UNIVERSITÀ DELL'INSUBRIA
DIPARTIMENTO DI SCIENZA E ALTA TECNOLOGIA

DOCTORAL THESIS

Numerical Iterative Methods For Nonlinear Problems

Author:
Malik Zaka Ullah

Supervisor:
Prof. Stefano Serra-Capizzano

Thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the program

Mathematics of Computing: Models, Structures,
Algorithms, and Applications

June 2015

UNIVERSITÀ DELL'INSUBRIA
DIPARTIMENTO DI SCIENZA E ALTA TECNOLOGIA

DOCTORAL THESIS

Numerical Iterative Methods For Nonlinear Problems

Thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the program

Mathematics of Computing: Models, Structures,
Algorithms, and Applications

MALIK ZAKA ULLAH

Supervisor & Coordinator: Prof. Stefano Serra-Capizzano

Signed: .....

June 2015

UNIVERSITÀ DELL'INSUBRIA
DIPARTIMENTO DI SCIENZA E ALTA TECNOLOGIA

Doctor of Philosophy

Numerical Iterative Methods For Nonlinear Problems

by Malik Zaka Ullah

Abstract

The primary focus of research in this thesis is to address the construction of iterative methods for nonlinear problems coming from different disciplines. The present manuscript sheds light on the development of iterative schemes for scalar nonlinear equations, for computing the generalized inverse of a matrix, for general classes of systems of nonlinear equations and specific systems of nonlinear equations associated with ordinary and partial differential equations. Our treatment of the considered iterative schemes consists of two parts: in the first called the 'construction part' we define the solution method; in the second part we establish the proof of local convergence and we derive convergence-order, by using symbolic algebra tools. The quantitative measure in terms of floating-point operations and the quality of the computed solution, when real nonlinear problems are considered, provide the efficiency comparison among the proposed and the existing iterative schemes. In the case of systems of nonlinear equations, the multi-step extensions are formed in such a way that very economical iterative methods are provided, from a computational viewpoint. Especially in the multi-step versions of an iterative method for systems of nonlinear equations, the Jacobians inverses are avoided which make the iterative process computationally very fast. When considering special systems of nonlinear equations associated with ordinary and partial differential equations, we can use higher-order Frechet derivatives thanks to the special type of nonlinearity: from a computational viewpoint such an approach has to be avoided in the case of general systems of nonlinear equations due to the high computational cost. Aside from nonlinear equations, an efficient matrix iteration method is developed and implemented for the calculation of weighted Moore-Penrose inverse. Finally, a variety of nonlinear problems have been numerically tested in order to show the correctness and the computational efficiency of our developed iterative algorithms.

Acknowledgements

In the name of Almighty Lord, Who created for us the Earth as a habitat place and the sky as a marquee, and has given us appearance and made us good-looking and has furnished us with good things.

I would like to deliberate my special gratitude and recognition to my Advisor Professor Dr. Stefano Serra Capizzano, who has been a fabulous mentor for me. I would like also to thank Prof. Abdulrahman Labeed Al-Malki, Prof. Abdullah Mathker Alotaibi, Prof. Mohammed Ali Alghamdi, Dr. Eman S. Al-Aidarous, Dr. Marco Donatelli, Dr. Fazlollah Soleymani, and Mr. Fayyaz Ahmad for their moral support and advices to pursue my PhD.

A special thank all of my family and to the friends who supported me in writing, and pushed me to strive towards my goals. In the end, I would like express appreciation to my beloved wife, who always supported me also in the dark moments of discouragement.

Contents

List of Figures

List of Tables

To my family

Chapter 1

Introduction

Nonlinear problems arise in diverse areas of engineering, mathematics, physics, chemistry, biology, etc., when modelling several types of phenomena. In many situations, the nonlinear problems naturally appear in the form of nonlinear equations or systems of nonlinear equations. For instance a standard second order centered Finite Difference discretization of a nonlinear boundary-value problem of the form

$$y'' + y^2 = cos(x)^2 + cos(x), \quad y(0) = -1,\ y(\pi) = 1 \tag{1.1}$$

produces the following system of nonlinear equations

$$y_{i+1} - 2y_i + y_{i-1} + h^2 y_i^2 = h^2(cos(x_i)^2 + cos(x_i)), \quad i \in \{1, 2, \cdots, n\}. \tag{1.2}$$

In real applications, finding the solution of nonlinear equations or systems of equations has enough motivation for researchers to develop new computationally efficient iterative methods. The analytical solution of most types of nonlinear equations or systems of nonlinear equations is not possible in close form, and the role of numerical methods becomes crystal clear. For instance, the solution of general quintic equation can not be expressed algebraically which is demonstrated by Abel's theorem [1]. In general it is not always possible to get the analytical solution for linear or nonlinear problems and hence numerical iterative methods are best suited for the purpose. The work of Ostrowski [2] and Traub [3] provides the necessary basic treatment of the theory of iterative methods for solving nonlinear equations. Traub [3] divided the numerical iterative methods into two classes namely one-point iterative methods and multi-point iterative methods. A further classification divides the aforementioned iterative methods into two

sub-classes: One-Point iterative methods with and without memory and multi-point iterative methods with and without memory. The formal definitions of aforementioned classification are given in the following.

We can describe the iterative methods with help of iteration functions. For instance in the Newton method

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

the iteration function is $\phi(x_i) = f(x_i)/f'(x_i)$. In fact Newton method is a one-pint method. In general, if x_{i+1} is determined only by the information at x_i and no older information is reused then the iterative method

$$x_{i+1} = \phi(x_i),$$

is called one-point method and ϕ is called one-point iteration function. If x_{i+1} is determined by some information at x_i and reused information at $x_{i-1}, x_{i-2} \cdots, x_{i-n}$ i.e.

$$x_{i+1} = \phi(x_i; x_{i-1}, \cdots, x_{i-n}),$$

then the iterative method is called one-point iterative method with memory. In this case the iteration function ϕ is an iteration function with memory. The construction of multipoint iterative methods uses the new information at different points and do not reuse any old information. symbolically we can describe it as

$$x_{i+1} = \phi\left[x_i, \omega_1(x_i), \cdots, \omega_k(x_i)\right],$$

where ϕ is called multipoint iteration function. In similar fashion one can define multipoint iterative method with memory if we reuse the old information

$$x_{i+1} = \phi(z_i; z_{i-1}, \cdots, z_{i-n}),$$

where z_j represent the $k+1$ quantities $x_j, \omega_1(x_j), \cdots, \omega_k(x_j)$. Before to proceed further, we provide the definitions of different types of convergence orders. Let $\{x_n\} \subset \mathbb{R}^N$ and $x^* \in \mathbb{R}^N$. Then

- $x_n \to x^*$ q-quadratically if $x_n \to x^*$ and there is $K > 0$ such that

$$||x_{n+1} - x^*|| \leq K||x_n - x^*||^2.$$

- $x_n \to x^*$ q-superlinearly with q-order $\alpha > 1$ if $\quad x_n \to x^*$ and there is $K > 0$ such that
$$||x_{n+1} - x^*|| \leq K||x_n - x^*||^{\alpha}.$$
- $x_n \to x^*$ q-superlinearly if $\quad \lim_{n\to\infty} \frac{||x_{n+1}-x^*||}{||x_n-x^*||} = 0.$
- $x_n \to x^*$ q-linearly with q-order $\sigma \in (0,1)$ if $||x_{n+1} - x^*|| \leq \sigma||x_n - x^*||$ for n sufficiently large.

The performances of an iterative method are measured principally by its convergence-order (CO), computational efficiency and radius of convergence, but mainly the first two issues are addressed, while the third is often very difficult to deal with. Recently researchers have started to handle the convergence-radius by plotting the dynamics of iterative methods in the complex plane [4–21]. To clarify the difference between one- and multi-point iterative methods, we provide some examples. Examples of one-point iterative methods without-memory of different orders, are:

$$\begin{cases} x_{n+1} = x_n - t_1 \quad \text{(2nd order Newton-Raphson method)} \\ x_{n+1} = x_n - t_1 - t_2 \quad \text{(3rd order)} \\ x_{n+1} = x_n - t_1 - t_2 - t_3 \quad \text{(4th order)} \\ x_{n+1} = x_n - t_1 - t_2 - t_3 - t_4 \quad \text{(5th order)} \\ x_{n+1} = x_n - t_1 - t_2 - t_3 - t_4 - t_5 \quad \text{(6th order)}, \end{cases} \tag{1.3}$$

where $c_2 = f''(x_n)/2!f'(x_n)$, $c_3 = f'''(x_n)/3!f'(x_n)$, $c_4 = f''''(x_n)/4!f'(x_n)$, $c_5 = f'''''(x_n)/5!f'(x_n)$, $t_1 = f(x_n)/f'(x_n)$, $t_2 = c_2 t_1^2$, $t_3 = (-c_3 + 2c_2^2)t_1^3$, $t_4 = (-5c_2c_3 + 5c_2^3 + c_4)t_1^4$, $t_5 = (-c_5 + 3c_3^2 + 14c_2^4 - 21c_3c_2^2 + 6c_2c_4)t_1^5$ and $f(x) = 0$ is a nonlinear equation. Further examples of single-point iterative methods are the Euler method [3, 22]:

$$\begin{cases} L(x_n) = \frac{f(x_n)f''(x_n)}{f'(x_n)^2}, \\ x_{n+1} = x_n - \frac{2}{1+\sqrt{1-2L(x_n)}} \frac{f(x_n)}{f'(x_n)}. \end{cases} \tag{1.4}$$

the Halley method [23, 24]:

$$x_{n+1} = x_n - \frac{2}{2 - L(x_n)} \frac{f(x_n)}{f'(x_n)}. \tag{1.5}$$

the Chebyshev method[3]:

$$x_{n+1} = x_n - \left(1 + \frac{L(x_n)}{2}\right) \frac{f(x_n)}{f'(x_n)}. \tag{1.6}$$

One may notice that all the information (function and all its higher order derivatives) are provided at a single-point x_n. The well-know Ostrowski' method [2]:

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f'(x_n)}, \\ x_{n+1} = y_n - \frac{f(y_n)}{f'(x_n)} \frac{f(x_n)}{f(x_n) - 2f(y_n)} \end{cases} \tag{1.7}$$

has convergence-order four and it is an example of multi-point iterative method without-memory. Clearly the information is distributed among different points. The one-point iterative method without memory (1.3) uses four functional evaluations to achieve the fourth-order convergence while iterative method (1.7) requires only three functional evaluations to attain the same convergence-order. Generally speaking, one-point iterative methods can not compete with multi-points iterative method due to their low convergence-order when they use the same number of functional evaluations. The low convergence-order is not the only bad feature of one-point iterative methods, but they also suffer from narrow convergence-region compared with multi-step iterative methods. A valuable information about multi-points iterative methods for scalar nonlinear equations can be found in [25–42] and references therein. The secant-method:

$$x_{n+1} = x_n - \frac{x_{n-1} - x_n}{f(x_{n-1}) - f(x_n)} f(x_n), \tag{1.8}$$

is an example of single-point iterative method with-memory and, in most cases, it shows better convergence-order than Newton-Raphson method (1.3). Usually the multi-point iterative methods with-memory are constructed from derivative-free iterative methods for scalar nonlinear equations [43–54], but some of them are not derivative-free [55–57].

Alicia Cordero et. al. [43] developed the following iterative method with-memory:

$$\begin{cases} w_n = x_n + \gamma f(x_n), \\ N_3(t) = N_3(t; x_n, y_{n-1}, x_{n-1}, w_{n-1}), \\ N_4(t) = N_4(t; w_n, x_n, y_{n-1}, w_{n-1}, x_{n-1}), \\ \gamma_n = \frac{-1}{N_3'(x_n)}, \\ \lambda_n = -\frac{N_4''(w_n)}{2N_4'(w_n)}, \\ y_n = x_n - \frac{f(x_n)}{f[x_n, w_n] + \lambda f(w_n)}, \\ x_{n+1} = y_n - \frac{f(x_n)}{f[x_n, y_n] + (y_n - x_n) f[x_n, w_n, y_n]}, \end{cases} \tag{1.9}$$

where N_3 and N_4 are Newton interpolating polynomials. The convergence-order of (1.9) is four if we consider it without-memory, but with-memory it shows seventh-order convergence in the vicinity of a root. The iterative methods without-memory satisfy a specific criterion for convergence-order. According to Kung-Traub conjecture [58], if an iterative scheme without-memory uses n-functional evaluations then it achieves maximum convergence-order 2^{n-1}, and we call it optimal convergence-order. For scalar nonlinear equations, many researchers have proposed optimal-order (in the sense of Kung-Traub) iterative schemes derivative-free [59–64] and derivative-bases [65–70]. One of the derivative-free optimal sixteenth-order iterative scheme constructed by R. Thukral [63]:

$$\begin{cases} w_n = x_n + f(x_n), \\ y_n = x_n - \frac{f(x_n)}{f[w_n, x_n]}, \\ \phi_3 = f[x_n, w_n] f[y_n, w_n]^{-1}, \\ z_n = y_n - \phi_3 \frac{f(y_n)}{f[x_n, y_n]}, \\ \eta = (1 + 2u_3 u_4^2)^{-1} (1 - u^2)^{-1}, \\ a_n = z_n - \eta \left(\frac{f(z_n)}{f[y_n, z_n] - f[x_n, y_n] + f[x_n, z_n]} \right), \\ \sigma = 1 + u_1 u_2 - u_1 u_3 u_4^2 + u_5 + u_6 + u_1^2 u_4 + u_2^2 u_3 + 3u_1 u_4^2 (u_3^2 - u_4^2) f[x_n, y_n]^{-1}, \\ x_{n+1} = z_n - \sigma \left(\frac{f[y_n, z_n] f(a_n)}{f[y_n, a_n] f[z_n, a_n]} \right). \end{cases} \tag{1.10}$$

The iterative methods for scalar nonlinear equations also have application in the construction of iterative methods to find generalized inverses. For instance, the Newton method has a connection with quadratically convergent Schulz iterative method [71] for

finding the Moore-Penrose inverse that is written as

$$X_{n+1} = X_n(2I - AX_n), \tag{1.11}$$

where A is a matrix. Consider $f(X) = A - X^{-1}$ then the Newton iterate to find the zero of $f(X)$ is the following

$$X_{n+1} = X_n - \left(A - X_n^{-1}\right) X_n^2 = X_n(2I - AX_n). \tag{1.12}$$

Recently considerable researchers got attention to develop matrix iterative methods to find the generalized inverses [71–103]. A high-order (twelfth-order) stable numerical method for matrix inversion is presented in [96]. Actually the iterative method (1.12) for scalar nonlinear equations is used to develop a higher-order iterative method [96] (1.12) to compute matrix inverse.

$$\begin{cases} y_n = x_n - 1/2 f'(x_n)^{-1} f(x_n), \\ z_n = x_n - f'(y_n)^{-1} f(x_n), \\ u_n = z_n - ((z_n - x_n)^{-1}(f(z_n) - f(x_n)))^{-1} f(z_n), \\ g_n = u_n - f(u_n)^{-1} f(u_n), \\ x_{n+1} = g_n - ((g_n - u_n)^{-1}(f(g_n) - f(u_n)))^{-1} f(g_n). \end{cases} \tag{1.13}$$

By applying iterative method (1.13) on matrix nonlinear equation $AV - I = O$, the iterative method (1.14) is constructed.

$$\begin{cases} \psi_n = AV_n, \\ \xi_n = 171 + \psi_n(-28I + \psi_n(22I + \psi_n(-8I + \psi_n))), \\ \kappa_n = \psi_n \xi_n, \\ V_{n+1} = \frac{1}{64} V_n \xi_n (48I + \kappa_n(-12I + \kappa_n)). \end{cases} \tag{1.14}$$

A vast research has been conducted in the area of iterative methods for nonlinear scalar equations, but the iterative methods to solve systems of nonlinear equations are relatively less explored. Some of the iterative methods that are originally devised for scalar nonlinear equations are equally valid for systems of nonlinear equations. The well-know

Newton-Raphson iterative method for the system of nonlinear equations is:

$$\begin{cases} F'(\boldsymbol{x}_n)\boldsymbol{\phi}_1 = F(\boldsymbol{x}_n), \\ \boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \boldsymbol{\phi}_1, \end{cases} \tag{1.15}$$

where $F(x) = 0$ is the system nonlinear equations. The fourth-order Jarratt [104] iterative scheme scalar version can be written for systems of nonlinear equations as:

$$\begin{cases} F'(\boldsymbol{x}_n)\boldsymbol{\phi}_1 = F(\boldsymbol{x}_n), \\ \boldsymbol{y}_n = \boldsymbol{x}_n - \frac{2}{3}\boldsymbol{\phi}_1, \\ (3F'(\boldsymbol{y}_n) - F'(\boldsymbol{x}_n))\boldsymbol{\phi}_2 = 3F'(\boldsymbol{y}_n) + F(\boldsymbol{x}_n), \\ \boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \frac{1}{2}\boldsymbol{\phi}_2\boldsymbol{\phi}_1. \end{cases} \tag{1.16}$$

However it is not true that every iterative method for scalar nonlinear equations can be adopted to solve systems of nonlinear equations. For example in the well-know Ostrowski method (1.7) the term $f(\boldsymbol{x}_n)/(f(\boldsymbol{x}_n) - 2f(\boldsymbol{y}_n))$ does not make any sense for systems of nonlinear equations. The notion of optimal convergence-order is not defined for systems of nonlinear equations. The iterative methods that use less number of functional evaluations, Jacobian evaluations, Jacobian inverses, matrix-matrix multiplications and matrix-vector multiplications are considered to be computationally efficient. H. Montazeri et. al. [105] constructed an efficient iterative scheme with four convergence-order:

$$\begin{cases} F'(\boldsymbol{x}_n)\boldsymbol{\phi}_1 = F(\boldsymbol{x}_n), \\ \boldsymbol{y}_n = \boldsymbol{x}_n - \frac{2}{3}\boldsymbol{\phi}_1, \\ F'(\boldsymbol{x}_n)T = F'(\boldsymbol{y}_n), \\ \boldsymbol{\phi}_2 = T\boldsymbol{\phi}_1, \\ \boldsymbol{\phi}_3 = T\boldsymbol{\phi}_2, \\ \boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \frac{23}{8}\boldsymbol{\phi}_1 + 3\boldsymbol{\phi}_2 - \frac{9}{8}\boldsymbol{\phi}_3. \end{cases} \tag{1.17}$$

The iterative scheme (1.17) uses only one functional evaluation, two Jacobian evaluations, one Jacobian inversion (in the sense of LU-decomposition), two matrix-vector multiplications and four scalar-vector multiplications. The iterative scheme described above is efficient because it requires only one matrix inversion. An excellent dissertation about systems of the nonlinear equations can be consulted in [40, 106–134] and references therein. The next step in the construction of iterative methods for systems of the

nonlinear equations is the construction of multi-step methods. The multi-step methods are much interesting in the case of systems of nonlinear equations but for scalar nonlinear equation case they are not efficient. The multi-step version of Newton-Raphson iterative method is:

$$NS_1 = \begin{cases} \text{Number of steps} & = m \\ \text{Convergence-order} & = m+1 \\ \text{Function evaluations} & = m \\ \text{Jacobian evaluations} & = 1 \\ \text{LU-decomposition} & = 1 \\ \text{solutions of systems of linear equations when right hand side is a vector} & = m \end{cases} \begin{cases} \text{Base-method} \rightarrow \begin{cases} F'(\boldsymbol{x}_n)\boldsymbol{\phi}_1 = F(\boldsymbol{x}_n), \\ \boldsymbol{y}_1 = \boldsymbol{x}_n - \boldsymbol{\phi}_1 \end{cases} \\ \text{Multi-step} \rightarrow \begin{cases} \text{for } \; i = 1 : m-1 \\ F'(\boldsymbol{x}_n)\boldsymbol{\phi}_i = F(\boldsymbol{y}_i), \\ \boldsymbol{y}_{i+1} = \boldsymbol{y}_i - \boldsymbol{\phi}_i, \\ \text{end} \\ \boldsymbol{x}_{n+1} = \boldsymbol{y}_m. \end{cases} \end{cases} \tag{1.18}$$

The iterative method NS_1 is not efficient for scalar nonlinear equations because according to Kung-Traub conjecture for each evaluation of a function the enhancement in the convergence-order is a multiplier of two. In the iterative scheme (1.18) for each evaluation, there is an increase in convergence-order by an additive factor of one. But for systems of nonlinear equations it is efficient as it requires a single Jacobian inversion (LU-decomposition). H. Montazeri et. al. [105] have already developed the multi-step version (1.19) of iterative scheme (1.17) which is more efficient than multi-step iterative scheme (1.18):

$$
HM = \begin{cases}
\text{Number of steps} & = m \geq 2 \\
\text{Convergence-order} & = 2m \\
\text{Function evaluations} & = m-1 \\
\text{Jacobian evaluations} & = 2 \\
\text{LU-decomposition} & = 1 \\
\text{Solutions of systems} & \\
\text{of linear equations when} & \\
\text{right hand-side is a vector} & = m-1 \\
\text{right hand-side is a matrix} & = 1 \\
\text{Matrix-vector multiplications} & = m \\
\text{Vector-vector multiplications} & = 2m
\end{cases}
\begin{cases}
\text{Base-method} \rightarrow \begin{cases}
F'(\boldsymbol{x}_n)\boldsymbol{\phi}_1 = F(\boldsymbol{x}_n), \\
\boldsymbol{y}_1 = \boldsymbol{x}_n - \frac{2}{3}\boldsymbol{\phi}_1, \\
F'(\boldsymbol{x}_n)T = F'(\boldsymbol{y}_n), \\
\boldsymbol{\phi}_2 = T\boldsymbol{\phi}_1, \\
\boldsymbol{\phi}_3 = T\boldsymbol{\phi}_2, \\
\boldsymbol{y}_2 = \boldsymbol{x}_n - \frac{23}{8}\boldsymbol{\phi}_1 \\
+3\boldsymbol{\phi}_2 - \frac{9}{8}\boldsymbol{\phi}_3
\end{cases} \\
\text{Multi-step} \rightarrow \begin{cases}
\text{for } \; i = 1 : m-2 \\
F'(\boldsymbol{x}_n)\boldsymbol{\phi}_{i+3} \\
= F(\boldsymbol{y}_{i+1}), \\
\boldsymbol{y}_{i+2} = \boldsymbol{y}_{i+1} \\
-\frac{5}{2}\boldsymbol{\phi}_{i+3} + \frac{3}{2}T\boldsymbol{\phi}_{i+3}, \\
\text{end} \\
\boldsymbol{x}_{n+1} = \boldsymbol{y}_m .
\end{cases}
\end{cases}
\tag{1.19}
$$

