi' + Pr_o B\frac{1}{2}\psi'\eta. \tag{4.76}$$

$$f_3(\eta, \psi, \psi^{'}) == -\frac{N_t}{N_b}\frac{p_{k-1} - 2p_k + p_{k+1}}{h^2} - L_e g Pr_o\frac{\psi_k - \psi_{k-1}}{h} + Pr_o B\frac{1}{2}\frac{\psi_k - \psi_{k-1}}{h}\eta \tag{4.77}$$

Similarly, the coefficeints for equation (4.54) are written as

$$X_{nnn} = L_e g_k Pr_o - Pr_o B\frac{1}{2}\psi'\eta, \qquad Y_{nnn} = 0. \tag{4.78}$$

$$f_4(\eta, \chi, \chi^{'}) = -L_b Pr_o g\chi' + P_e[\chi\psi'' + \psi'\chi'] + L_b Pr_o B\frac{1}{2}\chi'\eta. \tag{4.79}$$

$$f_4(\eta, \chi, \chi^{'}) = -L_b Pr_o g\frac{\chi_k - \chi_{k-1}}{h} + P_e[\chi\frac{\psi_{k-1} - 2\psi_k + \psi_{k+1}}{h^2} + \frac{\psi_k - \psi_{k-1}}{h}\frac{\chi_k - \chi_{k-1}}{h}] + L_b Pr_o B\frac{1}{2}\frac{\chi_k - \chi_{k-1}}{h}\eta. \tag{4.80}$$

Similarly, the coefficeints for equation (4.55) are written as

$$X_4 = L_b Pr_o g_k Pr_o - L_b Pr_o B\frac{1}{2}\eta - P_e\frac{\psi_k - \psi_{k-1}}{h}, \qquad Y_4 = -P_e\frac{\psi_{k-1} - 2\psi_k + \psi_{k+1}}{h^2}. \tag{4.81}$$

Table 4.4. $g''(0), p'(0)$ and $-\psi'(0)$ comparison when $P_e = 1$,$\epsilon_1 = 1$,$L_e = 2$,B $= 0$, $N_t = N_b = 0.5$, and $Pr_o = 1$ and also set and B $= 1$.

	Ibrahim et al.[18]	Zaimi et al.[19]	Naganthran et al.[20]	SFDM	CPU Time(sec)
$G'(0)$	0	0	0	0	4.872384
$k'(0)$	0.4767	0.474737	0.476737	0.4626	4.872384
$\psi'(0)$	1.0452	1.045154	1.045154	1.0956	4.872384

4.6 Results and discussion

The results by the simplified difference method show accuracy when compare with other methods. The application of this method is easy and reliable for a different number of coupled equations.

Chapter 5

Summary

The main purpose of this research is the solution of coupled ODEs. Chapter 1 of the thesis consists of some basic definitions and introduction of different methods that are used for the approximation of the differential equations like FDM, FEM, FVM and spectral methods. Defining the quasilinearzation for the transformation of non-linear coupled ordinary differential equations into linear differential equations. Thomas algorithm and bvp4c are defined.

In chapter 2, the method is defined for the solution of general second and third-order linear differential equations by FDM. The Thomas algorithm for SFDM is described. The Numerical procedure of SFDM is explained which is used in the next chapters for the solution of coupled ODEs.

In chapter 3, we worked on two coupled ODEs and solved it by SFDM and compared the result with bvp4c. We check the accuracy of SFDM by comparing it with bvp4c results. The CPU time of SFDM is also checked by comparing it with bvp4c. The results are accurate.

In chapter 4, we solve three and four coupled ODEs with SFDM. We also checked its CPU time of SFDM. The results show accuracy when compare with other methods and show reliability when it works with more number of coupled equations.

The results are accurate and the method is reliable because it can be used for a different number of coupled ODEs and give an accurate result.

Bibliography

[1] Abarbanel, Saul, Adi Ditkowski, and Bertil Gustafsson. "On error bounds of finite difference approximations to partial differential equationsâĂŤtemporal behavior and rate of convergence." Journal of Scientific Computing 15, no. 1 (2000): 79-116.

[2] Mickens, R. E. "Analytical and numerical study of a non-standard finite difference scheme for the unplugged van der Pol equation." Journal of sound and vibration 245, no. 4 (2001): 757-761.

[3] Farjadpour, Ardavan, David Roundy, Alejandro Rodriguez, Mihai Ibanescu, Peter Bermel, John D. Joannopoulos, Steven G. Johnson, and Geoffrey W. Burr. "Improving accuracy by subpixel smoothing in the finite-difference time domain." Optics letters 31, no. 20 (2006): 2972-2974.