The multi-step iterative methods can also be defined for ordinary differential equations (ODEs) and partial differential equations (PDEs). The idea of quasilinearization was introduced by R. E. Bellman, and R. E. Kalaba [135] for ODEs and PDEs. The quasilinearization [135–142] and its multi-step version for ODEs is written as

$$
\begin{aligned}
\text{ODE} \rightarrow & \begin{cases}
L(x(t)) + f(x(t)) - b = 0 \\
L \text{ is linear differential operator} \\
f(x(t)) \text{ is a nonlinear function of } x(t)
\end{cases} \\
\text{Quasilinearization } (CO = 2) \rightarrow & \begin{cases}
L(x_{n+1}) + f'(x_n)x_{n+1} = f'(x_n)x_n - f(x_n) + b \\
\text{or} \\
(L + f'(x_n))\,x_{n+1} = f'(x_n)x_n - f(x_n) + b
\end{cases} \\
\text{Multi-step } (CO = m+1) \rightarrow & \begin{cases}
(L + f'(x_n))\,y_1 = f'(x_n)x_n - f(x_n) + b \\
(L + f'(x_n))\,y_2 = f'(y_1)y_1 - f(y_1) + b \\
\vdots \\
(L + f'(x_n))\,y_m = f'(y_{m-1})y_{m-1} - f(y_{m-1}) + b.
\end{cases}
\end{aligned}
\tag{1.20}
$$

Overview of thesis

The introduction is enclosed in the first chapter. The second chapter deals with the construction of an optimal sixteenth-order iterative method for scalar nonlinear equations. The idea behind the development of an optimal-order iterative method is to use rational functions, which usually provide wider convergence-regions. A set of problems is solved and compared by using newly developed optimal-order scheme. In the majority of cases, our iterative scheme shows better results compared with other existing iterative schemes for solving nonlinear equations. In the third chapter, a general class of multi-step iterative methods to solve systems of nonlinear equations is developed, and their respective convergence-order proofs are also given. The validity, accuracy and computational efficiency are tested by solving systems of nonlinear equations. The numerical experiments show that our proposals also deal efficiently with systems of nonlinear equations stemming from partial differential equations. The chapter four and five shed light on the construction of multi-step iterative methods for systems of nonlinear equations associated with ODEs and PDEs. The distinctive idea was to address the economical usage of higher-order Fréchet derivatives in the multi-step iterative methods for systems of nonlinear equations extracted from ODEs and PDEs. The multi-step iterative methods are computationally efficient, because they re-use the Jacobian information to enhance the convergence-order. In chapter six, we proposed an iterative scheme for a general class of systems of nonlinear equations which is more efficient than the work presented in the third chapter. By plotting the convergence-region, we try to convince that higher-order iterative schemes have narrow convergence-region. Finally, chapter seven addresses the construction of a matrix iterative method to compute the weighted Moore-Penrose inverse of a matrix which is obtained using an iterative method for nonlinear equations. Several numerical tests show the superiority of our twelve-order convergent matrix iterative scheme.

Our main contribtion in this research is to explore computationally economical multi-step iterative methods for solving nonlinear problems. The multi-step iterative methods consist of two parts namely base method and multi-step part. The multi-step methods are efficients because the inversion information (LU factors) of frozen Jacobian in base method is used repeadly in the muti-step part. The burdern of computing LU factors is over base method and in multi-step part we only solve triangular systems of linear equations. In order to compute Moore-Penrose inverse of a matrix we use mutipoint iterative method. Actually the matrix version of algorithm is developed from

the scalar version of multipoint method and detailed analysis shows that the proposed matrix version of iterative method is correct, accurate and efficient.

Papers:

1. M. Z. Ullah, A. S. Al-Fhaid, and F. Ahmad, Four-Point Optimal Sixteenth-Order Iterative Method for Solving Nonlinear Equations, Journal of Applied Mathematics, Volume 2013, Article ID 850365, 5 pages.
   http://dx.doi.org/10.1155/2013/850365

2. M. Z. Ullah, F. Soleymani, and A. S. Al-Fhaid, Numerical solution of nonlinear systems by a general class of iterative methods with application to nonlinear PDEs, Numerical Algorithms, September 2014, Volume 67, Issue 1, pp 223-242.

3. M. Z. Ullah, S. Serra-Capizzano, and F. Ahmad, An efficient multi-step iterative method for computing the numerical solution of systems of nonlinear equations associated with ODEs, Applied Mathematics and Computation 250 (2015) 249-259.

4. M. Z. Ullah, S. Serra-Capizzano, and F. Ahmad, A Higher Order Multi-step Iterative Method for Computing the Numerical Solution of Systems of Nonlinear Equations Associated with Nonlinear PDEs and ODEs. (under review in Mediterranean Journal of Mathematics)

5. M. Z. Ullah, S. Serra-Capizzano, and F. Ahmad, Higher Order Multi-step Iterative Method for Computing the Numerical Solution of Systems of Nonlinear Equations: Application to Nonlinear PDEs and ODEs. (under review in Journal of Applied Mathematics and Computation)

6. M. Z. Ullah, F. Soleymani, and A. S. Al-Fhaid, An efficient matrix iteration for computing weighted Moore-Penrose inverse, Applied Mathematics and Computation 226 (2014) 441-454.

Chapter 2

Four-Point Optimal Sixteenth-Order Iterative Method for Solving Nonlinear Equations

We present an iterative method for solving nonlinear equations. The proposed iterative method has optimal order of convergence sixteen in the sense of Kung-Traub conjecture (Kung and Traub, 1974), since each iteration uses five functional evaluations to achieve the order of convergence. More precisely,the proposed iterative method utilizes one derivative and four function evaluations. Numerical experiments are carried out in order to demonstrate the convergence and the efficiency of our iterative method.

2.1 Introduction

According to the Kung and Traub conjecture, a multipoint iterative method without memory could achieve optimal convergence order 2^{n-1} by performing n evaluations of function or its derivatives [58]. In order to construct an optimal sixteenth-order convergent iterative method for solving nonlinear equations, we require four and eight optimal-order iterative schemes. Many authors have been developed optimal eighth-order iterative methods namely Bi-Ren-Wu [143], Bi-Wu-Ren [144], Guem-Kim [145], Liu-Wang [146], Wang-Liu [147] and F. Soleymani [91, 148, 149]. Some recent applications of nonlinear equation solvers in matrix inversion for regular or rectangular matrices have been introduced in [91, 148, 150].

For the proposed iterative method, we developed two new optimal fourth and eighth order iterative methods to construct optimal sixteenth-order iterative scheme. On the other hand,it is known that the rational weight functions give a better convergence radius. By keeping this fact in mind, we introduced rational terms in weight functions to achieve optimal sixteen order.

For the sake of completeness, we listed some existing optimal sixteenth order convergent methods. Babajee-Thukral [151] suggested 4-point sixteenth-order king family of iterative methods for solving nonlinear equations **(BT)**:

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f'(x_n)}, \\ z_n = y_n - \frac{1+\beta t_1}{1+(\beta-2)t_1}\frac{f(y_n)}{f'(x_n)}, \\ w_n = z_n - (\theta_0+\theta_1+\theta_2+\theta_3)\frac{f(y_n)}{f'(x_n)}, \\ x_{n+1} = w_n - (\theta_0+\theta_1+\theta_2+\theta_3+\theta_4+\theta_5+\theta_6+\theta_7)\frac{f(w_n)}{f'(x_n)}, \end{cases} \tag{2.1}$$

where

$$t_1 = \frac{f(y_n)}{f(x_n)}, \quad t_2 = \frac{f(z_n)}{f(x_n)}, \quad t_3 = \frac{f(z_n)}{f(y_n)}, \quad t_4 = \frac{f(w_n)}{f(x_n)}, \quad t_5 = \frac{f(w_n)}{f(z_n)}, \quad t_6 = \frac{f(w_n)}{f(y_n)},$$

$$\theta_0 = 1, \quad \theta_1 = \frac{1+\beta t_1+3/2\beta t_1^2}{1+(\beta-2)t_1+(3/2\beta-1)t_1^2} - 1, \quad \theta_2 = t_3, \quad \theta_3 = 4t_2, \quad \theta_4 = t_5+t_1t_2,$$

$$\theta_5 = 2t_1t_5+4(1-\beta)t_1^3t_3+2t_2t_3, \quad \theta_6 = 2t_6+(7\beta^2-47/2\beta+14)t_3t_1^4+(2\beta-3)t_2^2$$

$$+(5-2\beta)t_5t_1^2-t_3^3, \quad \theta_7 = 8t_4+(-12\beta+2\beta^2+12)t_5t_1^3-4t_3^3t_1+(-2\beta^2+12\beta-22)$$

$$t_3^2t_1^3+(-10\beta^3+127/2\beta^2-105\beta+46)t_2t_1^4.$$

In 2011, Geum-Kim [152] proposed a family of optimal sixteenth-order multipoint methods (**GK2**):

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f'(x_n)}, \\ z_n = -K_f(u_n)\frac{f(y_n)}{f'(x_n)}, \\ s_n = z_n - H_f(u_n,v_n,w_n)\frac{f(z_n)}{f'(x_n)}, \\ x_{n+1} = s_n - W_f(u_n,v_n,w_n,t_n)\frac{f(s_n)}{f'(x_n)}, \end{cases} \tag{2.2}$$

where

$$u_n = \frac{f(y_n)}{f(x_n)}, \quad v_n = \frac{f(z_n)}{f(y_n)}, \quad w_n = \frac{f(z_n)}{f(x_n)}, \quad t_n = \frac{f(s_n)}{f(z_n)},$$
$$K_f(u_n) = \frac{1+\beta u_n + (-9+5/2\beta)u_n^2}{1+(\beta-2)u_n+(-4+\beta/2)u_n^2},\ H_f = \frac{1+2u_n+(2+\sigma)w_n}{1-v_n+\sigma w_n},$$
$$W_f = \frac{1+2u_n}{1-v_n-2w_n-t_n} + G(u_n, v_n, w_n),$$

one of the choice for $G(u_n, v_n, w_n)$ along with $\beta = 24/11$ and $\sigma = -2$:

$$G(u_n, v_n, w_n) = -6u_n^3 v_n - 244/11 u_n^4 w_n + 6w_n^2 + u_n(2v_n^2 + 4v_n^3 + w_n - 2w_n^2).$$

In the same year, Geum-Kim [153] presented a biparametric family of optimally convergent sixteenth-order multipoint methods (**GK1**):

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f'((x_n)}, \\ z_n = -K_f(u_n)\frac{f(y_n)}{f'(x_n)}, \\ s_n = z_n - H_f(u_n, v_n, w_n)\frac{f(z_n)}{f'(x_n)}, \\ x_{n+1} = s_n - W_f(u_n, v_n, w_n, t_n)\frac{f(s_n)}{f'(x_n)}, \end{cases} \tag{2.3}$$

where

$$u_n = \frac{f(y_n)}{f(x_n)}, \quad v_n = \frac{f(z_n)}{f(y_n)}, \quad w_n = \frac{f(z_n)}{f(x_n)}, \quad t_n = \frac{f(s_n)}{f(z_n)},$$
$$K_f(u_n) = \frac{1+\beta u_n + (-9+5/2\beta)u_n^2}{1+(\beta-2)u_n+(-4+\beta/2)u_n^2},\ H_f = \frac{1+2u_n+(2+\sigma)w_n}{1-v_n+\sigma w_n},$$
$$W_f = \frac{1+2u_n+(2+\sigma)v_n w_n}{1-v_n-2w_n-t_n+2(1+\sigma)v_n w_n} + G(u_n, w_n),$$

one of the choice for $G(u_n, w_n)$ along with $\beta = 2$ and $\sigma = -2$:

$$G(u_n, w_n) = -1/2\left[u_n w_n(6+12u_n+(24-11\beta)u_n^2 + u_n^3\phi_1 + 4\sigma)\right] + \phi_2 w_n^2,$$
$$\phi_1 = (11\beta^2 - 66\beta + 136),\ \phi_2 = (2u_n(\sigma^2 - 2\sigma - 9) - 4\sigma - 6).$$

2.2 A new method and convergence analysis

The proposed sixteenth-order iterative method is described as follows **(MA)**:

$$\begin{cases} y_n = x_n - \frac{f(x_n)}{f'(x_n)}, \\ z_n = y_n - \frac{1+2t_1-t_1^2}{1-6t_1^2}\frac{f(y_n)}{f'(x_n)}, \\ w_n = z_n - \frac{1-t_1+t_3}{1-3t_1+2t_3-t_2}\frac{f(z_n)}{f'(x_n)}, \\ x_{n+1} = w_n - (q_1+q_2+q_3+q_4+q_5+q_6+q_7)\frac{f(w_n)}{f'(x_n)}, \end{cases} \tag{2.4}$$

where

$$t_1 = \frac{f(y_n)}{f(x_n)}, \quad t_2 = \frac{f(z_n)}{f(y_n)}, \quad t_3 = \frac{f(z_n)}{f(x_n)}, \quad t_4 = \frac{f(w_n)}{f(x_n)}, \quad t_5 = \frac{f(w_n)}{f(y_n)}, \quad t_6 = \frac{f(w_n)}{f(z_n)},$$
$$q_1 = \frac{1}{1-2(t_1+t_1^2+t_1^3+t_1^4+t_1^5+t_1^6+t_1^7)}, \quad q_2 = \frac{4t_3}{1-\frac{31}{4t_3}}, \quad q_3 = \frac{t_2}{1-t_2-20t_2^3},$$
$$q_4 = \frac{8t_4}{1-t_4} + \frac{2t_5}{1-t_5} + \frac{t_6}{1-t_6}, \quad q_5 = \frac{15t_1t_3}{1-\frac{131}{15t_3}}, \quad q_6 = \frac{54t_1^2t_3}{1-t_1^2t_3}$$
$$q_7 = 7t_2t_3 + 2t_1t_6 + 6t_6t_1^2 + 188t_3t_1^3 + 18t_6t_1^3 + 9t_2^2t_3 + 64t_1^4t_3.$$

Theorem 2.1. *Let $f : D \subseteq \mathbb{R} \to \mathbb{R}$ be a sufficiently differentiable function, and $\alpha \in D$ is a simple root of $f(x) = 0$, for an open interval D. If x_0 is chosen sufficiently close to α, then the iterative scheme (2.3) converges to α and shows an order of convergence at least equal to sixteen.*

Proof. Let error at step n be denoted by $e_n = x_n - \alpha$ and $c_1 = f'(\alpha)$ and $c_k = \frac{1}{k!}\frac{f^{(k)}(\alpha)}{f'(\alpha)}$, $k = 2, 3, \cdots$. We provided maple based computer assisted proof in Figure 2.1 and got the following error equation:

$$e_{n+1} = -c_4c_3c_2^2(c_5c_3c_2^2 + 2c_4c_2c_3^2 - 20c_3^4 - 51c_3^3c_2^2 + 522c_3^2c_2^4 - 2199c_3c_2^6 + 2c_2^8 - 30c_4c_3c_2^3 + 54c_4c_2^5)e_n^{16} + O(e_n^{17}). \tag{2.5}$$

□

2.3 Numerical results

If the convergence order η is defined as

$$lim_{n\to\infty}\frac{|e_{n+1}|}{|e_n|^{\eta}} = c \neq 0, \tag{2.6}$$

then the following expression approximates the computational order of convergence (COC) [10] as follows

$$\rho \approx \frac{ln|(x_{n+1}-\alpha)/(x_n-\alpha)|}{ln|(x_n-\alpha)/(x_{n-1}-\alpha)|}, \tag{2.7}$$

where α is the root of nonlinear equation. Further the efficiency index is $\rho^{\frac{1}{\#F.E.}}$, where #F.E. is total number of function and derivative evaluations in single cycle of iterative method. A set of five nonlinear equations is used for numerical computations in Table 2.1. Three iterations are performed to calculate the absolute error ($|x_n-\alpha|$) and computational order of convergence. Table 2.2 shows absolute error and computational order of convergence respectively.

Functions	Roots
$f_1(x) = e^x sin(x) + log(1+x^2)$	$\alpha = 0$
$f_2(x) = (x-2)(x^(10)+x+1)e^{-x-1}$	$\alpha = 2$
$f_3(x) = sin(x)^2 - x^2 + 1$	$\alpha = 1.40449\cdots$
$f_4(x) = e^{-x} - cos(x)$	$\alpha = 0$
$f_5(x) = x^3 + log(x)$	$\alpha = 0.70470949\cdots$

TABLE 2.1: Set of six nonlinear functions

2.4 Summary

An optimal sixteenth order iterative scheme has been developed for solving nonlinear equations. A Maple program is provided to calculate error equation, which actually shows the optimal order of convergence in the sense of the Kung-Traub conjecture. The computational order of convergence also verifies our claimed order of convergence. The proposed scheme uses four function and one derivative evaluations per full cycle which gives 1.741 as the efficiency index. We also have shown the validity of our

$(f_n(x),x_0)$	Iter/COC	MA	BT	GK1	GK2
f_1, 1.0	1	0.00268	0.00183	0.0111	0.00230
	2	2.03e-37	1.71e-37	6.35e-24	5.61e-34
	3	**2.47e-583**	3.53e-582	1.37e-363	1.03e-523
	COC	16	16	16	16
f_2, 2.5	1	0.04086	0.0639	0.0296	0.00866
	2	6.16e-9	650.0	5.35e-14	2.53e-21
	3	1.50e-121	divergent	4.79e-201	**1.89e-317**
	COC	16.5	-	15.9	16.0
f_3, 2.5	1	0.0000326	0.0000303	0.000497	0.0000677
	2	4.87e-73	1.70e-72	1.56e-51	1.14e-64
	3	**3.11e-1158**	1.63e-1148	1.42e-811	4.52e-1021
	COC	16	16	16	16
f_4, 1/6	1	2.79e-7	0.0000864	1.28e-7	0.000167
	2	1.00e-109	1.18e-63	2.28e-107	9.28e-57
	3	**2.80e-1851**	1.72e-1005	2.24e-1703	7.82e-893
	COC	17	16	16	16
f_5, 3.0	1	0.0486	0.135	0.0949	0.0133
	2	1.95e-22	1.81e-17	1.78e-19	1.11e-35
	3	8.46e-349	1.79e-271	6.86e-302	**2.61e-563**
	COC	16.0	16.0	15.9	16.0

TABLE 2.2: Numerical comparison of absolute error $|x_n - \alpha|$, number of iterations =3

proposed iterative scheme by comparing it with other existing optimal sixteen-order iterative methods. The numerical results demonstrate that the considered iterative scheme is competitive when compared with other methods taken from the relevant literature.

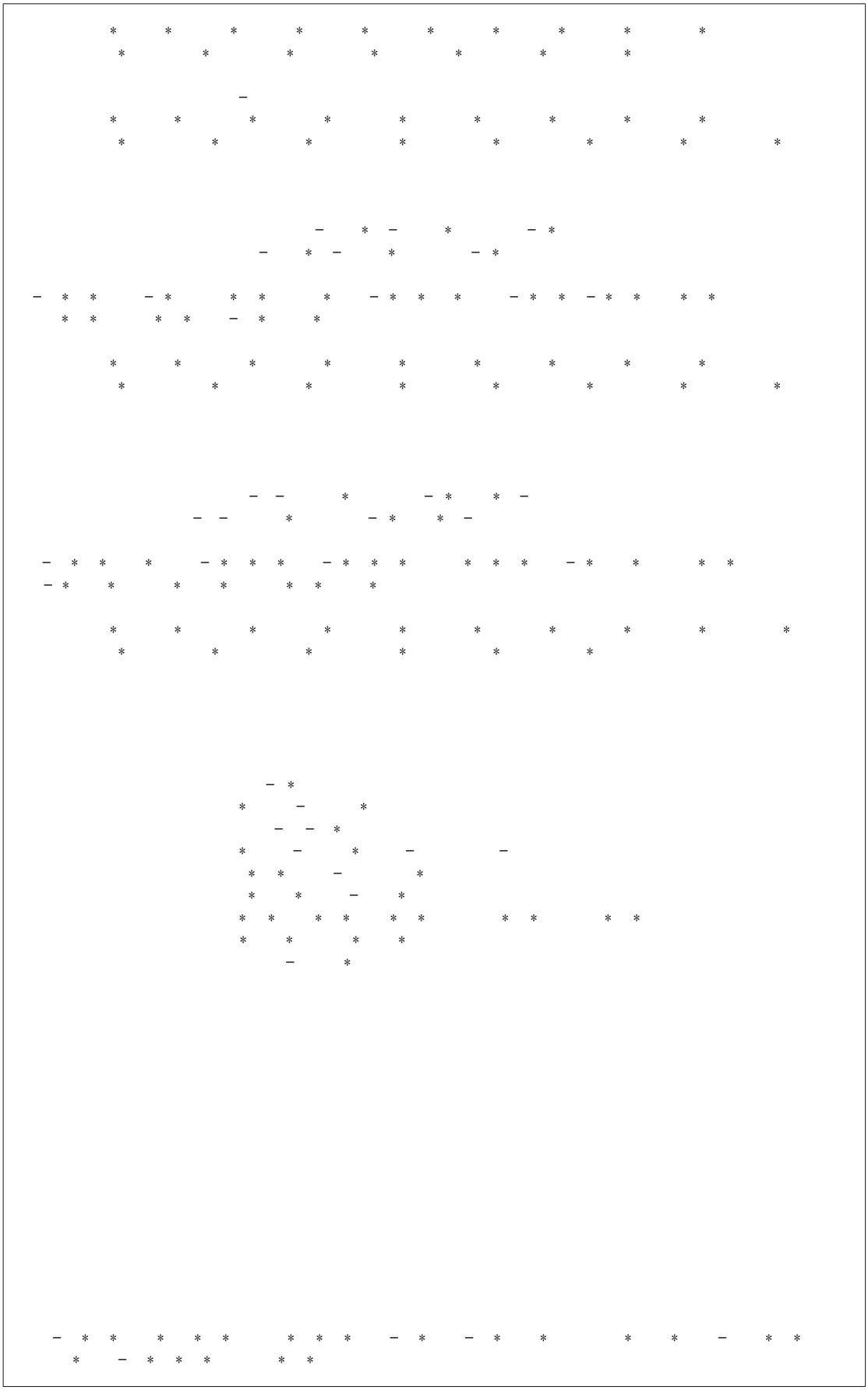

FIGURE 2.1: Algorithm: The maple code for finding the error equation.

Chapter 3

Numerical Solution of Nonlinear Systems by a General Class of Iterative Methods with Application to Nonlinear PDEs

A general class of multi-step iterative methods for finding approximate real or complex solutions of nonlinear systems is presented. The well-known technique of undetermined coefficients is used to construct the first method of the class while, the higher order schemes will be attained by making use of a frozen Jacobian. The 'point of attraction' theory will be taken into account to prove the convergence behavior of the main proposed iterative method. Then, it will be observed that a m-step method converges with $2m$-order. A discussion of the computational efficiency index alongside numerical comparisons with the existing methods will be given. Finally, we illustrate the application of the new schemes in solving nonlinear partial differential equations.

3.1 Introduction

In this chapter, we introduce a general class of multi-step iterative methods free from second or higher-order Frechet derivatives for solving the nonlinear system of equations $F(x) = 0$, where $F : D \subset \mathbb{R}^N \to \mathbb{R}^N$ is a continuously differentiable mapping. We are

interested in high-order fast methods for which the computational load and efficiency are reasonable to deal with nonlinear systems.

Let $F(x)$ be differentiable enough and x^* be the solution such that $F(x^*) = 0$ and $\det(F'(x^*)) \neq 0$. Then, according to [3] $F(x)$ has an inverse $G : \mathbb{R}^N \to \mathbb{R}^N$, where $x = G(y)$ defined in a neighborhood of $F(x^*) = 0$. We further consider that $x^{(0)} \in D$ is a starting vector or guess of x^* and $y^{(0)} = F(x^{(0)})$.

By approximating $G(x)$ with its first-order Taylor series around $y^{(0)}$, we have $G(y) \simeq G(y_0) + G'(y_0)(y - y_0)$. As in the scalar case, we can obtain a new approximation $x^{(1)}$ of $G(0) = x^*$ by making

$$x^{(1)} = G(y_0) - G'(y_0)y_0 = x^{(0)} - F'(x^{(0)})^{-1}F(x^{(0)}). \tag{3.1}$$

This iteration method leads to the well-known Newton's method in several variables

$$x^{(n+1)} = x^{(n)} - F'(x^{(n)})^{-1}F(x^{(n)}), \quad n = 0, 1, 2, \cdots. \tag{3.2}$$

Another famous and efficient scheme for solving systems of nonlinear equations is the Jarratt fourth-order method [154] defined as follows

$$\begin{cases} y^{(n)} = x^{(n)} - \frac{2}{3}F'(x^{(n)})^{-1}F(x^{(n)}), \\ x^{(n+1)} = x^{(n)} - \frac{1}{2}(3F'(y^{(n)}) - F'(x^{(n)}))^{-1} \\ \qquad\qquad \cdot(3F'(y^{(n)}) + F'(x^{(n)}))F'(x^{(n)})^{-1}F(x^{(n)}). \end{cases} \tag{3.3}$$

As is known, the scheme (3.2) reaches second order using the evaluations $F(x^{(n)})$ and $F'(x^{(n)})$, while the scheme (3.3) achieves fourth-order by applying $F(x^{(n)})$, $F'(x^{(n)})$ and $F'(y^{(n)})$ per computing step.

Such solvers have many applications. For example, Pourjafari et al. in [155] discussed the application of such methods in optimization and engineering problems. Besides that, Waziri et al. in [156] showed their application in solving integral equations by proposing an approximation for the Jacobian matrix per computing step in the form of a diagonal matrix. In this work, we will show the application of such solvers for nonlinear problems arising in the solution of Partial Differential Equations (PDEs). Interested readers may refer to [40], [157], [158], and [159] for further pointers.

In this chapter, in order to improve the convergence behavior of the above well-known methods and to find an efficient scheme to tackle the nonlinear problems, we

propose a new one with eighth-order convergence to find both real and complex solutions. The new proposed method only needs first-order Frechet derivative evaluations and is in fact free from higher-order Frechet derivatives. As a matter of fact, third-order methods such as Halley's and Chebyshev's methods need second-order Frechet derivative evaluations, which makes the computation process time-consuming [160].

The remaining sections of the chapter are organized in what follows. Section 3.2 provides the construction of a new iteration method. Section 3.6.1 discusses the convergence order of the method using the theory of point of attraction due to [161]. Then in Section 3.4, we propose a general multi-step nonlinear solver for systems of equations. Next, Section 3.5 the computational efficiency will be discussed. Numerical results are included in Section 3.6 to validate the effectiveness of the proposed method in finding real and complex solutions of nonlinear systems with applications. Ultimately, Section 3.7 draws a conclusion of this study.

3.2 The construction of the method

We here construct a new scheme. Let us first consider the well-known technique of undetermined coefficients to develop a high order method as follows in the scalar case $(n = 0, 1, 2, \cdots)$

$$\begin{cases} y_n = x_n - \frac{2}{3}\frac{f(x_n)}{f'(x_n)}, \\ z_n = x_n - \frac{1}{2}\frac{3f'(y_n)+f'(x_n)}{3f'(y_n)-f'(x_n)}\frac{f(x_n)}{f'(x_n)}, \\ w_n = z_n - \frac{f(z_n)}{q_1 f'(x_n)+q_2 f'(y_n)}, \\ x_{n+1} = w_n - \frac{f(w_n)}{q_1 f'(x_n)+q_2 f'(y_n)}. \end{cases} \tag{3.4}$$

As could be seen, the structure (3.4) includes four steps in which the denominator of the third and fourth steps are considered to be the same on purpose. To illustrate further, this assumption leads us to improve the order of convergence from the third step to the fourth one, while the computational burden in order to solve the involved linear systems and the Jacobian is low. Once we generalize (3.4) to N dimensions, the Jacobians $F'(x^n)$ and $F'(y^n)$ will be computed once per cycle and the last two sub-steps will not impose high burdensome load to the technique. Note also that the constructed method in this way would be different form to the combination of Jarratt method with the Chord method discussed in [162].

Now we employ the Computer Algebra System Mathematica [163] to find the unknown quantities. It should also be remarked that the first two steps of (3.4) is the Jarratt fourth-order method (3.3). Let us assume $e = x - x^*$, $u = y - x^*$, $v = z - x^*$, $t = w - x^*$, $ee = x_{new} - x^*$, (without the index n) and $dfa = f'(x^*)$. Using its Taylor series, the function can be defined as follows

```
f[e_] := dfa*(e + c2*e^2 + c3*e^3
            + c4*e^4 + c5*e^5 + c6*e^6 + c7*e^7 + c8*e^8).
```

Now, according to the first two steps of (3.4), we can write the Taylor expansion in what follows

```
u = e - (2/3)*Series[f[e]/f'[e], {e, 0, 6}] //Simplify;
v = e - Series[(1/2)*((3*f'[u] + f'[e])/(3*f'[u]
                - f'[e]))*(f[e]/f'[e]), {e, 0, 6}]//Simplify.
```

At this time when we obtain fourth-order convergence (the attained error equation reveals this fact), we keep going to the third step:

```
t = v - f[v]/(q1*f'[e] + q2*f'[u]) // Simplify.
```

Let us now have the coefficients of the fourth and fifth terms in the obtained error equation

```
a4 = Coefficient[t, e^4] // FullSimplify
a5 = Coefficient[t, e^5] // FullSimplify.
```

To attain sixth-order convergence, we now solve a simultaneous linear system of two equations using a command in Mathematica as follows:

```
Solve[{a4 == 0, a5 == 0}, {q1, q2}] // FullSimplify.
```

This gives the following results

$$q_1 \to -\frac{1}{2}, \quad q_2 \to \frac{3}{2}. \tag{3.5}$$

Due to the fact that the correcting factors in the third and fourth steps of our structure (3.4) are equal, thus we have further acceleration in the convergence rate at the end of the structure (3.4). Implementing again the Taylor expansion, we obtain

```
ee = t - f[t]/((-1/2)*f'[e] + (3/2)*f'[u]) // Simplify;
```

Thus, the final asymptotic error constant of our scheme (3.4) can be produced by applying

```
a8 = Coefficient[ee, e^8] // FullSimplify
```

We now propose our high-order method for finding real and complex solutions of the nonlinear systems in what follows

$$\begin{cases} y^{(n)} = x^{(n)} - \frac{2}{3}F'(x^{(n)})^{-1}F(x^{(n)}), \\ z^{(n)} = x^{(n)} - \frac{1}{2}(3F'(y^{(n)}) - F'(x^{(n)}))^{-1} \\ \qquad\qquad \cdot(3F'(y^{(n)}) + F'(x^{(n)}))F'(x^{(n)})^{-1}F(x^{(n)}), \\ w^{(n)} = z^{(n)} - (\frac{-1}{2}F'(x^{(n)}) + \frac{3}{2}F'(y^{(n)}))^{-1}F(z^{(n)}), \\ x^{(n+1)} = w^{(n)} - (\frac{-1}{2}F'(x^{(n)}) + \frac{3}{2}F'(y^{(n)}))^{-1}F(w^{(n)}). \end{cases} \tag{3.6}$$

Per computing step, the new method (3.6) requires to compute F at three different points and the Jacobians F' at two points. In order to prove the convergence order of (3.6), we need to firstly remind some important lemmas in the theory of point of attraction.

Lemma 3.1. *(Perturbation Theory [160]) Suppose that $A, C \in L(\mathbb{R}^N)$, where $L(\mathbb{R}^N)$ is the space of linear operators from $\mathbb{R}^N$ to $\mathbb{R}^N$. Let A be nonsingular and $||A^{-1}|| \leq \alpha$, $||A - C|| \leq \mu$, $\alpha\mu < 1$, then C is nonsingular and*

$$||C^{-1}|| \leq \frac{\alpha}{1 - \alpha\mu}. \tag{3.7}$$

Lemma 3.2. *[160] Assume that $G : D \subset \mathbb{R}^N \to \mathbb{R}^N$ has a fixed point $x^* \in int(D)$ and $G(x)$ is Frechet-differentiable on x^* if*

$$\rho(G'(x^*)) = \sigma < 1, \tag{3.8}$$

then there exists $S = S(x^*, \delta) \subset D$, for any $x^{(0)} \in S$. Thus, x^* is an attraction point of $x^{(n+1)} = G(x^{(n)})$.

Lemma 3.3. *[160] Suppose $F : D \subset \mathbb{R}^N \to \mathbb{R}^N$ has a p^{th} Frechet-derivative and its $F^{(p)}$ be hemi-continuous at each point of a convex set $D_0 \subset D$, then for any $u, v \in D_0$, if $F^{(p)}$ is bounded, that is,*

$$\|F^{(p)}(u)\| \leq k_p, \tag{3.9}$$

then, one has

$$\left\| F(v) - F(u) - \sum_{j=1}^{p-1} \frac{1}{j!} F^{(j)}(u)(v-u)^j \right\| \leq \frac{k_p}{p!} \|v-u\|^p. \tag{3.10}$$

Lemma 3.4. *[164] Suppose that $C : D \subset \mathbb{R}^N \to \mathbb{R}^N$ and $\mathbf{M} : D \subset \mathbb{R}^N \to \mathbb{R}^N$ are functional depending on F with $C(x^*) = \mathbf{0}$ and $\mathbf{M}(x^*) = x^*$ and $\mathbf{M}$ and C are Frechet differentiable at a point $x^* \in int(D)$. Let $A : S_0 \to L(\mathbb{R}^N)$ be defined on an open neighborhood $S_0 \subset D$ of x^* and continuous at x^*. Assume further that $A(x^*)$ is nonsingular. Then, there exists a ball*

$$S = \overline{S}(x^*, \delta) = \left\{ \|x^* - x\| \leq \delta \right\} \subset S_0, \ \delta > 0,$$

on which the mapping

$$G : S \to \mathbb{R}^N, \quad G(x) = \mathbf{M}(x) - A(x)^{-1} C(x), \, forall \ , x \in S,$$

is well-defined; moreover, G is Frechet differentiable at x^*, thus

$$G'(x^*) = \mathbf{M}'(x^*) - A(x^*)^{-1} C'(x^*).$$

Theorem 3.5. *(Attraction Theorem for Newton Method) [160] Let $F : D \subset \mathbb{R}^N \to \mathbb{R}^N$ be twice Frechet-differentiable in an open ball $S = S(x^*, \delta) \subset D$, where $x^* \in int(D)$, and $F''(x)$ be bounded on S. Assume that $F(x^*) = 0$ and that $F'(x^*)$ is non-singular, that is, $\|F'(x^*)^{-1}\| = \alpha$. Then x^* is a point of attraction of the iteration defined by Newton iteration function*

$$N(x) = x - F'(x)^{-1} F(x). \tag{3.11}$$

That is, $N(x)$ is well defined and satisfies an estimate of the form

$$\|N(x) - x^*\| \leq \lambda_1 \|\mathbf{e}\|^2, \tag{3.12}$$

for all $x \in S_1$, on some ball $S_1 = \overline{S}(x^*, \delta_1) \subset S$ where $\mathbf{e} = x - x^*$.

3.3 Convergence analysis

In this section, we study the convergence behavior of the method (3.6) using point of attraction theory.

Theorem 3.6 (Attraction Theorem for the Jarratt Method [161])**.** *Under the conditions of Theorem 3.5, suppose that $F''(x)$ is bounded and $F^{(3)}(x)$ is both bounded and Lipschitz continuous. Then x^* is a point of attraction of the iteration defined by process $z^{(n)} = G_{4^{th}JM}(x^{(n)})$. The iteration function is well defined and satisfies an estimate of the form*

$$\|G_{4^{th}JM}(x) - x^*\| \leq \lambda_2 \|\mathbf{e}\|^4, \tag{3.13}$$

for all $x \in S_2$, on some ball $S_2 = \overline{S}(x^*, \delta_2) \subset S_1$.

Note that $G_{4^{th}JM}$ and $G_{8^{th}JM}$ are the fourth and eighth order Jarratt-type methods (3.3) and (3.6), respectively. Similar notations have been used throughout. We begin the rest of the work by proving two important lemmas.

Lemma 3.7. *Let*

$$A(x) = -\frac{1}{2}F'(x) + \frac{3}{2}F'(y(x)), \tag{3.14}$$

wherein $y(x) = x + \frac{2}{3}(N(x) - x)$. Under the conditions of Theorem 3.6, we have

$$||A(x) - F'(x^*)|| \leq (\frac{10}{27}k_3 + \lambda_1 k_2)||e||^2 + \frac{k_3\lambda_1}{3}||e||^3, \tag{3.15}$$

for all $x \in S_3$, on some ball $S_3 = \overline{S}(x^*, \delta_3) \subset S_2$. Furthermore $A(x)^{-1}$ exists and

$$||A(x)^{-1}|| \leq \frac{\alpha}{1 - [(\frac{10}{27}\alpha k_3 + \alpha\lambda_1 k_2)||e||^2 + \frac{\alpha k_3\lambda_1}{3}||e||^3]}, \tag{3.16}$$

whenever $\alpha\left(\frac{10}{27}k_3 + \lambda_1 k_2\right)\delta_3^2 + \frac{\alpha k_3\lambda_1}{3}\delta_3^3 < 1$.

Proof. Since $F(x^*) = 0$, we have $y(x^*) = x^*$ and thus $A(x^*) = F'(x^*)$. Now

$$y(x) - x^* = \frac{x - x^*}{3} + \frac{2}{3}(N(x) - x^*). \tag{3.17}$$

Let us find the upper bound of $||A(x) - F'(x^*)||$. By the mean value theorem for integrals, we have

$$\begin{aligned} A(x) - A(x^*) &= -\frac{1}{2}(F'(x) - F(x^*)) + \frac{3}{2}(F'(y(x)) - F'(x^*)) \\ &= -\frac{1}{2}\int_0^1 F''(x^* + t(x - x^*))dt(x - x^*) \\ &\quad + \frac{3}{2}\int_0^1 F''(x^* + s(y(x) - x^*))ds(y(x) - x^*)), \end{aligned}$$

which, using eq. (3.17), simplifies to

$$A(x)-A(x^*)=W_1(x-x^*)+W_2(N(x)-x^*), \tag{3.18}$$

wherein $W_1=\int_0^1\int_0^1 F''(x^*+s(y(x)-x^*))-F''(x^*+t(x-x^*))dsdt$ and $W_2=\int_0^1 F''(x^*+s(y(x)-x^*))ds$. Furthermore, using the bounds on F''', the Schwartz inequality and eq. (3.17), we have

$$\begin{aligned} ||W_1|| &= ||\int_0^1\int_0^1\int_0^1 F'''\Big(x^*+t(x-x^*)+w(s(y(x)-x^*) \\ &\quad -t(x-x^*))\Big)\times\ s(y(x)-x^*)-t(x-x^*)dsdtdw|| \qquad (3.19)\\ &\leq k_3\int_0^1\int_0^1 ||s(y(x)-x^*)-t(x-x^*)dsdt|| \qquad (3.20)\\ &= k_3\int_0^1\int_0^1 ||(s/3-t)(x-x^*)+\frac{2s}{3}(N(x)-x^*)dsdt||. \end{aligned}$$

Now, by applying the triangular inequality, one may have

$$||W_1||\leq k_3||x-x^*||\int_0^1\int_0^1|\frac{s}{3}-t|dsdt+k_3||N(x)-x^*||\int_0^1\int_0^1\frac{2s}{3}dsdt \tag{3.21}$$

which reduces to

$$||W_1||\leq k_3\frac{10}{27}||x-x^*||+\frac{k_3}{3}||N(x)-x^*||. \tag{3.22}$$

We also have $||W_2||\leq k_2$ using the bounds of F''. We further could write using eqs. (3.22) and (3.12):

$$\begin{aligned} \|A(x)-F'(x^*)\| &\leq ||W_1||\ ||x-x^*||+||W_2||| \ |N(x)-x^*|| \qquad (3.23)\\ &\leq \frac{10}{27}k_3||x-x^*||^2+\frac{k_3}{3}||x-x^*||\ ||N(x)-x^*|| \qquad (3.24)\\ &\quad +k_2||N(x)-x^*||, \end{aligned}$$

and

$$\|A(x)-F'(x^*)\| \leq \frac{10}{27}k_3||e||^2+\frac{k_3}{3}\lambda_1||e||^3+\lambda_1k_2||e||^2, \tag{3.25}$$

which proves eq. (3.15). Since $||e||\leq\delta_3$ for all $x\in S_3$ and $||A(x^*)^{-1}||=||F'(x^*)^{-1}||=\alpha$, we have

$$||A(x^*)^{-1}||\ \|A(x)-F'(x^*)\|\leq\alpha\big[\frac{10}{27}k_3+\lambda_1k_2\big]\delta_3^2+\alpha\frac{k_3}{3}\lambda_1\delta_3^3<1, \tag{3.26}$$

which satisfies our assumption. By the Perturbation Lemma, $A(x)^{-1}$ exists and its bound is given by

$$\begin{aligned} ||A(x)^{-1}|| &\leq \frac{||A(x^*)^{-1}||}{1-||A(x^*)^{-1}||\,||A(x)-A(x^*)||} && (3.27)\\ &\leq \frac{\alpha}{1-[(\frac{10}{27}\alpha k_3+\alpha\lambda_1 k_2)||e||^2+\frac{\alpha k_3\lambda_1}{3}||e||^3]}. && (3.28)\end{aligned}$$

The proof is now complete. □

Lemma 3.8. *Under the conditions of Theorem 3.6,*

$$\begin{aligned} ||F(G(x))-A(x)(G(x)-x^*)|| &\leq \frac{k_2}{2}||G(x)-x^*||^2+[(\frac{10}{27}k_3 \\ &\quad +\lambda_1 k_2)||e||^2+\frac{k_3\lambda_1}{3}||e||^3]||G(x)-x^*||, \quad (3.29)\end{aligned}$$

for all $x \in S_3 \subset S_2$ and $G(x)$ is any iteration function.

Proof. Using eq. (3.10) with $v = G(x)$, $u = x^*$ and $p = 2$ in Lemma 3.1 and eq. (3.15), we have

$$\begin{aligned} ||F(G(x))-A(x)(G(x)-x^*)|| &= ||F(G(x))-F'(x^*)(G(x)-x^*) \\ &\quad +(F'(x^*)-A(x))(G(x)-x^*)|| \\ &\leq ||F(G(x))-F'(x^*)(G(x)-x^*)|| \quad (3.30)\\ &\quad +||A(x)-F'(x^*)||\,||G(x)-x^*|| \\ &\leq \frac{k_2}{2}||G(x)-x^*||^2+[(\frac{10}{27}k_3+\lambda_1 k_2)||e||^2 \\ &\quad +\frac{k_3\lambda_1}{3}||e||^3]||G(x)-x^*||.\end{aligned}$$

The proof is ended. □

Theorem 3.9. *Under the conditions of Theorem 3.6, x^* is a point of attraction of the iteration defined by process $\mathbf{w}^{(n)} = G_{6^{th}JM}(x^{(n)})$. Thus, the iteration function is well defined and satisfies an estimate of the form*

$$\|G_{6^{th}JM}(x)-x^*\| \leq \lambda_3\|\mathbf{e}\|^6, \quad (3.31)$$

for all $x \in S_3$, on some ball $S_3 = \overline{S}(x^*, \delta_3) \subset S_2$.