[4] McGee, Shelly, and Padmanabhan Seshaiyer. "Finite difference methods for coupled flow interaction transport models." In Electron. J. Differ. Equ. Conf, vol. 17, pp. 171-184. 2009.

[5] Dolicanin, C. B., V. B. Nikolic, and D. C. Dolicanin. "Application of finite difference method to study of the phenomenon in the theory of thin plates." Scientific Publications of the State University of Novi Pazar 2, no. 1 (2010): 29-43.

[6] Mehra, Mani, Nutan Patel, and Rahul Kumar. "Comparison between different numerical methods for discretization of PDEsâĂŘA short review." In AIP Conference Proceedings, vol. 1281, no. 1, pp. 599-602. American Institute of Physics, 2010.

[7] Chambolle, Antonin, Stacey E. Levine, and Bradley J. Lucier. "An upwind finite-difference method for total variationâĂŞbased image smoothing." SIAM Journal on Imaging Sciences 4, no. 1 (2011): 277-299.

[8] I. SÃijngÃij and H. Demir, "Application of the Hybrid Differential Transform Method to the Nonlinear Equations," Applied Mathematics, Vol. 3 No. 3, 2012, pp. 246-250.

[9] Izadian, Jalal, Nasibeh Ranjbar, and Maryam Jalili. "The generalized finite difference method for solving elliptic equation on irregular mesh." World Appl. Sci. J.(ISSN 1818 (2013): 95-100.

[10] Lakshmi, R., and M. Muthuselvi. "Numerical solution for boundary value problem using finite difference method." Int. J. Innovative Res. Sci., Eng. Technol 2 (2013): 5305-5313.

[11] Gulkac, Vildan. "An Implicit Finite-Difference Method for Solving the Heat-Transfer Equation." International Journal of Scientific and Innovative Mathematical Research 3 (2015): 39-44

[12] Fang, Tiegang, Ji Zhang, and Yongfang Zhong. "Boundary layer flow over a stretching sheet with variable thickness." Applied Mathematics and Computation 218, no. 13 (2012): 7241-7252.

[13] Zobia Wajid. "Numerical study of boundary layer flow with variable fluid properties on a nonlinear and exponentially stretching sheet." MS Thesis, 2017

[14] Muhammad Irfan, Muhammad Asif Farooq, and Tousif Iqra. "A New Computational Technique Design for EMHD Nanofluid Flow Over a Variable Thickness Surface With Variable Liquid Characteristics." Frontiers in Physics 8 (2020): 66.

[15] Muhammad Irfan, Muhammad Asif Farooq, and Tousif Iqra. ”A Simplified Finite Difference Method for EMHD Powell-Eyring Nanofluid Flow over a Variable Thicked Surface and Variable Liquid Properties”(Submitted) .

[16] Daniel YS, Aziz ZA, Ismail Z, Salah F. Impact of thermal radiation on electrical MHD flow of nanofluid over nonlinear stretching sheet with variable thickness. Alexandria Engineering Journal. 2018 Sep 1;57(3): 2187-97.

[17] Khader, M. M., and Ahmed M. Megahed. "Numerical solution for boundary layer flow due to a nonlinearly stretching sheet with variable thickness and slip velocity." The European physical journal plus 128, no. 9 (2013): 100.

[18] Ibrahim, Wubshet, Bandari Shankar, and Mahantesh M. Nandeppanavar. "MHD stagnation point flow and heat transfer due to nanofluid towards a stretching sheet." International journal of heat and mass transfer 56, no. 1-2 (2013): 1-9.

[19] Zaimi, Khairy, Anuar Ishak, and Ioan Pop. "Stagnation-point flow toward a stretching/shrinking sheet in a nanofluid containing both nanoparticles and gyrotactic microorganisms." Journal of heat transfer 136, no. 4 (2014): 2-8

[20] Naganthran, Kohilavani, Md Faisal Md Basir, Sayer Obaid Alharbi, Roslinda Nazar, Anas M. Alwatban, and Iskander Tlili. "Stagnation point flow with time-dependent bionanofluid past a sheet: Richardson extrapolation technique." Processes 7, no. 10 (2019): 722.

[21] Na, Tsung Yen, ed. Computational methods in engineering boundary value problems. Academic Press, 1980.

[22] L F Shampine, J Kierzenka and M W Reichelt Available at http://www.mathsworks.com (2003)