Proof. We have $\mathbf{M}(x) = G_{4^{th}JM}(x)$ and $C(x) = F(G_{4^{th}JM}(x))$ which are differentiable functions at x^*. Also, $A(x)$ is continuous at x^* for all $x \in S_3$. All the conditions of Lemma

3.4 are satisfied. Therefore $G'_{6^{th}JM}(x^*) = \mathbf{0}$ since $G'_{4^{th}JM}(x^*) = \mathbf{0}$ and $\rho(G'_{6^{th}JM}(x^*)) = 0 < 1$. By Ostrowski's Theorem, it follows that x^* is a point of attraction of $G_{6^{th}JM}(x)$ which is well defined in S_3. Now, for any $x \in S_3$,

$$||G_{6^{th}JM} - x^*|| \leq ||A(x)^{-1}||\ ||A(x)(G_{4^{th}JM}(x) - x^*) - F(G_{4^{th}JM})||. \tag{3.32}$$

Using eqs. (3.16), (3.13) and (3.29), eq. (3.32) simplifies to

$$||G_{6^{th}JM} - x^*|| \leq \frac{\alpha}{1 - [(\frac{10}{27}\alpha k_3 + \alpha\lambda_1 k_2)||e||^2 + \frac{\alpha k_3 \lambda_1}{3}||e||^3]}$$

$$\left[\frac{k_2\lambda_2^2}{2}||e||^8 + (\frac{10}{27}\lambda_2 k_3 + \lambda_1\lambda_2 k_2)||e||^6 + \frac{k_3\lambda_1\lambda_2}{3}||e||^7\right] = \lambda_3||e||^6, \tag{3.33}$$

where

$$\lambda_3 = \frac{\alpha}{1 - [(\frac{10}{27}\alpha k_3 + \alpha\lambda_1 k_2)||e||^2 + \frac{\alpha k_3\lambda_1}{3}||e||^3]}$$

$$\left[\frac{k_2\lambda_2^2}{2}||e||^2 + (\frac{10}{27}\lambda_2 k_3 + \lambda_1\lambda_2 k_2) + \frac{k_3\lambda_1\lambda_2}{3}||e||\right],$$

which establish the sixth order convergence. We here note that k_2 and k_3 are upper bounds on F'' and F''' based on eq. (3.9) for their definitions and defining λ_2 as constant depending on bounds of F'' and also the bounds and Lipschitz continuity of F'''. This completes the proof. □

Theorem 3.10. *Under the conditions of Theorem 3.6, x^* is a point of attraction of the iteration defined by process $x^{(n+1)} = G_{8^{th}JM}(x^{(n)})$. The iteration function is well defined and satisfies an estimate of the form*

$$\|G_{8^{th}JM}(x) - x^*\| \leq \lambda_4\|\mathbf{e}\|^8, \tag{3.34}$$

for all $x \in S_4$, on some ball $S_4 = \overline{S}(x^*, \delta_4) \subset S_3$.

Proof. We have $\mathbf{M}(x) = G_{6^{th}JM}(x)$ and $C(x) = F(G_{6^{th}JM}(x))$, which are differentiable functions at x^*. Also, $A(x)$ is continuous at x^* for all $x \in S_4$. All the conditions of Lemma 3.4 are satisfied. Therefore $G'_{8^{th}JM}(x^*) = \mathbf{0}$ since $G'_{6^{th}JM}(x^*) = \mathbf{0}$ and $\rho(G'_{8^{th}JM}(x^*)) = 0 < 1$. By Ostrowski's Theorem, it follows that x^* is a point of attraction of $G_{8^{th}JM}(x)$ which is well defined in S_4. Now, for any $x \in S_4$,

$$||G_{8^{th}JM} - x^*|| \leq ||A(x)^{-1}||\ ||A(x)(G_{6^{th}JM}(x) - x^*) - F(G_{6^{th}JM})||. \tag{3.35}$$

Using eqs. (3.16), (3.31) and (3.29), eq. (3.35) simplifies to

$$||G_{8^{th}JM} - x^*|| \leq \frac{\alpha}{1 - [(\frac{10}{27}\alpha k_3 + \alpha\lambda_1 k_2)||e||^2 + \frac{\alpha k_3 \lambda_1}{3}||e||^3]}$$

$$\left[\frac{k_2\lambda_2^2}{2}||e||^{12} + (\frac{10}{27}\lambda_2 k_3 + \lambda_1\lambda_2 k_2)||e||^8 + \frac{k_3\lambda_1\lambda_2}{3}||e||^9\right] = \lambda_4||e||^8, \tag{3.36}$$

where

$$\lambda_4 = \frac{\alpha}{1 - [(\frac{10}{27}\alpha k_3 + \alpha\lambda_1 k_2)||e||^2 + \frac{\alpha k_3 \lambda_1}{3}||e||^3]}$$

$$\left[\frac{k_2\lambda_2^2}{2}||e||^4 + (\frac{10}{27}\lambda_2 k_3 + \lambda_1\lambda_2 k_2) + \frac{k_3\lambda_1\lambda_2}{3}||e||\right],$$

and it establishes the eighth order convergence. □

When dealing with hard nonlinear systems of equations, the computational efficiency of the considered iterative method is important. In what follows, we first try to simplify the proposed scheme to be computationally efficient. Simplifying the third and fourth step of (3.6) results in a same correcting factor for these sub-steps and the same as the second step, i.e.

$$\begin{cases} y^{(n)} = x^{(n)} - \frac{2}{3}F'(x^{(n)})^{-1}F(x^{(n)}), \\ z^{(n)} = x^{(n)} - \frac{1}{2}(3F'(y^{(n)}) - F'(x^{(n)}))^{-1} \\ \qquad\qquad \cdot(3F'(y^{(n)}) + F'(x^{(n)}))F'(x^{(n)})^{-1}F(x^{(n)}), \\ w^{(n)} = z^{(n)} - 2(3F'(y^{(n)}) - F'(x^{(n)}))^{-1}F(z^{(n)}), \\ x^{(n+1)} = w^{(n)} - 2(3F'(y^{(n)}) - F'(x^{(n)}))^{-1}F(w^{(n)}). \end{cases} \tag{3.37}$$

Now the implementation of (3.37) quite depends on the involved linear algebra problems. Hopefully the linear system $F'(x^{(n)})V_n = F(x^{(n)})$ could be computed once per step in order to avoid computing the inverse $F'(x^{(n)})^{-1}$, and its vector solution will

be used twice per step as follows

$$\begin{cases} y^{(n)} = x^{(n)} - \frac{2}{3}V_n, \\ z^{(n)} = x^{(n)} - \frac{1}{2}M_n^{-1}(3F'(y^{(n)}) + F'(x^{(n)}))V_n, \\ w^{(n)} = z^{(n)} - 2M_n^{-1}F(z^{(n)}), \\ x^{(n+1)} = w^{(n)} - 2M_n^{-1}F(w^{(n)}), \end{cases} \tag{3.38}$$

wherein $M_n = 3F'(y^{(n)}) - F'(x^{(n)})$. Another interesting point in the formulation (3.38) is that the LU decomposition of M_n needs to be done only once, but it could effectively be used three times per computing step to increase the rate of convergence without imposing much computational burden and time. A thorough discussion of this would be given in Section 3.5.

3.4 Further extensions

This section presents a general class of multi-step nonlinear solvers. In fact, the new scheme (3.38) can simply be improved by considering the Jacobian M_n to be frozen. In such a way, we are able to propose a general m-step multi-point class of iterative methods in the following structure:

$$\begin{cases} \vartheta_1^{(n)} = x^{(n)} - \frac{2}{3}V_n, \\ \vartheta_2^{(n)} = x^{(n)} - \frac{1}{2}\rho_n^{-1}(3F'(\vartheta_1^{(n)}) + F'(x^{(n)}))V_n, \\ \vartheta_3^{(n)} = \vartheta_2^{(n)} - 2\rho_n^{-1}F(\vartheta_2^{(n)}), \\ \vartheta_4^{(n)} = \vartheta_3^{(n)} - 2\rho_n^{-1}F(\vartheta_3^{(n)}), \\ \vdots \\ x^{(n+1)} = \vartheta_m^n = \vartheta_{m-1}^{(n)} - 2\rho_n^{-1}F(\vartheta_{m-1}^{(n)}), \end{cases} \tag{3.39}$$

wherein $\rho_n = 3F'(\vartheta_1^{(n)}) - F'(x^{(n)})$. It is fascinating to mention that in this structure, the LU factorization of the Jacobian matrix ρ_n would be computed only once and it would be consumed $m-2$ times through one full cycle (we have $m-2$ linear systems with the same coefficient matrix ρ_n and multiple right hand sides). This fully reduce the computational load of the linear algebra problems involved in implementing (3.39).

In the iterative process (3.39) each added step will impose one more N-dimensional function whose cost is N scalar evaluations the convergence order will improved to

$2+O(m-1)$, wherein $O(m-1)$ is the order of the previous sub-steps. Considering the well-known Mathematical induction, it would be easy to deduce the following theorem for (3.39):

Theorem 3.11. *The m-step ($m \geq 3$) iterative process (3.39) has the local convergence order $2m$ using $m-1$ evaluations of the function F and two first-order Frechet derivative F' per full iteration.*

Proof. The proof of this theorem is based on Mathematical induction and is straightforward. Hence, it is omitted. □

As an example, the six-step twelfth-order method from the new class has the following structure:

$$\begin{cases} \vartheta_1^{(n)} = x^{(n)} - \frac{2}{3}V_n, \\ \vartheta_2^{(n)} = x^{(n)} - \frac{1}{2}\rho_n^{-1}(3F'(\vartheta_1^{(n)}) + F'(x^{(n)}))V_n, \\ \vartheta_3^{(n)} = \vartheta_2^{(n)} - 2\rho_n^{-1}F(\vartheta_2^{(n)}), \\ \vartheta_4^{(n)} = \vartheta_3^{(n)} - 2\rho_n^{-1}F(\vartheta_3^{(n)}), \\ \vartheta_5^{(n)} = \vartheta_4^{(n)} - 2\rho_n^{-1}F(\vartheta_4^{(n)}), \\ x^{(n+1)} = \vartheta_6^n = \vartheta_5^{(n)} - 2\rho_n^{-1}F(\vartheta_5^{(n)}). \end{cases} \tag{3.40}$$

3.5 Comparison on computational efficiency index

When considering the iterative method (3.38), one has to solve a linear system of equations $M_n K_n = b_i$, $i = 1,2,3$, that is to say, a linear system with three multiple right hand sides. In such a case, one could compute a factorization of the matrix and use it repeatedly.

The way that MATHEMATICA 8 allows users to re-use the factorization is really simple. It is common to save the factorization and use it to solve repeated problems. Herein, this is done by using a one-argument form of `LinearSolve`; this returns a functional that one can apply to different vectors/matrices to obtain the solutions.

We here apply MATHEMATICA 8 due to the fact that it performs faster for large scale problems. See Figure 3.1 (left and right), in which the comparison among this programming package and Maple have been illustrated based on [165] with entries between 10^{-5} and 10^5 for the test matrices.

The iterative method (3.38) has the following cost: N evaluations of scalar functions for $F(x)$, N evaluations of scalar functions for $F(z)$, N evaluations of scalar functions for $F(w)$, N^2 evaluations of scalar functions for the Jacobian $F'(x)$, again N^2 evaluations of scalar functions for the Jacobian $F'(y)$ and two LU decompositions for solving the linear systems involved.

To be more precise, the computing of the LU decomposition by any of the existing algorithms in the literature normally requires $\frac{2N^3}{3}$ flops in the floating point arithmetic, while the floating point operations for solving the two triangular systems will be $2N^2$ when the right hand side of the systems is a vector, and $2N^3$, or roughly N^3 as considered herein, when the right hand side is a matrix.

In computing, flops (for FLoating-point Operations Per Second) is a measure of computer performance, especially in fields of scientific calculations that make heavy use of floating-point calculations. Alternatively, the singular FLOP (or flop) is used as an abbreviation for "FLoating-point OPeration", and a flop count is a count of these operations (e.g., required by a given algorithm or computer program). We remind that in Matlab, the flops for solving the two triangular systems is $1.5N^2$ using High Performance Computing (HPC) challenge instead of $2N^2$. High Performance Computing (HPC) aims at providing reasonably fast computing solutions to both scientific and real life technical problems [166]. The advent of multicore architectures is noteworthy in the HPC history, because it has brought the underlying concept of multiprocessing into common consideration and has changed the landscape of standard computing.

There are numerous indices for assessing the computational efficiency of the nonlinear system solvers. We now provide the comparison of efficiency indices for the method (3.2), the scheme (3.3), the sixth-order method of Cordero et al. (CM) in [154], which is in fact the first three steps of the algorithm (3.37), the new method (3.38) and also one method from the general multi-step iteration (3.39), we choose e.g. the twelfth-order iterative method (3.40). In what follows, we consider two different approaches for comparing the efficiencies. One is the traditional efficiency index which is given by $E = p^{\frac{1}{c}}$, where p is the order of convergence and c is the number of functional evaluations and more practically the flops-like efficiency index by $E = p^{\frac{1}{C}}$, where C stands for the total computational cost per iteration in terms of the number of functional evaluations along with cost of LU decompositions and solving two triangular systems (based on the flops). So, we considered a same cost for the operations and function evaluations. It is obvious that the second approach is much more practical in evaluating the performance of nonlinear equations solvers.

However, we tried to compare different methods in a same and similar environment in this work, not in HPC. In fact, herein we consider that the cost of scalar function evaluation is nearly the same to the cost of doing typical operations in factiorization of a matrix and they are unity. Note that it is only an assumption and in general the relation between the cost of scalar function evaluation and the cost of doing operations are related to the specifications of the computer.

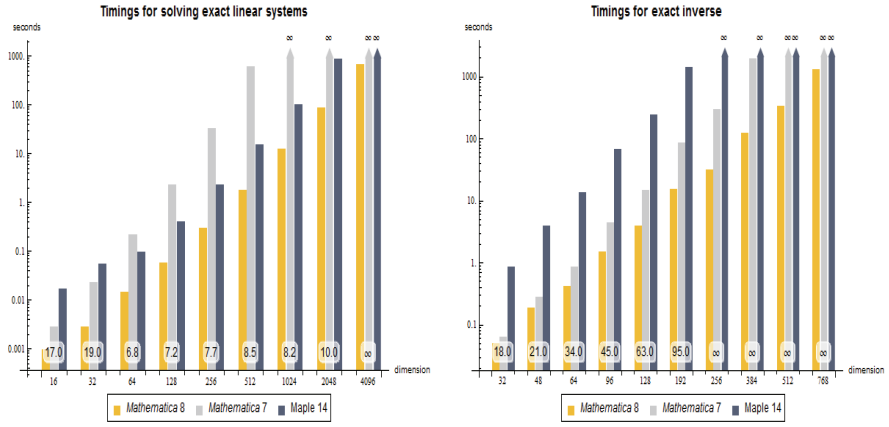

FIGURE 3.1: Timings for solving linear systems of different sizes (left) and Timings for computing the inverse of integer matrices with different sizes (right).

It is clearly obvious that the new methods (3.38) and (3.40) beat the other schemes (3.2), (3.3) and (CM), for any $N \geq 2$, for both traditional efficiency index and the flops-like efficiency index. Note that for the traditional efficiency index, we have $TEI_{(3.2)} = 2^{\frac{1}{N+N^2}}$, $TEI_{(3.3)} = 4^{\frac{1}{N+2N^2}}$, $TEI_{(CM)} = 6^{\frac{1}{2N+2N^2}}$ and for the proposed methods $TEI_{(3.38)} = 8^{\frac{1}{3N+2N^2}}$ and $TEI_{(3.40)} = 12^{\frac{1}{5N+2N^2}}$. And for the flops-like efficiency indices, we have $FEI_{(3.2)} = 2^{\frac{1}{N+3N^2+\frac{2N^3}{3}}}$, $FEI_{(3.3)} = 4^{\frac{1}{N+4N^2+\frac{7N^3}{3}}}$, $FEI_{(CM)} = 6^{\frac{1}{2N+6N^2+\frac{7N^3}{3}}}$ and for the proposed methods $FEI_{(3.38)} = 8^{\frac{1}{3N+8N^2+\frac{7N^3}{3}}}$ and $FEI_{(3.40)} = 12^{\frac{1}{5N+12N^2+\frac{7N^3}{3}}}$.

The comparison of the traditional efficiency index and the flops-like efficiency index are given in Figures 3.2 and 3.3, respectively. In these figures, the colors blue, red, purple, brown and black stand for (3.40), (3.6), (3.3), (CM) and (3.2) respectively. In Figure 3.3, and based on the practical flops-like efficiency index, there is no clear winner for low dimensional systems, while for higher dimensions, the scheme (3.40) is the best one. Clearly, higher order methods from the general class of iterative methods (3.40), will have much better practical efficiencies.

3.6 Numerical results and applications

We divide this section into two sub-sections, where the first part only contains pure academical tests, while the second one focuses on application-oriented problems.

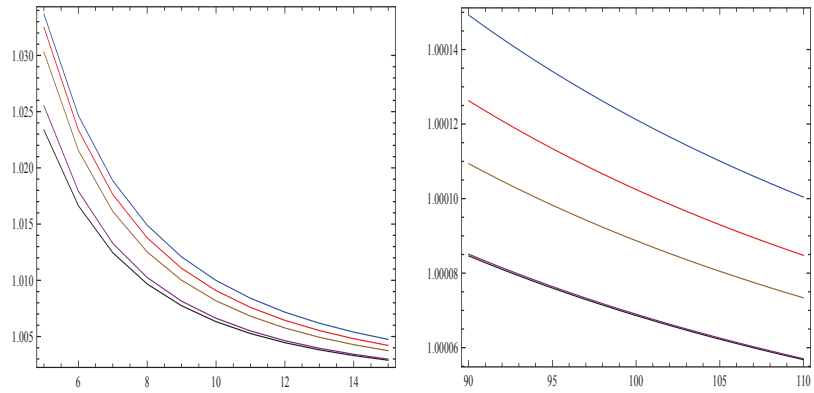

FIGURE 3.2: The comparison of the traditional efficiency indices for different methods (left $N = 5, ..., 15$) and (right $N = 90, ..., 110$). The colors blue, red, purple, brown and black stand for (3.40), (3.38), (3.3), (CM) and (3.2).

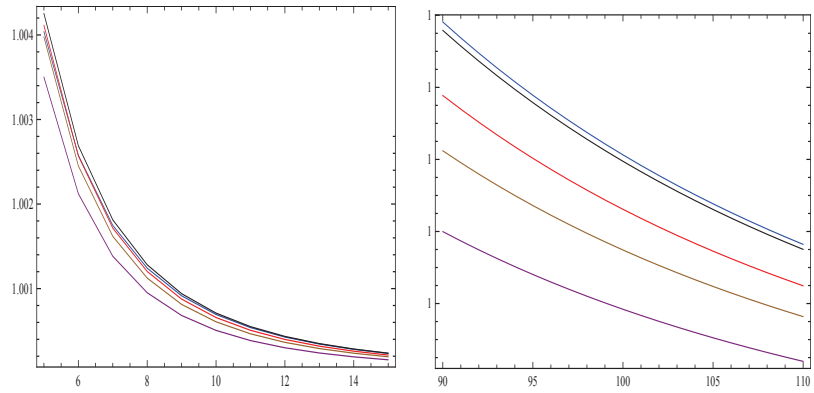

FIGURE 3.3: The comparison of the flops-like efficiency indices for different methods (left $N = 5, ..., 15$) and (right $N = 90, ..., 110$). The colors blue, red, purple, brown and black stand for (3.40), (3.38), (3.3), (CM) and (3.2).

We employ here the second order method of Newton (3.2) denoted by NM, the fourth-order scheme of Jarratt (3.3) denoted by JM, the sixth-order method of Cordero et al. CM and the proposed eighth-order methods (3.38) denoted by PM8 in sub-section 3.6.1, and the twelfth-order method (3.40) denoted by PM12, in the subsection 3.6.2 to compare the numerical results obtained from these methods in solving test nonlinear systems.

3.6.1 Academical tests

Experiment 1. As a first problem, we take into account the following system of three nonlinear equations

$$\begin{cases} x_1^3 - x_2^4 + x_3 = 0, \\ x_2^2 - x_3 x_1 = 0, \\ x_3^2 - x_1 x_2^4 = 0. \end{cases} \tag{3.41}$$

Due to page limitations, we show the solution of each system herein up to 20 decimal places. In this test problem the solution is the following vector: $x^* \approx (1.0000000000000000000,\ 1.2720196495140689643,\ 1.6180339887498948482)^T$.

Experiment 2. In order to reveal the capability of the suggested method in finding complex solutions, we consider the following nonlinear system:

$$\begin{cases} \sin(x_1) - \cos(x_3) - x_3 + x_4^8 + x_5^2 - x_6^2 = 0, \\ 3x_1 + x_2 x_3 - \tan(x_4) + x_6^2 = 0, \\ x_2^3 - x_1 - 81 x_3^{x_5} - 10 = 0, \\ x_1 x_2 + 2x_3 - \sin(x_4) = 0, \\ x_3^5 + \exp(x_6) = 0, \\ \sin(x_1) - x_2^{x_3} + 10x_4 - x_6^3 = 0, \end{cases} \tag{3.42}$$

where its complex root is as follows

$$x^* \approx \begin{pmatrix} 0.39749982364780502837 + 0.26108554547088111068i \\ -1.9503599303349781065 + 3.5327977271774986368i \\ 0.91919518428164788413 - 0.30271085317212191688i \\ 0.13551997536770926739 + 0.28832297358589852369i \\ 0.34902615622431383784 - 1.16057339570295584425i \\ -0.1638711842606608810 + 1.5509142363607599627i \end{pmatrix}. \tag{3.43}$$

Experiment 3. We consider the following test problem

$$\begin{cases} 1-\arctan(x_1x_2)+\sin(x_3)+\exp(\sin(x_5))=0,\\ x_2^{x_4}-x_4^2-2\sin(x_5)=0,\\ 1-\sin(x_1)+\sin(x_4)=0,\\ x_1^4+x_2^3+x_3^2-x_4+x_5=0,\\ x_1^{10}-x_5^3-10=0, \end{cases} \tag{3.44}$$

where its complex solution is as follows

$$x^* \approx \begin{pmatrix} 1.7823769260571925309-0.0009598583894715211i\\ 1.3869672965337494318+2.2356662518971732430i\\ -1.2859657551711001939+0.6583636453356390906i\\ -0.022301196074677766935+0.000201625720320392369i\\ 6.7939198734625295879-0.012584617398393257 2i \end{pmatrix}. \tag{3.45}$$

In order to have a fair comparison, we let the methods of lower orders perform a greater number of full cycles. We report the numerical results for solving the Experiments 1-3 in the Tables 3.1,3.2 and 3.3 based on the initial guesses. The residual norm along with the number of iterations and computational time using the command `AbsoluteTiming[]` in MATHEMATICA 8 are reported in Tables 3.1,3.2 and 3.3. An efficient way to numerically observe the behavior of the order of convergence is to use the local computational order of convergence (COC) that can be defined by

$$COC \approx \frac{\ln(||F(x^{(n+1)})||/||F(x^{(n)})||)}{\ln(||F(x^{(n)})||/||F(x^{(n-1)})||)}, \tag{3.46}$$

for the N-dimensional case.

Iterative methods	(NM)	(JM)	(CM)	(PM8)
Number of iterations	17	8	7	6
The residual norm	1.32×10^{-103}	4.33×10^{-97}	2.56×10^{-187}	$5.98*10^{-118}$
COC	2.00	4.05	6.06	8.19
The elapsed time	0.093	0.070	0.062	0.046

TABLE 3.1: Results of comparisons for different methods in Experiment 1 using $x^{(0)} = (14,10,10)$

Iterative methods	(NM)	(JM)	(CM)	(PM8)
Number of iterations	*Div.*	5	4	4
The residual norm	–	1.97×10^{-33}	1.27×10^{-33}	2.37×10^{-76}
COC	–	3.98	6.00	7.97
The elapsed time	–	0.51	0.46	0.50

TABLE 3.2: Results of comparisons for different methods in Experiment 2 using $x^{(0)} = (I, 2I, 1, I, I, 3I)$ and $I = \sqrt{-1}$

In numerical comparisons, we have chosen the fixed point arithmetics to be 200 using the command `SetAccuracy` [`expr`, `200`] in the written codes. We employ a stopping criterion based on the residual norm of the multi-variate function with tolerance parameter $1.E-97$, with norm 2. Note that the computer specifications are Microsoft Windows XP Intel(R), Pentium(R) 4 CPU, 3.20GHz with 4GB of RAM.

Results for Experiment 1 reveals that the proposed method requires less or equal number of iterations to obtain higher accuracy in contrast to the other methods. Tables 3.2 and 3.3 also reveals the importance of the initial guess. For example, although the new scheme (3.6) with two corrector steps (the third and fourth steps) per full computing step is much more better than the schemes NM, JM and CM, a simple line search could be done in solving nonlinear systems of equations to let the initial guess arrive at the convergence basin. Robust strategies for providing enough accurate initial guesses have been discussed in [167] and [168].

Tables 3.1, 3.2 and 3.3 also include the local computational order of convergence for different methods (by the use of the last three full steps of each scheme) to numerically observe the analytical discussion given in Section on the rate of convergence for the contributed methods. Note that the elapsed time in this chapter are expressed in seconds.

We also remind that in case of facing with singular matrices if a badly chosen initial approximation be chosen, then one might use the generalized outer inverses (see e.g. [169, 170] and [171]) instead of the regular inverse without losing the convergence.

3.6.2 Application-oriented tests

This sub-section reveals the application of the new method in solving a nonlinear problem arising from the discretization of nonlinear PDEs. Generally speaking, when solving a nonlinear problem such as nonlinear PDEs, it would be reduced to

Iterative methods	(NM)	(JM)	(CM)	(PM8)
Number of iterations	10	*Div.*	*Div.*	7
The residual norm	5.14×10^{-65}	–	–	7.67×10^{-85}
COC	2.00	–	–	8.02
The elapsed time	0.26	–	–	0.31

TABLE 3.3: Results of comparisons for different methods in Experiment 3 using $x^{(0)} = (2.1, I, 1.9, -I1, 2)$

solving a system of nonlinear algebraic equations. Another point is that for such cases, basically numerical results of lower accuracy in terms of the precision are needed. Hence, we here try to find the solution using finite difference discretization with the stopping criterion $||F(x)||_2 < 10^{-8}$. In the following tests, we consider $u = u(x,t)$, which is the exact solution of the nonlinear PDE. The approximate solution is denoted by $w_{i,j} \simeq u(x_i,t_j)$ at the node i, j on the considered mesh [172]. Here, we assume M and N be the number of steps along the space and time, and $m = M-1$, $n = N-1$.

Experiment 4. A simplified model of fluid flow is the Burgers' equation with Dirichlet boundary conditions and the diffusion coefficient D:

$$\begin{cases} u_t + uu_x = Du_{xx}, \\ u(x,0) = \frac{2D\beta\pi\sin(\pi x)}{\alpha+\beta\cos(\pi x)}, \ \ 0 \le x \le 2, \\ u(0,t) = 0, \ t \ge 0, \\ u(1,t) = 0, \ t \ge 0. \end{cases} \tag{3.47}$$

To solve this PDE, we use the backward finite difference for the first derivative along the time (the independent variable t): $u_t(x_i,t_j) \simeq \frac{w_{i,j}-w_{i,j-1}}{k}$, where k is the step size, and the central finite difference for the other involved pieces of the equations, i.e., $u_x(x_i,t_j) \simeq \frac{w_{i+1,j}-w_{i-1,j}}{2h}$, and $u_{xx}(x_i,t_j) \simeq \frac{w_{i+1,j}-2w_{i,j}+w_{i-1,j}}{h^2}$, wherein h is the step size along the space (x). We consider $\alpha = 5$, $\beta = 4$, $D = 0.05$, and $T = 1$.

In solving (3.47), the procedure will end in a nonlinear system of algebraic equations having a large sparse Jacobian matrix, which have been solved and compared through the tested methods applied in Section 3.6.1. The solution has been plotted in Figure 3.4. Table 3.4 presents the comparison results for this test. For this test we have chosen $M = N = 21$, to obtain a nonlinear system of size 400, with the starting vector $\mathtt{x_0} = \mathtt{Table}[0.6, \{\mathtt{i}, 1, \mathtt{m} * \mathtt{n}\}]$ in the Mathematica environment with the machine precision.

Iterative methods	(NM)	(JM)	(CM)	(PM12)
Number of iterations	4	2	2	1
The elapsed time	5.03	3.78	4.39	3.34

TABLE 3.4: Results of comparisons for different methods in Experiment 4

Experiment 5. An interesting category of nonlinear PDEs is comprised of reaction-diffusion equation. A fundamental example of such a equation is due to the evolutionary biologist and geneticist R.A. Fisher. The equation was originally derived to model how genes propagate. In what follows, consider solving the Fisher's equation with homogenous Neumann boundary conditions:

$$\begin{cases} u_t = Du_{xx} + u(1-u), \\ u(x,0) = \sin(\pi x), \ \ 0 \le x \le 1, \\ u_x(0,t) = 0, \ t \ge 0, \\ u_x(1,t) = 0, \ t \ge 0. \end{cases} \tag{3.48}$$

Note that $f(u) = u(1-u)$, implying that $f'(u) = 1-2u$. For solving (3.48) using the same discretizations as in Experiment 4, we will obtain a system of nonlinear equations. The greatest difficulty in this case is that, unlike the previous case, for Neumann boundary conditions, two sets of new nonlinear equations at the grid points will be included to the system. That is, the following discretization equations must be added: $u_x(0,t_j) \simeq \frac{-3w_{0,j}+4w_{1,j}-w_{2,j}}{2h}$, and $u_x(1,t_j) \simeq \frac{-3w_{m-2,j}+4w_{m-1,j}-w_{m,j}}{2h}$. The numerical comparison of solving this test have been given in Table 3.5, while the solution has been plotted in Figure 3.5. In this test, we have chosen $M = N = 23$, to obtain a nonlinear system of the size 528, which provides a large sparse Jacobian with $\mathtt{x_0} = \mathtt{Table}[0., \{\mathtt{i}, 1, \mathtt{m} * \mathtt{n}\}]$ as the starting vector.

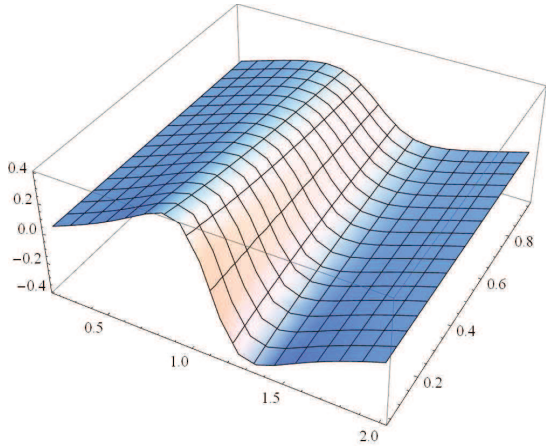

FIGURE 3.4: The approximate solution of Burgers' equation using Finite Difference scheme and our novel iterative method PM12.

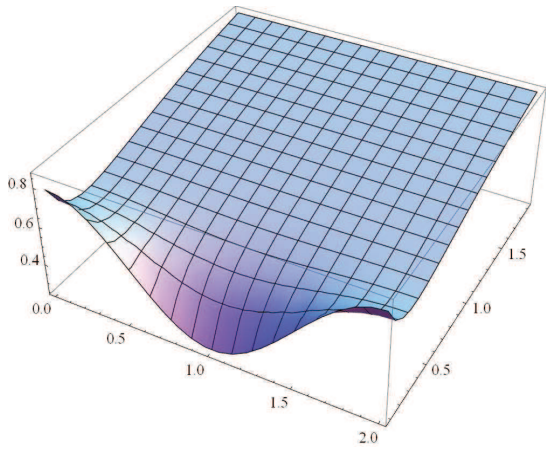

FIGURE 3.5: The approximate solution of Fisher's equation using Finite Difference scheme and our novel iterative method PM12.

3.7 Summary

The contribution of chapter consists in a high-order class of multi-step iterative methods for finding the solution of nonlinear systems of equations. The construction of the suggested schemes allow us to achieve high convergence orders using appropriate computations of the Jacobian and solving few linear systems per full step of the class (3.39).

We have also supported the proposed iteration by a valid mathematical proof through the point of attraction theory. This let us to analytically find the eighth order of convergence for the first method from the proposed class while using mathematical induction, it will be easy to see the convergence order $2m$ for an m-step method. Per computing step, the method is free from second-order Frechet derivative, which is

Iterative methods	(NM)	(JM)	(CM)	(PM12)
Number of iterations	4	2	2	1
The elapsed time	7.76	5.64	6.73	5.28

TABLE 3.5: Results of comparisons for different methods in Experiment 5

also important and does not restrict the applicability of the scheme in solving hard test problems.

The computational efficiency of the method has been tested by applying two different types of index of efficiencies and Figures 3.2 and 3.3 attested the good efficiency of the methods. Besides, some different numerical tests have been used to compare the consistency and stability of the proposed iteration in contrast to the existing methods. The numerical results obtained in Section 3.6 confirm the theoretical derivations of the present chapter.

We have also revealed that the method can efficiently be used for complex zeros. We here also note that per full step of our class (3.39), and so as to avoid computing inverse of matrices, we solve linear systems using LU factorization because the left hand-side matrix is fixed and only the right hand-side vector gets numerical updates as iteration advances. This was done using the powerful command of `LinearSolve`, which allow us automatically to work with large scale and sparse nonlinear systems.

The solution of large-scale linear and nonlinear systems of equations constitutes the most time consuming task in solving many numerical simulation problems arising in scientific computing. Hence, in order to motivate our methods and test their applicability, we solved two large-scale nonlinear systems with sparse Jacobian matrices, originating from the discretization of nonlinear PDEs. Experimental results indicated that our algorithms perform very well also in such cases.

It should also be remarked that like all other iterative methods in this category, the new ones should be combined with a method such as line search to find a convergence basin in order to have appropriate initial points for achieving the convergence rate and few number of iterations. In summary, we can conclude that the novel iterative methods leads to acceptable performances in solving systems of nonlinear equations with applications.

Chapter 4

An Efficient Multi-step Iterative Method for Computing the Numerical Solution of Systems of Nonlinear Equations Associated with ODEs

We developed multi-step iterative method for computing the numerical solution of nonlinear systems, associated with ordinary differential equations (ODEs) of the form $L(x(t)) + f(x(t)) = g(t)$: here $L(\cdot)$ is a linear differential operator and $f(\cdot)$ is a nonlinear smooth function. The proposed iterative scheme only requires one inversion of Jacobian which is computationally very efficient if either LU-decomposition or GMRES-type methods are employed. The higher-order Frechet derivatives of the nonlinear system stemming from the considered ODEs are diagonal matrices. We used the higher-order Frechet derivatives to enhance the convergence-order of the iterative schemes proposed in this chapter and indeed the use of a multi-step method significantly increases the convergence-order. The second- order Frechet derivative is used in the first step of an iterative technique which produced third-order convergence. In a second step we constructed a matrix polynomial to enhance the convergence-order by three. Finally, we freeze the product of a matrix polynomial by the Jacobian inverse to generate the multi-step method. Each additional step will increase the convergence-order by three, with minimal computational effort. The convergence-order (CO) obeys the formula CO $= 3m$, where m is the number of steps per full-cycle of the considered iterative scheme.

4.1 Introduction

In this study, we consider scalar ordinary differential equations of the form:

$$L(x(t)) + f(x(t)) = g(t) \quad \text{where } t \in D, \tag{4.1}$$

where $L(\cdot)$ is a linear differential operator, $f(\cdot)$ is a differentiable nonlinear function and D is an interval subset of $\mathbb{R}$. The linear operator and the nonlinear function are well defined over the domain of problem. Furthermore, we suppose that A is the discrete approximation of the linear differential operator L over a partition $\{t_1, t_2, t_3, \cdots, t_n\}$ of domain D and

$$\boldsymbol{x} = [x(t_1), x(t_2), \cdots, x(t_n)]^T. \tag{4.2}$$

Then we can write (4.1) as follows

$$F(\boldsymbol{x}) = A\boldsymbol{x} + f(\boldsymbol{x}) - \boldsymbol{g} = \boldsymbol{0}, \tag{4.3}$$

$$F'(\boldsymbol{x}) = A + diag(f'(\boldsymbol{x})) \quad \text{under the condition} \quad det(F'(\boldsymbol{x})) \neq 0, \tag{4.4}$$

where $diag(f'(\boldsymbol{x_k}))$ is a diagonal matrix whose main diagonal is constructed from vector $f'(\boldsymbol{x_k}) = [f'((x_1)_k), f'((x_2)_k), \cdots, f'((x_n)_k)]^T$, $\boldsymbol{g} = [g(t_1), g(t_2), \cdots, g(t_n)]^T$ and $\boldsymbol{0} = [0, 0, \cdots, 0]^T$. The quasi-linearization iterative method was constructed to solve (4.1) [3, 136, 138, 139]. The original idea is to make the linear approximation of nonlinear function f with respect to solution $x(t)$. Let $x_{k+1}(t) = x_k(t) + \varepsilon(t)$ be the solution of (4.1):

$$L(x_{k+1}(t)) + f(x_{k+1}(t)) = g(t), \tag{4.5}$$

$$L(x_k(t) + \varepsilon(t)) + f(x_k(t) + \varepsilon(t)) = g(t), \tag{4.6}$$

$$L(x_k(t)) + L(\varepsilon(t)) + f(x_k(t)) + \frac{df(x)}{dx}\varepsilon(t) \approx g(t), \tag{4.7}$$

$$\left(L + \frac{df(x)}{dx}\right)\varepsilon(t) \approx -(L(x_k(t)) + f(x_k(t)) - g(t)), \tag{4.8}$$

where t is the independent variable and k is the index for iteration. After elimination of $\varepsilon(t)$ from (4.8), we obtain the following iterative method:

$$L(x_{k+1}) + f'(x_k)x_{k+1} = f'(x_k)x_k - f(x_k) + g(t), \tag{4.9}$$

which is a quasi-linear iterative method to solve (4.1) with convergence-order two. Then (4.9) can be written as:

$$A\boldsymbol{x_{k+1}} + diag(f'(\boldsymbol{x_k}))\boldsymbol{x_{k+1}} = diag(f'(\boldsymbol{x_k}))\boldsymbol{x_k} - f(\boldsymbol{x_k}) + \boldsymbol{g}, \tag{4.10}$$

$$\boldsymbol{x_{k+1}} = \boldsymbol{x_k} - (A + diag(f'(\boldsymbol{x_k})))^{-1}(A\boldsymbol{x_k} - f(\boldsymbol{x_k}) + \boldsymbol{g}), \tag{4.11}$$

The Eqn. (4.11) can be written as:

$$\boldsymbol{x_{k+1}} = \boldsymbol{x_k} - F'(\boldsymbol{x_k})^{-1}F(\boldsymbol{x_k}). \tag{4.12}$$

This shows that quasi-linearization is equivalent to Newton-Raphson iterative scheme for systems of nonlinear equations which has quadratic convergence. Multi-step quasi-linearization methods are constructed in [141, 142], but both methods face low convergence-order. Many authors [25–28, 34, 64, 107, 173] have investigated the iterative methods for nonlinear equations with higher convergence-order. For systems of nonlinear equations recent advancements are addressed in [37, 40, 109, 154, 174–180]. Recently a multi-step class of iterative methods for nonlinear systems is constructed by Fazlollah et. al. [37]:

$$\begin{cases} \boldsymbol{y}^{(k)} = \boldsymbol{x}^{(k)} - \frac{2}{3}F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{x}^{(k)}), \\ \boldsymbol{t}^{(k)} = \frac{1}{2}(3F'(\boldsymbol{y}^{(k)}) - F'(\boldsymbol{x}^{(k)}))^{-1}(3F'(\boldsymbol{y}^{(k)}) + F'(\boldsymbol{x}^{(k)})), \\ \boldsymbol{z}^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{t}^{(k)}F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{x}^{(k)}), \\ \boldsymbol{x}^{(k+1)} = \boldsymbol{z}^{(k)} - (\boldsymbol{t}^{(k)})^2F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{z}^{(k)}). \end{cases} \tag{4.13}$$

The convergence-order of (4.13) is six and it requires two evaluations of Jacobian and their inversions at different points. The multi-step version of (4.13) states that the inclusion of a further step $\boldsymbol{\phi}^{(k)} - (\boldsymbol{t}^{(k)})^2F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{\phi}^{(k)})$ increases the convergence-rate by two. The efficiency of (4.13) is hidden in the frozen factor $(\boldsymbol{t}^{(k)})^2F'(\boldsymbol{x}^{(k)})^{-1}$ which converts the scheme into an efficient multi-step iterative method. Further improvements can be achieved by reducing two matrix inversions into one matrix inversion of Jacobian. The previous idea for the enhancement of efficiency has been described in [105].

The resulting iterative method in [105] is given below:

$$\begin{cases} \boldsymbol{y}^{(k)} = \boldsymbol{x}^{(k)} - \frac{2}{3}F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{x}^{(k)}), \\ \boldsymbol{t}^{(k)} = F'(\boldsymbol{x}^{(k)})^{-1}F'(\boldsymbol{y}^{(k)}), \\ \boldsymbol{z}^{(k)} = \boldsymbol{x}^{(k)} - \left(\frac{23}{8}I - 3\boldsymbol{t}^{(k)} + \frac{9}{8}{\boldsymbol{t}^{(k)}}^2\right)F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{x}^{(k)}), \\ \boldsymbol{x}^{(k+1)} = \boldsymbol{z}^{(k)} - \left(\frac{5}{2}I - \frac{3}{2}\boldsymbol{t}^{(k)}\right)F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{z}^{(k)}). \end{cases} \tag{4.14}$$

Additional sources of valuable discussions regarding the construction of iterative methods for systems of nonlinear equations can be found in [109, 174–177]. Most of the investigated iterative methods are constructed for the general class of systems of nonlinear equations and they only consider first order Frechet derivatives, because higher order Frechet derivatives are computationally expensive. There is a classical method for nonlinear systems, called Chebyshev-Halley's method [178], which uses the second-order Frechet derivative and is given by:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[I + \frac{1}{2}[I - \alpha\boldsymbol{P}]^{-1}\boldsymbol{P}\right](F'(\boldsymbol{x}_n))^{-1}F(\boldsymbol{x}_n), \tag{4.15}$$

where $\boldsymbol{P} = (F')^{-1}F''(F')^{-1}F$: for $\alpha = 1$ the technique in (4.15) is called Chebyshev method and the second order Frechet derivative is defined as $F''(\boldsymbol{x})\boldsymbol{v} = \frac{\partial}{\partial \boldsymbol{x}}\left(\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}}\boldsymbol{v}\right)$. The computational cost of the second-order Frechet derivative and of the related matrix inversion is high so from a practical point of view the considered method is not efficient.

The prime goal of the present chapter is to address the computational cost issue of higher-order Frechet derivatives for a particular class of systems of nonlinear equations associated with ODEs $L(x(t)) + f(x(t)) = g(t)$ and use the computed Frechet derivatives for the construction of iterative methods with better rate of convergence. The efficiency index is also compared with the general iterative method for systems of nonlinear equations.

4.2 The proposed method

The ODE (4.1) can be written as

$$F(\boldsymbol{x}) = A\boldsymbol{x} + f(\boldsymbol{x}) - \boldsymbol{g} = 0, \tag{4.16}$$

where the symbols have the same meaning as described in (4.11), $f(.)$ is a sufficient differentiable function and A could be the function of independent variable $\boldsymbol{t}$. The higher-order Frechet derivatives can be calculated as:

$$F'(\boldsymbol{x}) = A + diag(f'(\boldsymbol{x})), \tag{4.17}$$

$$F''(\boldsymbol{x}) = diag(f''(\boldsymbol{x})). \tag{4.18}$$

Clearly $F''(\boldsymbol{x})$ is a diagonal matrix and the count for function evaluations is same as in $f(\boldsymbol{x})$. The proposed iterative method is:

$$\begin{cases} \boldsymbol{y}_k = \boldsymbol{x}_k - (F'(\boldsymbol{x}_k))^{-1}\left[F(\boldsymbol{x}_k) + \frac{1}{2}F''(\boldsymbol{x}_k)\left(F'(\boldsymbol{x}_k)^{-1}F(\boldsymbol{x}_k)\right)^2\right], \\ T_k = (F'(\boldsymbol{x}_k))^{-1}F'(\boldsymbol{y}_k), \\ S_k = [3 - 3T_k + T_k^2](F'(\boldsymbol{x}_k))^{-1}, \\ \boldsymbol{z}_k = \boldsymbol{y}_k - S_k F(\boldsymbol{y}_k), \\ \boldsymbol{x}_{k+1} = \boldsymbol{z}_k - S_k F(\boldsymbol{z}_k). \end{cases} \tag{4.19}$$

We just require one matrix inversion of Jacobian. For large systems of linear equations, the matrix inversion is not efficient so one may use LU-decomposition or

GMRES-type iterative methods [37]. We can rewrite (4.19) in an efficient way:

$$\begin{cases} F'(\boldsymbol{x}_k)\boldsymbol{\phi}_k = F(\boldsymbol{x}_k), \\ F'(\boldsymbol{x}_k)\boldsymbol{\psi}_k = F''(\boldsymbol{x}_k)\boldsymbol{\phi}_k{}^2, \\ \boldsymbol{y}_k = \boldsymbol{x}_k - \boldsymbol{\phi}_k - \frac{1}{2}\boldsymbol{\psi}_k, \\ F'(\boldsymbol{x}_k)\boldsymbol{\varphi}_k = F(\boldsymbol{y}_k), \\ F'(\boldsymbol{x}_k)T_k = F'(\boldsymbol{y}_k), \\ W_k = 3 - 3T_k + T_k^2, \\ \boldsymbol{z}_k = \boldsymbol{y}_k - W_k\boldsymbol{\varphi}_k, \\ F'(\boldsymbol{x}_k)\boldsymbol{\lambda}_k = F(\boldsymbol{z}_k), \\ \boldsymbol{x}_{k+1} = \boldsymbol{z}_k - W_k\boldsymbol{\lambda}_k. \end{cases} \tag{4.20}$$

The convergence-rate of the iteration in (4.20) is nine. We can split (4.20) into component schemes as follows:

$$\begin{cases} F'(\boldsymbol{x}_k)\boldsymbol{\phi}_k = F(\boldsymbol{x}_k), \\ F'(\boldsymbol{x}_k)\boldsymbol{\psi}_k = F''(\boldsymbol{x}_k)\boldsymbol{\phi}_k{}^2, \\ \boldsymbol{y}_k = \boldsymbol{x}_k - \boldsymbol{\phi}_k - \frac{1}{2}\boldsymbol{\psi}_k, \end{cases} \tag{4.21}$$

and

$$\begin{cases} F'(\boldsymbol{x}_k)\boldsymbol{\phi}_k = F(\boldsymbol{x}_k), \\ F'(\boldsymbol{x}_k)\boldsymbol{\psi}_k = F''(\boldsymbol{x}_k)\boldsymbol{\phi}_k{}^2, \\ \boldsymbol{y}_k = \boldsymbol{x}_k - \boldsymbol{\phi}_k - \frac{1}{2}\boldsymbol{\psi}_k, \\ F'(\boldsymbol{x}_k)\boldsymbol{\varphi}_k = F(\boldsymbol{y}_k), \\ F'(\boldsymbol{x}_k)T_k = F'(\boldsymbol{y}_k), \\ W_k = 3 - 3T_k + T_k^2, \\ \boldsymbol{z}_k = \boldsymbol{y}_k - W_k\boldsymbol{\varphi}_k. \end{cases} \tag{4.22}$$

The convergence-rates of the methods given in (4.21) and (4.22) are three and six, respectively. Our proposal for the multi-step iterative methods is the following:

$$\begin{cases} F'(\boldsymbol{x}_k)\boldsymbol{\phi}_k = F(\boldsymbol{x}_k), \\ F'(\boldsymbol{x}_k)\boldsymbol{\psi}_k = F''(\boldsymbol{x}_k)\boldsymbol{\phi}_k{}^2, \\ \boldsymbol{z}_{\mathbf{1}k} = \boldsymbol{x}_k - \boldsymbol{\phi}_k - \frac{1}{2}\boldsymbol{\psi}_k, \\ F'(\boldsymbol{x}_k)T_k = F'(\boldsymbol{z}_{\mathbf{1}k}), \\ W_k = 3 - 3T_k + T_k^2, \\ F'(\boldsymbol{x}_k)\boldsymbol{\lambda}_{\mathbf{1}k} = F(\boldsymbol{z}_{\mathbf{1}k}), \\ \boldsymbol{z}_{\mathbf{2}k} = \boldsymbol{z}_{\mathbf{1}k} - W_k\boldsymbol{\lambda}_{\mathbf{1}k}, \\ F'(\boldsymbol{x}_k)\boldsymbol{\lambda}_{\mathbf{2}k} = F(\boldsymbol{z}_{\mathbf{2}k}), \\ \boldsymbol{z}_{\mathbf{3}k} = \boldsymbol{z}_{\mathbf{2}k} - W_k\boldsymbol{\lambda}_{\mathbf{2}k}, \\ F'(\boldsymbol{x}_k)\boldsymbol{\lambda}_{\mathbf{3}k} = F(\boldsymbol{z}_{\mathbf{3}k}), \\ \boldsymbol{z}_{\mathbf{4}k} = \boldsymbol{z}_{\mathbf{3}k} - W_k\boldsymbol{\lambda}_{\mathbf{3}k}, \\ \vdots \\ F'(\boldsymbol{x}_k)\boldsymbol{\lambda}_{\boldsymbol{m-1}k} = F(\boldsymbol{z}_{\boldsymbol{m-1}k}), \\ \boldsymbol{x}_{k+1} = \boldsymbol{z}_{\boldsymbol{m-1}k} - W_k\boldsymbol{\lambda}_{\boldsymbol{m-1}k}. \end{cases} \tag{4.23}$$

We claim the convergence-order of (4.23) is $3m$. It is noticeable that W_k and the LU-factors of $F'(\boldsymbol{x}_k)$ are fixed for each k, which makes the proposed multi-step iterative scheme computationally very efficient.

4.3 Convergence analysis

In this section, first we will prove that the local convergence-order of the technique reported in (4.20) is nine and later we will establish a proof for the multi-step iterative scheme (4.23), by using mathematical induction. We used symbolic non-commutative algebra package of Mathematica software for the symbolic calculation in the constructed proof.

Theorem 4.1. *Let $F : \Gamma \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be sufficiently Frechet differentiable on an open convex neighborhood Γ of $\boldsymbol{x}^* \in \mathbb{R}^n$ with $F(\boldsymbol{x}^*) = 0$ and $det(F'(\boldsymbol{x}^*)) \neq 0$. Then the sequence $\{\boldsymbol{x}_k\}$ generated by the iterative scheme (4.20) converges to $\boldsymbol{x}^*$ with local order*

of convergence nine, and produces the following error equation

$$\boldsymbol{e}_{k+1} = \mathbf{L}\boldsymbol{e}_k{}^9 + O(\boldsymbol{e}_k{}^{10}), \tag{4.24}$$

where $\boldsymbol{e}_k = \boldsymbol{x_k} - \boldsymbol{x}^*$, $\boldsymbol{e}_k{}^p = \overbrace{(\boldsymbol{e}_k, \boldsymbol{e}_k, \cdots, \boldsymbol{e}_k)}^{p\text{-times}}$ *and* $L = (-2C_2C_3C_2C_3^2 + 4C_2C_3C_2C_3C_2^2 + 12C_2^4C_3^2 - 120C_2^6C_3 + 240C_2^8 - 24C_2^4C_3C_2^2 - 40C_2C_3C_2^5 + 20C_2C_3C_2^3C_3)\boldsymbol{e}^9$ *is a* $p-$ *linear function i.e.* $\mathbf{L} \in \mathscr{L}\overbrace{(\mathbb{R}^n, \mathbb{R}^n, \cdots, \mathbb{R}^n)}^{p\text{-times}}$ *and* $\mathbf{L}\boldsymbol{e}_k{}^p \in \mathbb{R}^n$.

Proof. Let $F : \Gamma \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be sufficiently Frechet differentiable function in Γ. The qth Frechet derivative of F at $v \in \mathbb{R}^n$, $q \geq 1$, is the $q-linear$ function $F^{(q)}(v) : \overbrace{\mathbb{R}^n\mathbb{R}^n\cdots\mathbb{R}^n}^{\text{q-times}}$ such that $F^{(q)}(v)(u_1, u_2, \cdots, u_q) \in \mathbb{R}^n$ [154]. The Taylor's series expansion of $F(\boldsymbol{x}_k)$ around $\boldsymbol{x}^*$ can be written as:

$$\begin{aligned}
F(\boldsymbol{x}_k) &= F(\boldsymbol{x}^* + \boldsymbol{x}_k - \boldsymbol{x}^*) = F(\boldsymbol{x}^* + \boldsymbol{e}_k), && (4.25)\\
&= F(\boldsymbol{x}^*) + F'(\boldsymbol{x}^*)\boldsymbol{e}_k + \frac{1}{2!}F''(\boldsymbol{x}^*)\boldsymbol{e}_k{}^2 + \frac{1}{3!}F^{(3)}(\boldsymbol{x}^*)\boldsymbol{e}_k{}^3 + \cdots, && (4.26)\\
&= F'(\boldsymbol{x}^*)\Big[\boldsymbol{e}_k + \frac{1}{2!}F'(\boldsymbol{x}^*)^{-1}F''(\boldsymbol{x}^*)\boldsymbol{e}_k{}^2 + \frac{1}{3!}F'(\boldsymbol{x}^*)^{-1}F^{(3)}(\boldsymbol{x}^*)\boldsymbol{e}_k{}^3 + \cdots\Big], && (4.27)\\
&= C_1\Big[\boldsymbol{e}_k + C_2\boldsymbol{e}_k{}^2 + C_3\boldsymbol{e}_k{}^3 + \cdots\Big], && (4.28)
\end{aligned}$$

where $C_1 = F'(\boldsymbol{x}^*)$ and $C_s = \frac{1}{s!}F'(\boldsymbol{x}^*)^{-1}F^{(s)}(\boldsymbol{x}^*)$ for $s \geq 2$. From (4.28), we can calculate the Frechet derivative of F:

$$F'(\boldsymbol{x}_k) = C_1\Big[I + 2C_2\boldsymbol{e}_k + 3C_3\boldsymbol{e}_k{}^2 + 4C_3\boldsymbol{e}_k{}^3 + \cdots\Big], \tag{4.29}$$

where I is the identity matrix. Furthermore, we calculate the inverse of the Jacobian matrix

$$\begin{aligned}F'(\boldsymbol{x}_k)^{-1} &= \big[I - 2C_2\boldsymbol{e}_k + (4C_2^2 - 3C_3)\boldsymbol{e}_k^2 + (-8C_2^3 + 6C_3C_2 + 6C_2C_3 - 4C_4)\boldsymbol{e}_k^3 \\ &\quad + (-12C_3C_2^2 - 12C_2^2C_3 - 12C_2C_3C_2 + 8C_4C_2 + 9C_3^2 + 8C_2C_4 + 16C_2^4 \\ &\quad - 5C_5)\boldsymbol{e}_k^4 + (-16C_4C_2^2 - 18C_3C_2C_3 - 18C_3^2C_2 - 18C_2C_3^2 - 16C_2^2C_4 \\ &\quad - 16C_2C_4C_2 + 24C_2^3C_3 + 24C_2^2C_3C_2 + 24C_2C_3C_2^2 - 32C_2^5 + 10C_5C_2 \\ &\quad + 12C_4C_3 + 12C_3C_4 + 10C_2C_5 + 24C_3C_2^3 - 6C_6)\boldsymbol{e}_k^5 + (-24C_3C_4C_2 \\ &\quad - 24C_3C_2C_4 - 27C_3^3 - 48C_3C_2^4 + 64C_2^6 - 24C_4C_3C_2 - 24C_4C_2C_3 \\ &\quad - 20C_2C_5C_2 - 24C_2C_4C_3 + 32C_4C_2^3 - 24C_2C_3C_4 - 20C_2^2C_5 \\ &\quad + 36C_2^2C_3^2 + 32C_2^3C_4 - 48C_2^2C_3C_2^2 + 36C_2C_3C_2C_3 - 48C_2^3C_3C_2 \\ &\quad - 48C_2^4C_3 - 48C_2C_3C_2^3 + 36C_3C_2^2C_3 + 36C_3C_2C_3C_2 + 12C_2C_6 \\ &\quad + 16C_4^2 + 15C_3C_5 + 15C_5C_3 + 12C_6C_2 + 36C_2C_3^2C_2 + 32C_2C_4C_2^2 \\ &\quad - 7C_7 + 36C_3^2C_2^2 + 32C_2^2C_4C_2 - 20C_5C_2^2)\boldsymbol{e}_k^6 + \cdots\big]C_1^{-1} \end{aligned} \tag{4.30}$$

By multiplying $F'(x_k)^{-1}$ and $F(x_k)$, we obtain $\boldsymbol{\phi}_k$:

$$\begin{aligned}\boldsymbol{\phi}_k &= \boldsymbol{e}_k - C_2\boldsymbol{e}_k^2 + (2C_2^2 - 2C_3)\boldsymbol{e}_k^3 + (4C_2C_3 + 3C_3C_2 - 3C_4 - 4C_2^3)\boldsymbol{e}_k^4 \\ &\quad + (8C_2^4 - 6C_3C_2^2 + 6C_3^2 + 6C_2C_4 + 4C_4C_2 - 4C_5 - 8C_2^2C_3 - 6C_2C_3C_2)\boldsymbol{e}_k^5 \\ &\quad + (-12C_3C_2C_3 - 8C_2C_4C_2 - 12C_2^2C_4 - 12C_2C_3^2 + 12C_3C_2^3 - 16C_2^5 \\ &\quad - 9C_3^2C_2 + 8C_2C_5 + 9C_3C_4 + 8C_4C_3 + 5C_5C_2 - 5C_6 - 8C_4C_2^2 + 16C_2^3C_3 \\ &\quad + 12C_2^2C_3C_2 + 12C_2C_3C_2^2)\boldsymbol{e}_k^6 + \cdots. \end{aligned} \tag{4.31}$$

The expression for $\boldsymbol{\psi}_k$ is the following:

$$\begin{aligned}\boldsymbol{\psi}_k &= 2C_2\boldsymbol{e}_k^2 + (-8C_2^2 + 6C_3)\boldsymbol{e}_k^3 + (26C_2^3 - 18C_3C_2 - 20C_2C_3 + 12C_4)\boldsymbol{e}_k^4 \\ &\quad + (52C_2C_3C_2 + 54C_3C_2^2 - 32C_4C_2 - 76C_2^4 + 60C_2^2C_3 - 42C_3^2 - 36C_2C_4 \\ &\quad + 20C_5)\boldsymbol{e}_k^5 + (120C_3C_2C_3 + 86C_2C_4C_2 + 102C_2^2C_4 + 116C_2C_3^2 \\ &\quad - 150C_3C_2^3 + 208C_2^5 + 102C_3^2C_2 - 56C_2C_5 - 72C_3C_4 - 72C_4C_3 - 50C_5C_2 \\ &\quad + 30C_6 + 92C_4C_2^2 - 168C_2^3C_3 - 142C_2^2C_3C_2 - 146C_2C_3C_2^2)\boldsymbol{e}_k^6 + \cdots. \end{aligned} \tag{4.32}$$

By using (4.31) and (4.32), we obtain

$$\begin{aligned}\boldsymbol{y}_k-\boldsymbol{x}^*=\boldsymbol{e}_k-\boldsymbol{\phi}_k-\frac{1}{2}\boldsymbol{\psi}_k=(2C_2^2-C_3)\boldsymbol{e}_k^3+(6C_3C_2+6C_2C_3-3C_4-9C_2^3)\boldsymbol{e}_k^4\\+(12C_4C_2+15C_3^2+12C_2C_4-6C_5+30C_2^4-22C_2^2C_3-20C_2C_3C_2\\-21C_3C_2^2)\boldsymbol{e}_k^5+(-88C_2^5+20C_5C_2+28C_4C_3+27C_3C_4+20C_2C_5-10C_6\\+63C_3C_2^3-48C_3C_2C_3-42C_3^2C_2-46C_2C_3^2-39C_2^2C_4-35C_2C_4C_2\\+68C_2^3C_3+59C_2^2C_3C_2+61C_2C_3C_2^2-38C_4C_2^2)\boldsymbol{e}_k^6+\cdots\end{aligned}\tag{4.33}$$

$\boldsymbol{\varphi}_k$ and T_k are computed from (4.30), and thus T_k produces:

$$\begin{aligned}W_k=I+2C_2\boldsymbol{e}_k+3C_3\boldsymbol{e}_k^2+(2C_2C_3+4C_4-12C_2^3)\boldsymbol{e}_k^3+(6C_2C_4-24C_2^2C_3\\-12C_3C_2^2+42C_2^4+5C_5-20C_2C_3C_2)\boldsymbol{e}_k^4+(-18C_3C_2C_3-28C_2C_4C_2\\-40C_2^2C_4-42C_2C_3^2+48C_3C_2^3-72C_2^5-18C_3^2C_2+12C_2C_5+6C_6\\-16C_4C_2^2+72C_2^3C_3+56C_2^2C_3C_2+58C_2C_3C_2^2)\boldsymbol{e}_k^5+(-24C_3C_4C_2\\-24C_3C_2C_4-30C_3^3-108C_3C_2^4-48C_2^6-24C_4C_3C_2-24C_4C_2C_3\\-36C_2C_5C_2-62C_2C_4C_3+64C_4C_2^3-70C_2C_3C_4-60C_2^2C_5+116C_2^2C_3^2\\+102C_2^3C_4-74C_2^2C_3C_2^2+112C_2C_3C_2C_3-38C_2^3C_3C_2-106C_2^4C_3\\-106C_2C_3C_2^3+66C_3C_2^2C_3+60C_3C_2C_3C_2+20C_2C_6+84C_2C_3^2C_2\\+68C_2C_4C_2^2+7C_7+78C_3^2C_2^2+62C_2^2C_4C_2-20C_5C_2^2)\boldsymbol{e}_k^6+\cdots.\end{aligned}\tag{4.34}$$

By combining relations (4.33) and (4.34), we find an expression for $\boldsymbol{z}_k$, namely

$$\begin{aligned}\boldsymbol{z}_k-\boldsymbol{x}^*=(C_2C_3^2+20C_2^5-10C_2^3C_3-2C_2C_3C_2^2)\boldsymbol{e}_k^6+(-204C_2^6+24C_3C_2^4+3C_2C_4C_3\\+3C_2C_3C_4-18C_2^2C_3^2-30C_2^3C_4+36C_2^2C_3C_2^2-24C_2C_3C_2C_3+60C_2^3C_3C_2\\+117C_2^4C_3+45C_2C_3C_2^3-12C_3C_2^2C_3-6C_2C_3^2C_2-6C_2C_4C_2^2)\boldsymbol{e}_k^7+(32C_4C_2^4\\+36C_3C_2C_3C_2^2+72C_3C_2^2C_3C_2+6C_2C_5C_3+144C_3C_2^3C_3+6C_2C_3C_5\\+9C_2(C_4^2)+108C_2^2C_3^2C_2+56C_2^2C_4C_2^2+1161C_2^7-382C_2^3C_3C_2^2+83C_2C_4C_2^3\\+36C_3^2C_2^3+215C_2C_3C_2^2C_3+128C_2C_3C_2C_3C_2-28C_2^2C_4C_3-54C_2^2C_3C_4\\-60C_2^3C_5-12C_2C_3C_4C_2-66C_2C_3C_2C_4-48C_2(C_3^3)-18C_2C_4C_3C_2\\-46C_2C_4C_2C_3-12C_2C_5C_2^2+236C_2^3C_3^2-18C_3^2C_2C_3-36C_3C_2^2C_4-18C_3C_2C_3^2\\+120C_2^3C_4C_2+192C_2^2C_3C_2C_3+87C_2C_3^2C_2^2+291C_2^4C_4-542C_2^4C_3C_2\\-736C_2^5C_3-16C_4C_2^2C_3-252C_3C_2^5-330C_2^2C_3C_2^3-362C_2C_3C_2^4)\boldsymbol{e}_k^8+\cdots.\end{aligned}\tag{4.35}$$

From (4.30) and (4.35), we have

$$\begin{aligned}\boldsymbol{\lambda}_k &= (C_2C_3^2 + 20C_2^5 - 10C_2^3C_3 - 2C_2C_3C_2^2)\boldsymbol{e}_k^6 + (-20C_2^2C_3^2 - 244C_2^6 + 137C_2^4C_3 \\ &+ 40C_2^2C_3C_2^2 + 24C_3C_2^4 + 3C_2C_4C_3 + 3C_2C_3C_4 - 30C_2^3C_4 - 24C_2C_3C_2C_3 \\ &+ 60C_2^3C_3C_2 + 45C_2C_3C_2^3 - 12C_3C_2^2C_3 - 6C_2C_3^2C_2 - 6C_2C_4C_2^2)\boldsymbol{e}_k^7 \\ &+ (32C_4C_2^4 + 42C_3C_2C_3C_2^2 + 72C_3C_2^2C_3C_2 + 6C_2C_5C_3 + 174C_3C_2^3C_3 \\ &+ 6C_2C_3C_5 + 9C_2(C_4^2) + 120C_2^2C_3^2C_2 + 68C_2^2C_4C_2^2 + 1649C_2^7 - 462C_2^3C_3C_2^2 \\ &+ 83C_2C_4C_2^3 + 36C_3^2C_2^3 + 239C_2C_3C_2^2C_3 + 128C_2C_3C_2C_3C_2 - 34C_2^2C_4C_3 \\ &- 60C_2^2C_3C_4 - 60C_2^3C_5 - 12C_2C_3C_4C_2 - 66C_2C_3C_2C_4 - 48C_2(C_3^3) - 18C_2C_4C_3C_2 \\ &- 46C_2C_4C_2C_3 - 12C_2C_5C_2^2 + 276C_2^3C_3^2 - 18C_3^2C_2C_3 - 36C_3C_2^2C_4 \\ &- 21C_3C_2C_3^2 + 120C_2^3C_4C_2 + 240C_2^2C_3C_2C_3 + 87C_2C_3^2C_2^2 + 351C_2^4C_4 \\ &- 662C_2^4C_3C_2 - 1010C_2^5C_3 - 16C_4C_2^2C_3 - 312C_3C_2^5 - 420C_2^2C_3C_2^3 \\ &- 410C_2C_3C_2^4)\boldsymbol{e}_k^8 + \cdots . \end{aligned} \tag{4.36}$$

Finally, by using (4.34), (4.35) and (4.36), we obtain the error equations reported below and the proof is completed

$$\begin{aligned}\boldsymbol{e}_{k+1} &= (-2C_2C_3C_2C_3^2 + 4C_2C_3C_2C_3C_2^2 + 12C_2^4C_3^2 - 120C_2^6C_3 + 240C_2^8 - 24C_2^4C_3C_2^2 \\ &- 40C_2C_3C_2^5 + 20C_2C_3C_2^3C_3)\boldsymbol{e}_k^9 + O(\boldsymbol{e}_k^{10}). \end{aligned} \tag{4.37}$$

□

Theorem 4.2. *The multi-step iterative scheme (4.23) has the local convergence-order 3m, using m evaluations of a sufficiently differentiable function F, two first-order Frechet derivative F' and one second-order Frechet derivate F'' per full-cycle.*

Proof. The proof is established from mathematical induction. For $m = 1$ the multi-step iterative scheme given in (4.23) corresponds to the iterative scheme in (4.21). The convergence-order of (4.21) is produced in (4.33) which is three. Similarly for $m = 2,\ 3$ the multi-step scheme (4.23) reduced to the iterative schemes (4.22) and (4.20), respectively. The convergence-orders are calculated in (4.35) and (4.37) which are $3m = 32 = 6$ and $3m = 33 = 9$ respectively. Consequently our claim concerning the convergence-order $3m$ is true for $m = 1,\ 2,\ 3$.

We assume that our claim is true for $m = s > 3$, i.e., the convergence-order of (4.23) is $3s$. The sth-step and $(s-1)th$-step of iterative scheme(4.23) can be written as:

$$Frozen - factor = W_k F'(\boldsymbol{x}_k)^{-1}, \tag{4.38}$$

$$\boldsymbol{z_{s-1}}_k = \boldsymbol{z_{s-2}}_k - (Frozen - factor)F(\boldsymbol{z_{s-2}}_k), \tag{4.39}$$

$$\boldsymbol{z_s}_k = \boldsymbol{z_{s-1}}_k - (Frozen - factor)F(\boldsymbol{z_{s-1}}_k), \tag{4.40}$$

where $Frozen - factor$ is the product of W_k and $F'(\boldsymbol{x}_k)^{-1}$ which are calculated just once in one full-cycle of iterative method. The enhancement in the convergence-order of (4.23) from $(s-1)th$-step to sth-step is $3s-3(s-1) = 3$. Now we write the $(s+1)th$-step of (4.23):

$$\boldsymbol{z_{s+1}}_k = \boldsymbol{z_s}_k - (Frozen - factor)F(\boldsymbol{z_s}_k). \tag{4.41}$$

The increment in the convergence-order of (4.23), due to $(s+1)th$-step, is exactly three, because the use of the $Frozen - factor$ adds an additive constant in the convergence-order[15]. Finally the convergence-order after the addition of the $(s+1)th$-step is $3s+3 = 3(s+1)$, which completes the proof. □

4.4 Efficiency index

For the purpose of comparison we write the multi-step extensions of iterative schemes (4.13) and (4.14) as follows:

$$\begin{cases} \boldsymbol{y}^{(k)} = \boldsymbol{x}^{(k)} - \frac{2}{3}F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{x}^{(k)}), \\ \boldsymbol{t}^{(k)} = \frac{1}{2}(3F'(\boldsymbol{y}^{(k)}) - F'(\boldsymbol{x}^{(k)}))^{-1}(3F'(\boldsymbol{y}^{(k)}) + F'(\boldsymbol{x}^{(k)})), \\ \boldsymbol{z}^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{t}^{(k)}F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{x}^{(k)}), \\ \boldsymbol{w}^{(k)} = \boldsymbol{z}^{(k)} - (\boldsymbol{t}^{(k)})^2F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{z}^{(k)}), \\ \boldsymbol{x}^{(k+1)} = \boldsymbol{w}^{(k)} - (\boldsymbol{t}^{(k)})^2F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{w}^{(k)}), \end{cases} \tag{4.42}$$

and

$$\begin{cases} \boldsymbol{y}^{(k)} = \boldsymbol{x}^{(k)} - \frac{2}{3}F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{x}^{(k)}), \\ \boldsymbol{t}^{(k)} = F'(\boldsymbol{x}^{(k)})^{-1}F'(\boldsymbol{y}^{(k)}), \\ \boldsymbol{z}^{(k)} = \boldsymbol{x}^{(k)} - \left(\frac{23}{8}I - 3\boldsymbol{t}^{(k)} + \frac{9}{8}\boldsymbol{t}^{(k)^2}\right)F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{x}^{(k)}), \\ \boldsymbol{w}^{(k)} = \boldsymbol{z}^{(k)} - \left(\frac{5}{2}I - \frac{3}{2}t^{(k)}\right)F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{z}^{(k)}), \\ \boldsymbol{x}^{(k+1)} = \boldsymbol{w}^{(k)} - \left(\frac{5}{2}I - \frac{3}{2}\boldsymbol{t}^{(k)}\right)F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{w}^{(k)}). \end{cases} \tag{4.43}$$

Both multi-step iterative schemes (4.42) and (4.43) have convergence-order eight. Further steps can be added with the inclusion of the frozen factors $\boldsymbol{\zeta}^{(k)} - (\boldsymbol{t}^{(k)})^2F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{\zeta}^{(k)})$ and $\boldsymbol{\zeta}^{(k)} - \left(\frac{5}{2}I - \frac{3}{2}\boldsymbol{t}^{(k)}\right)F'(\boldsymbol{x}^{(k)})^{-1}F(\boldsymbol{\zeta}^{(k)})$ in (4.42) and (4.43), respectively, in order to increase the convergence-order. Let us discuss the computational cost of the considered multi-step iterative schemes. It is well-know that the LU-decomposition requires $\frac{2n^3}{3}$ flops and $2n^2$ flops are needed to solve the two resulting triangular systems, when the right-hand side is a vector. If the right-hand side is a matrix then $2n^3$, or approximately n^3 flops (as taken in this chapter) are required to solve two triangular systems. The scalar function evaluations in $F(\boldsymbol{x})$, $F'(\boldsymbol{x})$ and $F''(\boldsymbol{x})$ are n, because, in our setting, the Frechet derivatives are diagonal matrices of order n. In Table 4.1, we depicted the computational cost and efficiency index of different multi-step methods. The flops-like efficiency indices of the considered multi-steps methods are shown in Figure 4.1. Clearly our proposed multi-step method has better flops-like efficiency index [37] in comparison with others.

Iterative methods	(42)	(43)	(23)
Number of steps (m)	4	4	3
Rate of convergence	8	8	9
Number of functional evaluations	$5n$	$5n$	$6n$
The classical efficiency index	$2^{1/(5n)}$	$2^{1/(5n)}$	$2^{1/(6n)}$
Number of LU factorizations	2	1	1
Cost of LU factorizations	$\frac{4n^3}{3}$	$\frac{2n^3}{3}$	$\frac{2n^3}{3}$
Cost of linear systems	$\frac{7n^3}{3}+6n^2$	$\frac{5n^3}{3}+6n^2$	$\frac{5n^3}{3}+8n^2$
Flops-like efficiency index	$8^{1/(\frac{7n^3}{3}+6n^2+5n)}$	$8^{1/(\frac{5n^3}{3}+6n^2+5n)}$	$9^{1/(\frac{5n^3}{3}+8n^2+6n)}$

TABLE 4.1: Comparison of efficiency indices for different for multi-step methods

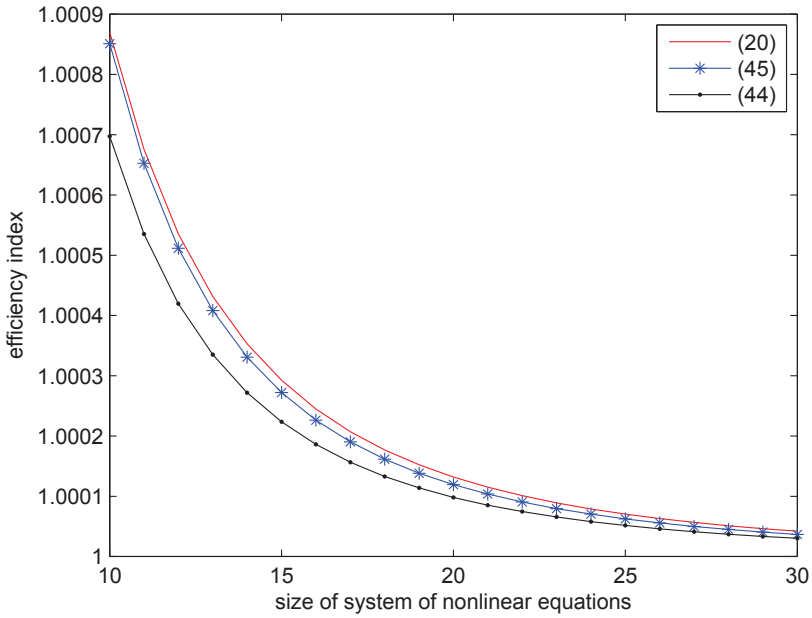

FIGURE 4.1: Flops-like efficiency indices for different multi-step methods

4.5 Numerical testing

In order to check the validity and efficiency of the proposed multi-step method (4.23), we select two boundary value and one initial value problem. The first is the Bratu-problem

$$x''(t) + \alpha e^{x(t)} = 0, \tag{4.44}$$

with boundary conditions given by $x(0) = 0$, $x(1) = 0$. The second is the Frank-Kamenetzkii-problem

$$x''(t) + \frac{\kappa}{t}x'(t) + \alpha e^{x(t)} = 0, \tag{4.45}$$

with boundary conditions given by $x'(0) = 0$, $x(1) = 0$. The closed form solution of (4.44) [181] and (4.45) for $\kappa = 1$ [182] are

$$\begin{cases} x(t) = -2log\left[\dfrac{cosh\left((t-\frac{1}{2})\frac{\theta}{2}\right)}{cosh\left(\frac{\theta}{4}\right)}\right], \\ \theta = \sqrt{2\alpha}\, cosh\left(\dfrac{\theta}{4}\right), \end{cases} \tag{4.46}$$

and

$$\begin{cases} c_1 = log\left[2(4-\alpha) \pm 4\sqrt{2(2-\alpha)}\right],\ x(t) = log\left[\frac{16e^{c_1}}{(2\alpha+e^{c_1}t^2)^2}\right], \\ c_2 = log\left[\frac{4-\alpha\pm 2\sqrt{2(2-\alpha)}}{2\alpha^2}\right],\ x(t) = log\left[\frac{16e^{c_2}}{(1+2\alpha e^{c_2}t^2)^2}\right], \end{cases} \tag{4.47}$$

respectively. More discussion about Bratu and Frank-Kamenetzkii can be found in [5, 6]. The last one is Lene-Emden problem

$$x''(t) + \frac{2}{t}x'(t) + x^p(t) = 0, \quad 0 < x < \infty, \tag{4.48}$$

with initial conditions $x(0) = 1$ and $x'(0) = 0$. The closed form solution of (4.48) for $p = 5$ is

$$x(t) = \left(1 + \frac{x^3}{3}\right)^{-1/2}. \tag{4.49}$$

We use the Chebyshev pseudo-spectral collocation method [142] for the approximation of the boundary-value problems (4.44),(4.45) and (4.48). For the verification of convergence-order, we use the following definition for the computational convergence-order (COC):

$$COC \approx \frac{log\left[Max(|\boldsymbol{x}_{k+2} - \boldsymbol{x}^*|)/Max(|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*|)\right]}{log\left[Max(|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*|)/Max(|\boldsymbol{x}_k - \boldsymbol{x}^*|)\right]}, \tag{4.50}$$

where $Max(|\boldsymbol{x}_{k+2} - \boldsymbol{x}^*|)$ is maximum absolute error.

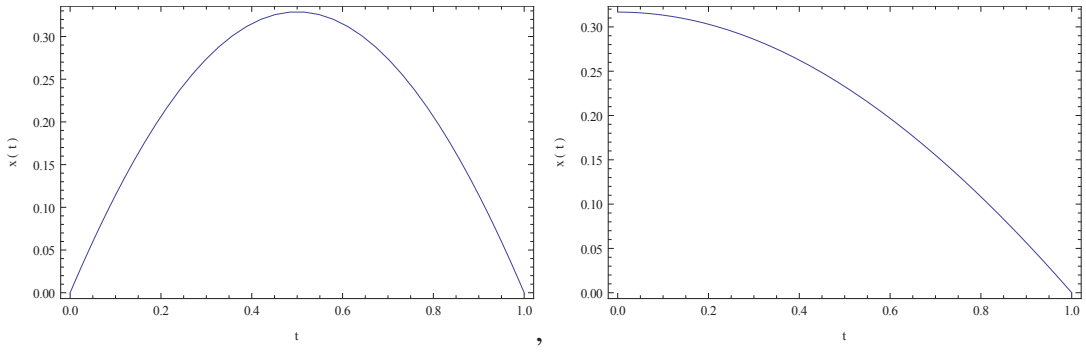

FIGURE 4.2: solution-curve of the Bratu-problem for $\alpha = 1$ (left), solution-curve of the Frank-Kamenetzkii-problem for $\alpha = 1,\ k = 1$ (right)

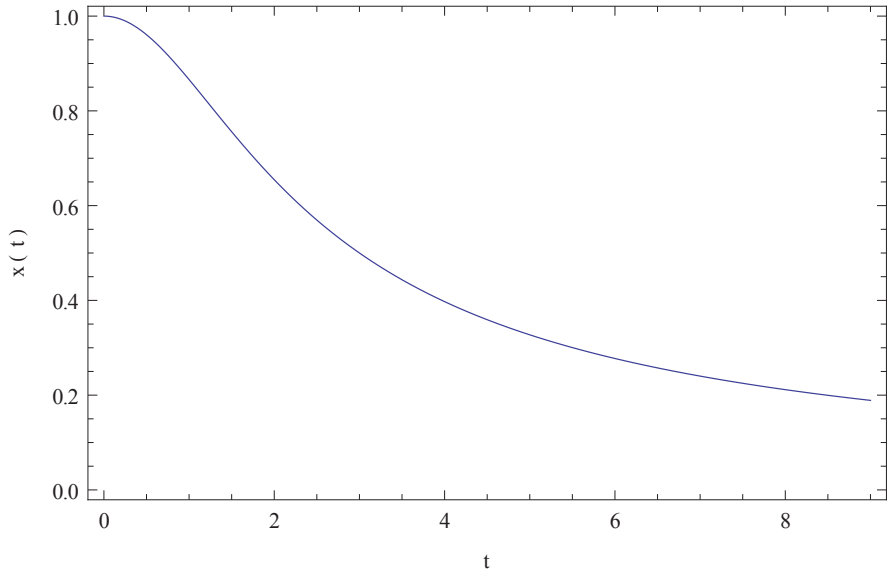

FIGURE 4.3: solution-curve of the Lene-Emden for $p = 5$, Domain=[0, 9]

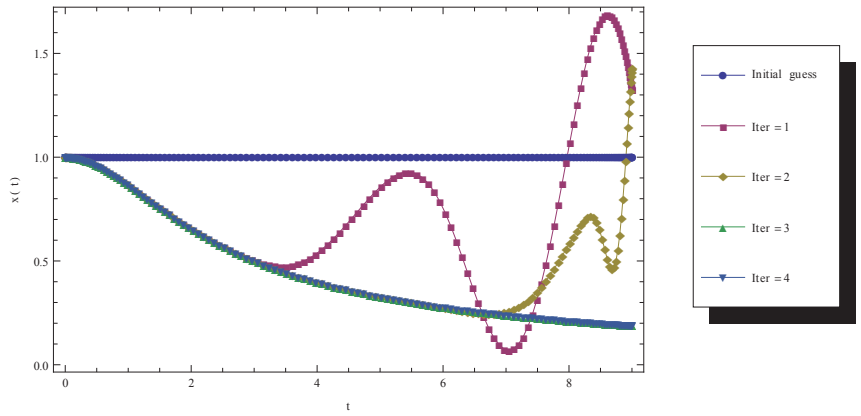

FIGURE 4.4: Convergence behavior of iterative method (4.23) for the Lene-Emden problem ($p = 5$, Domain=[0, 9])

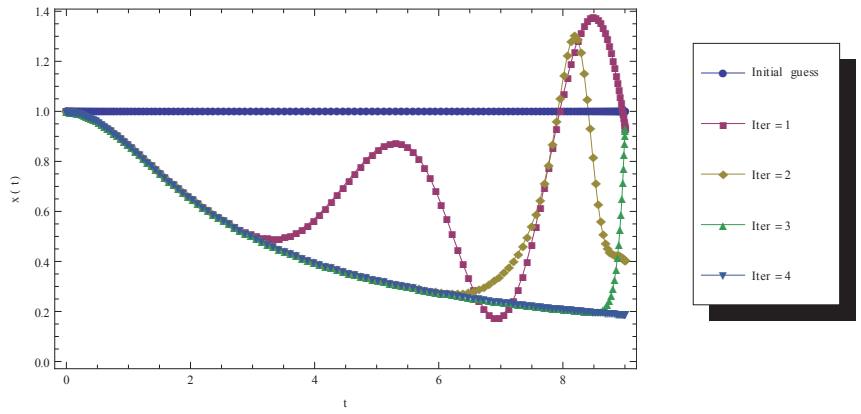

FIGURE 4.5: Convergence behavior of iterative method (4.43) for the Lene-Emden problem ($p = 5$, Domain=[0, 9])

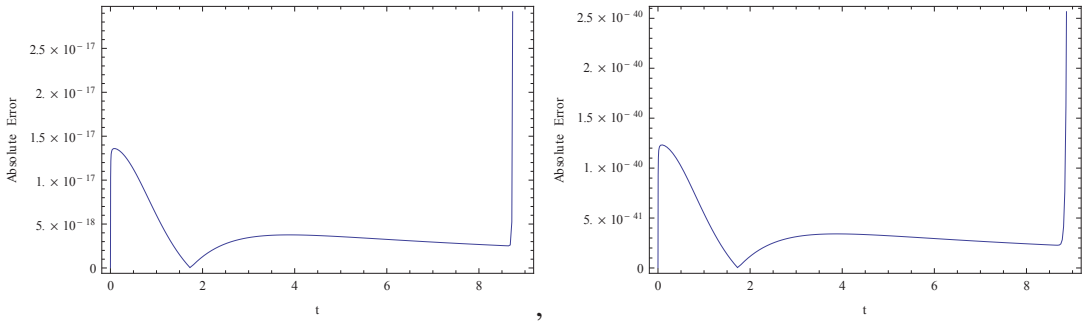

FIGURE 4.6: Absolute error curve for the Lene-Emden problem, left(4.23) and right (4.43)

Iterative methods	(42)	(43)	(23)
α	1	1	1
Size of problem	150	150	150
Number of steps (m)	4	4	3
Number of iterations	2	2	2
Max absolute error	$5.29e-117$	$1.87e-110$	$3.77e-142$
Execution time	29.60	20	19.61
α	2	2	2
Size of problem	200	200	200
Number of steps (m)	4	4	3
Number of iterations	2	2	2
Max absolute error	$1.66e-109$	$3.48e-99$	$5.33e-127$
Execution time	69.41	46.70	46.39
α	3	3	3
Size of problem	150	150	150
Number of steps (m)	4	4	3
Number of iterations	2	2	2
Max absolute error	$1.65e-46$	$8.41e-37$	$5.50e-47$
Execution time	29.49	20.76	20.69

TABLE 4.2: Comparison of performances for different multi-step methods in the case of the Bratu-problem

Iterative methods	(43)	(23)
Size of problem	200	200
Number of steps (m)	8	5
Theoretical convergence-order	16	15
Computational convergence-order	16.67	15.51
Number of iterations	3	3
Max absolute error	$1.078e-198$	$1.14e-198$
Execution time	97.13	71.80

TABLE 4.3: Comparison of performances for different multi-step methods in the case of the Frank-Kamenetzkii-problem ($\alpha = 1,\ \kappa = 1$)

Iterative methods		(43)	(23)
Size of problem		150	150
Number of steps (m)		3	2
Number of iterations		4	4
Max absolute error at iter	1	1.17994	1.49025
	2	1.0978	1.23935
	3	0.743812	0.0000128682
	4	$1.38e-17$	$2.5e-40$
Execution time		15.23	15.02

TABLE 4.4: Comparison of performances for different multi-step methods in the case of the Lene-Emden problem ($p = 5$, Domain=$[0,\ 9]$)

.

In Table 4.2, we compared the numerical performances of different multi-step iterative schemes in the case of the Bratu-problem, for different values of α. The multi-step iterative scheme (4.42) is computational expensive with respect to (4.43) and the proposed multi-step iterative scheme (4.23). The simulation execution time for (4.43) and (4.23) are almost the same for the test cases of Bratu-problem. The maximum absolute error produced in the solution curve for Bratu-problem of the proposed multi-step iterative scheme (4.23) is comparatively better than the other two, in all the considered cases, by varying also the size of the grids.

For the Frank-Kamenetzkii-problem, we did not consider the iterative scheme (4.42) due to its extremely high computational cost. In Table 4.3, we took the grid size 200: we performed three iterations and used different number of steps to produce higher convergence-order. The maximum absolute error in the solution curve is approximately the same, but the simulation execution time of our proposed iterative scheme (4.23) is less than that of the iterative scheme (4.43). The successive iteration of convergence behavior for the Lene-Emden problem are shown in Figure 4.4 and 4.5. Table 4.4 shows the superiority our iterative scheme in the case of Lene-Emden problem. A large domain for the Lene-Emden initial-value problem is selected to analyze the convergence behavior of iterative schemes (4.43) and (4.23).

4.6 Summary

Usually higher-order Frechet derivatives are avoided in the construction of iterative schemes for a general class of systems of nonlinear equations, owing to the resulting

high computational cost. In this study, we have shown that there are interesting classes of systems of nonlinear equations associated with ODEs where higher-order Frechet derivatives are just diagonal matrices (of course this is not the case in general). The computational cost of diagonal matrices is the same as that related to the Jacobian. The use of second-order Frechet derivative enhances substantially the convergence-order and the resulting multi-step iterative scheme (4.23) achieved better performance index. The numerical simulations for the selected boundary value problems have shown the validity and accuracy of our proposed iterative scheme, in comparison with general purpose multi-step iterative schemes. However, we have to stress that our iterative technique is only efficient when the systems of nonlinear equations associated with ODEs has the special structure considered in this chapter.

Chapter 5

A Higher Order Multi-step Iterative Method for Computing the Numerical Solution of Systems of Nonlinear Equations Associated with Nonlinear PDEs and ODEs

The main research focus in this chapter is to address the construction of an efficient higher order multi-step iterative method to solve systems of nonlinear equations associated with nonlinear partial differential equations (PDEs) and ordinary differential equations (ODEs). The construction includes second order Frechet derivatives. The proposed multi-step iterative method uses two Jacobian evaluations at different points and requires only one inversion (in the sense of LU-factorization) of the Jacobian. The enhancement of convergence-order (CO) is hidden in the formation of a proper matrix polynomial. Since the cost of matrix vector multiplication is expensive computationally, we developed a matrix polynomial of degree two for the base method and of degree one to perform the subsequent steps, so we need just one matrix vector multiplication to perform each further step. The base method has convergence order four and each additional step enhances the CO by three. The general formula for CO is $3s-2$ for $s \geq 2$ and 2 for $s = 1$ where s is the step number. The number of function evaluations including Jacobian are $s+2$ and the number of matrix vectors multiplications are s. Regarding the s-step iterative method, we solve s upper and lower triangular systems, when the right hand side is a vector, and a single pair of triangular systems, when the right hand

side is a matrix. When considering systems of nonlinear equations stemming from the approximation of specific PDEs and ODEs, it is shown that the computational cost is almost the same if we compare the Jacobian and the second order Frechet derivative. The accuracy and validity of proposed multi-step iterative method is numerically checked with different examples of PDEs and ODEs.

5.1 Introduction

We are interested in higher order multi-step solvers for systems of nonlinear equations. Since high order Frechet derivatives can be naturally involved in these methods, we face a critical computational problem (see [154] for a valuable discussion concerning Frechet derivatives): here for high order Frechet derivative we mean a Frechet derivative of order larger or equal to three. However in the following we will show how to avoid the use of high order Frechet derivatives in the construction of iterative methods for general systems of nonlinear equations: in particular, for specific classes of systems of nonlinear equations associated with ODEs and PDEs, we will show that the cost of the second order Frechet derivative is still acceptable, from a computational viewpoint. To make things simpler, consider a system of three nonlinear equations

$$\boldsymbol{F}(\boldsymbol{y}) = [f_1(\boldsymbol{y}), f_2(\boldsymbol{y}), f_3(\boldsymbol{y})]^T = 0, \tag{5.1}$$

where $\boldsymbol{y} = [y_1,\ y_2,\ y_3]^T$. The first order Frechet derivative (Jacobian) of (5.1) is

$$\boldsymbol{F}'(\boldsymbol{y}) = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \frac{\partial f_1}{\partial y_3} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \frac{\partial f_2}{\partial y_3} \\ \frac{\partial f_3}{\partial y_1} & \frac{\partial f_3}{\partial y_2} & \frac{\partial f_3}{\partial y_3} \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \tag{5.2}$$

Next we proceed for the calculation of second-order Frechet derivative. Suppose $\boldsymbol{h} = [h_1,\ h_2,\ h_3]^T$ is a constant vector.

$$\boldsymbol{F}'(\boldsymbol{y})\boldsymbol{h} = \begin{bmatrix} h_1 f_{11} + h_2 f_{12} + h_3 f_{13} \\ h_1 f_{21} + h_2 f_{22} + h_3 f_{23} \\ h_1 f_{31} + h_2 f_{32} + h_3 f_{33} \end{bmatrix}, \boldsymbol{F}''(\boldsymbol{y})\boldsymbol{h}^2 = \frac{\partial(F'(\boldsymbol{y})\boldsymbol{h})}{\partial(y_1, y_2, y_3)}\boldsymbol{h},$$

$$\begin{aligned}
\boldsymbol{F}''(\boldsymbol{y})\boldsymbol{h}^2 &= \begin{bmatrix} h_1 f_{111} + h_2 f_{121} + h_3 f_{131} & h_1 f_{112} + h_2 f_{122} + h_3 f_{132} & h_1 f_{113} + h_2 f_{123} + h_3 f_{133} \\ h_1 f_{211} + h_2 f_{221} + h_3 f_{231} & h_1 f_{212} + h_2 f_{222} + h_3 f_{232} & h_1 f_{213} + h_2 f_{223} + h_3 f_{233} \\ h_1 f_{311} + h_2 f_{321} + h_3 f_{331} & h_1 f_{312} + h_2 f_{322} + h_3 f_{332} & h_1 f_{313} + h_2 f_{323} + h_3 f_{333} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, \\
&= \begin{bmatrix} h_1^2 f_{111} + h_1 h_2 f_{121} + h_1 h_3 f_{131} + h_1 h_2 f_{112} + h_2^2 f_{122} + h_2 h_3 f_{132} + h_1 h_3 f_{113} + h_2 h_3 f_{123} + h_3^2 f_{133} \\ h_1^2 f_{211} + h_1 h_2 f_{221} + h_1 h_3 f_{231} + h_1 h_2 f_{212} + h_2^2 f_{222} + h_2 h_3 f_{232} + h_1 h_3 f_{213} + h_2 h_3 f_{223} + h_3^2 f_{233} \\ h_1^2 f_{311} + h_1 h_2 f_{321} + h_1 h_3 f_{331} + h_1 h_2 f_{312} + h_2^2 f_{322} + h_2 h_3 f_{332} + h_1 h_3 f_{313} + h_2 h_3 f_{323} + h_3^2 f_{333} \end{bmatrix}, \\
&= \begin{bmatrix} h_1^2 f_{111} + h_2^2 f_{122} + h_3^2 f_{133} + h_1 h_2 (f_{121} + f_{112}) + h_1 h_3 (f_{131} + f_{113}) + h_2 h_3 (f_{132} + f_{123}) \\ h_1^2 f_{211} + h_2^2 f_{222} + h_3^2 f_{233} + h_1 h_2 (f_{221} + f_{212}) + h_1 h_3 (f_{231} + f_{213}) + h_2 h_3 (f_{232} + f_{223}) \\ h_1^2 f_{311} + h_2^2 f_{322} + h_3^2 f_{333} + h_1 h_2 (f_{321} + f_{312}) + h_1 h_3 (f_{331} + f_{313}) + h_2 h_3 (f_{332} + f_{323}) \end{bmatrix}, \\
&= \begin{bmatrix} h_1^2 f_{111} + h_2^2 f_{122} + h_3^2 f_{133} + 2h_1 h_2 f_{121} + 2h_1 h_3 f_{113} + 2h_2 h_3 f_{123} \\ h_1^2 f_{211} + h_2^2 f_{222} + h_3^2 f_{233} + 2h_1 h_2 f_{212} + 2h_1 h_3 f_{213} + 2h_2 h_3 (f_{232} + f_{223}) \\ h_1^2 f_{311} + h_2^2 f_{322} + h_3^2 f_{333} + 2h_1 h_2 f_{312} + 2h_1 h_3 f_{313} + 2h_2 h_3 (f_{332} + f_{323}) \end{bmatrix}.
\end{aligned} \tag{5.3}$$

Finally, we get the expression for $\boldsymbol{F}''(\boldsymbol{y})\boldsymbol{h}^2$

$$\boldsymbol{F}''(\boldsymbol{y})\boldsymbol{h}^2 = \begin{bmatrix} f_{111} & f_{122} & f_{133} \\ f_{211} & f_{222} & f_{233} \\ f_{311} & f_{322} & f_{333} \end{bmatrix} \begin{bmatrix} h_1^2 \\ h_2^2 \\ h_3^2 \end{bmatrix} + 2 \begin{bmatrix} f_{121} & f_{113} & f_{123} \\ f_{212} & f_{213} & f_{223} \\ f_{312} & f_{313} & f_{323} \end{bmatrix} \begin{bmatrix} h_1 h_2 \\ h_1 h_3 \\ h_2 h_3 \end{bmatrix}. \tag{5.4}$$

Clearly, the computational cost for second-order Frechet derivative is high in the case of general systems of nonlinear equations. Many systems of nonlinear equations associated with PDEs and ODEs can be written as

$$\begin{cases} F(y) = L(y) + f(y) + w = 0, \\ \boldsymbol{F}(\boldsymbol{y}) = A\boldsymbol{y} + f(\boldsymbol{y}) + \boldsymbol{w} = 0, \end{cases} \tag{5.5}$$

where A is the discrete approximation to linear differential operator $L(\cdot)$ and $f(\cdot)$ is the nonlinear function. If we write down the second-order Frechet derivative of (5.5) by

using (5.4) we get

$$\boldsymbol{F}''(\boldsymbol{y})\boldsymbol{h}^2 = \begin{bmatrix} f''(y_1) & 0 & 0 & \cdots & 0 \\ 0 & f''(y_2) & 0 & \cdots & 0 \\ 0 & 0 & f''(y_3) & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & f''(y_n) \end{bmatrix} \begin{bmatrix} h_1^2 \\ h_2^2 \\ h_3^2 \\ \vdots \\ h_n^2 \end{bmatrix} \tag{5.6}$$

and hence the related second-order Frechet derivative is easy to handle from a computational viewpoint. Now, for the further analysis , we introduce suitable notation. Let $\boldsymbol{a} = [a_1,\ a_2,\ ,\cdots,a_n]^T$ and $\boldsymbol{b} = [b_1,\ b_2,\ ,\cdots,b_n]^T$ be two vectors, the we define the diagonal of a vector and the point-wise product between two vectors as

$$diag(\boldsymbol{a}) = \begin{bmatrix} a_1 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & 0 & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & a_n \end{bmatrix}, \boldsymbol{a} \odot \boldsymbol{b} = diag(\boldsymbol{a})\ \boldsymbol{b} = [a_1 b_1,\ a_2 b_2, \cdots, a_n b_n]^T. \tag{5.7}$$

For the motivation of readers we list some famous nonlinear ODEs and PDEs and their first- and second-order derivatives in scalar and vectorial forms (Frechet derivatives). Let D_x and D_t be the discrete approximations of differential operators in spatial and temporal dimensions and let u be the function of spatial variables (in some cases temporal variable is also considered). We also introduce a function h which is independent from u, while the symbols I_t and I_x denote the identity matrices size equal to the number of nodes in temporal and spatial dimensions, respectively.

5.1.1 Bratu problem

The Bratu problem is discussed in [183] and it is stated as

$$\begin{cases} f(u) = u'' + \lambda e^u = 0, \qquad u(0) = u(1) = 0, \\ \frac{df(u)}{du} h = h'' + \lambda e^u h, \\ \frac{d^2 f(u)}{du^2} h^2 = \lambda e^u h^2, \\ \boldsymbol{F}(\boldsymbol{u}) = D_x^2 \boldsymbol{u} + \lambda e^{\boldsymbol{u}} = \boldsymbol{0}, \\ \boldsymbol{F}'\boldsymbol{h} = D_x^2 \boldsymbol{h} + \lambda e^{\boldsymbol{u}} \odot \boldsymbol{h}, \\ \boldsymbol{F}' = D_x^2 + \lambda \ diag\big(e^{\boldsymbol{u}}\big), \\ \boldsymbol{F}''\boldsymbol{h}^2 = \lambda e^{\boldsymbol{u}} \odot \boldsymbol{h}^2. \end{cases} \tag{5.8}$$

The closed form solution of Bratu problem can be written as

$$\begin{cases} u(x) = -2log\left(\frac{cosh((x-0.5)(0.5\theta))}{cosh(0.25\theta)}\right), \\ \theta = \sqrt{2\lambda} cosh\big(0.25\theta\big). \end{cases} \tag{5.9}$$

The critical value of λ satisfies $4 = \sqrt{4\lambda_c} sinh(0.25\theta_c)$. The Bratu problem has two solutions, unique solution and no solution if $\lambda < \lambda_c$, $\lambda = \lambda_c$ and $\lambda > \lambda_c$ respectively. The critical value $\lambda_c = 3.51383071912516$.

5.1.2 Frank-Kamenetzkii problem

The Frank-Kamenetzkii problem [182] is written as

$$\begin{cases} u'' + \frac{1}{x} u' + \lambda e^u = 0, \qquad u'(0) = u(1) = 0, \\ \boldsymbol{F}(\boldsymbol{u}) = D_x^2 \boldsymbol{u} + \frac{1}{\boldsymbol{x}} \odot D_x \boldsymbol{u} + \lambda e^{\boldsymbol{u}} = \boldsymbol{0}, \\ \boldsymbol{F}'\boldsymbol{h} = D_x^2 \boldsymbol{h} + \frac{1}{\boldsymbol{x}} \odot D_x \boldsymbol{h} + \lambda e^{\boldsymbol{u}} \odot \boldsymbol{h}, \\ \boldsymbol{F}' = D_x^2 + diag\left(\frac{1}{\boldsymbol{x}}\right) D_x + \lambda \ diag\big(e^{\boldsymbol{u}}\big), \\ \boldsymbol{F}''\boldsymbol{h}^2 = \lambda e^{\boldsymbol{u}} \odot \boldsymbol{h}^2. \end{cases} \tag{5.10}$$

The Frank-Kamenetzkii problem has no solution ($\lambda > 2$), ($\lambda = 2$) and two solutions ($\lambda < 2$). The closed form solution of (5.10) is given as

$$\begin{cases} c_1 = log\left(2(4-\lambda) \pm 4\sqrt{2(2-\lambda)}\right), \\ c_2 = log\left(\frac{4-\lambda) \pm 2\sqrt{2(2-\lambda)}}{2\lambda^2}\right), \\ u(x) = log\left(\frac{16e^{c_1}}{(2\lambda + e^{c_1}x^2)^2}\right), \\ u(x) = log\left(\frac{16e^{c_1}}{(1+2\lambda e^{c_2}x^2)^2}\right). \end{cases} \tag{5.11}$$

5.1.3 Lane-Emden equation

The Lane-Emden equation is a classical equation [184] which has been introduced in 1870 by Lane and further investigated by Emden in (1907). Lane-Emden equation deals with mass density distribution inside a spherical star when it is in hydrostatic equilibrium. The lane-Emden equation for index $n = 5$ can be written as

$$\begin{cases} u'' + \frac{2}{x}u' + u^5 = 0, & u(0) = 1,\ u'(0) = 0, \\ \boldsymbol{F}(\boldsymbol{u}) = D_x^2\boldsymbol{u} + \frac{1}{\boldsymbol{x}} \odot D_x\boldsymbol{u} + \boldsymbol{u}^5, \\ \boldsymbol{F'h} = D_x^2\boldsymbol{h} + \frac{1}{\boldsymbol{x}} \odot D_x\boldsymbol{h} + 5\boldsymbol{u}^4 \odot \boldsymbol{h}, \\ \boldsymbol{F'} = D_x^2 + diag\left(\frac{1}{\boldsymbol{x}}\right)D_x + 5\ diag\left(\boldsymbol{u}^4\right), \\ \boldsymbol{F''h}^2 = 20\ \boldsymbol{u}^3 \odot \boldsymbol{h}^2. \end{cases} \tag{5.12}$$

The closed form solution of (5.12) can be written as

$$u(x) = \left(1 + \frac{x^2}{3}\right)^{-\frac{1}{2}}. \tag{5.13}$$

5.1.4 Klein-Gordan equation

Klein-Gordan equation is discussed and solved in [185].

$$\begin{cases} u_{tt} - c^2 u_{xx} + f(u) = p, & -\infty < x < \infty,\ t > 0 \\ \boldsymbol{F}(\boldsymbol{u}) = (D_t^2 - c^2 D_x^2)\boldsymbol{u} + f(\boldsymbol{u}) - \boldsymbol{p}, \\ \boldsymbol{F}'\boldsymbol{h} = (D_t^2 - c^2 D_x^2)\boldsymbol{h} + f'(\boldsymbol{u}) \odot \boldsymbol{h}, \\ \boldsymbol{F}' = D_t^2 - c^2 D_x^2 + diag(f'(\boldsymbol{u})), \\ \boldsymbol{F}''\boldsymbol{h}^2 = f''(\boldsymbol{u}) \odot \boldsymbol{h}^2, \end{cases} \tag{5.14}$$

where $f(u)$ is the odd function of u and initial conditions are

$$\begin{cases} u(x,0) = g_1(x), \\ u_t(x,0) = g_2(x). \end{cases} \tag{5.15}$$

We have calculated the second-order Frechet derivatives of four different nonlinear ODEs and PDEs. Clearly the computational costs of second-order Frechet derivatives are not higher than first-order Frechet derivatives or Jacobians. So we insist that the second-order Frechet derivatives for particular class of ODEs and PDEs are not expensive as they are in the case of general systems of nonlinear equations. The main source of information about iterative methods is the manuscript written by J. F. Traub [3] in 1964. Recently many researchers have contributed in the area of iterative method for systems of nonlinear equations [104, 109, 142, 186, 187, 187–191]. The major part of work is devoted to the construction of iterative methods for single variable nonlinear equations[103]. According to Traub's conjecture if we use n function evaluations, then the maximum CO is 2^{n-1} in the case of single variable nonlinear equations but for multi-variable case we do not have such claim. In the case of systems of nonlinear equations the multi-steps iterative methods are interesting because with minimum computational cost we are allowed to construct higher-order convergence iterative methods. For the better understanding we can divide multi-steps iterative methods in two parts : one is called base method, the other is called multi-steps. In the base method we construct an iterative method in way that it provides maximum enhancement in the convergence-order with minimum computational cost when we perform multi-steps. Malik et. al.

[192] proposed the following multi-step iterative method (MZ_1) :

$$MZ_1 = \begin{cases} \text{Number of steps} & = m \geq 2 \\ \text{CO} & = 2m \\ \text{Function evaluations} & = m+1 \\ \text{Inverses} & = 2 \\ \text{Matrix vector multiplications} & = 1 \\ \text{Number of solutions of systems of linear equations} & \\ \text{when right hand side is matrix} & = 1 \\ \text{when right hand side is vector} & = m-1 \end{cases} \begin{cases} \text{Base-Method} \rightarrow \begin{cases} \boldsymbol{F}'(\boldsymbol{x})\boldsymbol{\phi}_1 = \boldsymbol{F}(\boldsymbol{x}) \\ \boldsymbol{y}_1 = \boldsymbol{x} - \frac{2}{3}\boldsymbol{\phi}_1 \\ W = \frac{1}{2}\left(3\boldsymbol{F}'(\boldsymbol{y}_1) - \boldsymbol{F}'(\boldsymbol{x})\right) \\ WT = 3\boldsymbol{F}'(\boldsymbol{y}_1) + \boldsymbol{F}'(\boldsymbol{x}) \\ \boldsymbol{y}_2 = \boldsymbol{x} - \frac{1}{4}T\boldsymbol{\phi}_1 \end{cases} \\ (m-2)\text{-steps} \rightarrow \begin{cases} \text{For } s = 1, m-2 \\ \quad W\boldsymbol{\phi}_{s+1} = \boldsymbol{F}(\boldsymbol{y}_{s+1}), \\ \quad \boldsymbol{y}_{s+2} = \boldsymbol{y}_{s+1} - \boldsymbol{\phi}_{s+1}, \\ \text{End} \\ \boldsymbol{x} = \boldsymbol{y}_m. \end{cases} \end{cases}$$

In [37] F. Soleymani and co-researchers constructed an other multi-step iterative method (FS):

$$FS = \begin{cases} \text{Number of steps} & = m \geq 2 \\ \text{CO} & = 2m \\ \text{Function evaluations} & = m+1 \\ \text{Inverses} & = 2 \\ \text{Matrix vector multiplications} & = 2m-3 \\ \text{Number of solutions of systems of linear equations} & \\ \text{when right hand side is matrix} & = 1 \\ \text{when right hand side is vector} & = m-1 \end{cases} \begin{cases} \text{Base-Method} \rightarrow \begin{cases} \boldsymbol{F}'(\boldsymbol{x})\boldsymbol{\phi}_1 = \boldsymbol{F}(\boldsymbol{x}) \\ \boldsymbol{y}_1 = \boldsymbol{x} - \frac{2}{3}\boldsymbol{\phi}_1 \\ W = \frac{1}{2}\left(3\boldsymbol{F}'(\boldsymbol{y}_1) - \boldsymbol{F}'(\boldsymbol{x})\right) \\ WT = 3\boldsymbol{F}'(\boldsymbol{y}_1) + \boldsymbol{F}'(\boldsymbol{x}) \\ \boldsymbol{y}_2 = \boldsymbol{x} - T\boldsymbol{\phi}_1 \end{cases} \\ (m-2)\text{-steps} \rightarrow \begin{cases} \text{For } s = 1, m-2 \\ \quad \boldsymbol{F}'(\boldsymbol{x})\boldsymbol{\phi}_{s+1} = \boldsymbol{F}(\boldsymbol{y}_{s+1}), \\ \quad \boldsymbol{y}_{s+2} = \boldsymbol{y}_{s+1} - T^2\boldsymbol{\phi}_{s+1}, \\ \text{End} \\ \boldsymbol{x} = \boldsymbol{y}_m. \end{cases} \end{cases}$$

H. Montazeri et. al. [105] developed the more efficient multi-step iterative methods (*HM*):

$$HM = \begin{cases} \text{Number of steps} & = m \geq 2 \\ \text{CO} & = 2m \\ \text{Function evaluations} & = m+1 \\ \text{Inverses} & = 1 \\ \text{Matrix vector multiplications} & = m \\ \text{Number of solutions of systems of linear equations when right hand side is matrix} & = 1 \\ \text{when right hand side is vector} & = m-1 \end{cases} \begin{cases} \text{Base-Method} \rightarrow \begin{cases} \boldsymbol{F}'(\boldsymbol{x})\boldsymbol{\phi}_1 = \boldsymbol{F}(\boldsymbol{x}) \\ \boldsymbol{y}_1 = \boldsymbol{x} - \frac{2}{3}\boldsymbol{\phi}_1 \\ \boldsymbol{F}'(\boldsymbol{x})T = \boldsymbol{F}'(\boldsymbol{y}_1) \\ \boldsymbol{y}_2 = \boldsymbol{x} - \left(\frac{23}{8}I - 3T + \frac{9}{8}T^2\right)\boldsymbol{\phi}_1 \end{cases} \\ (m-2)\text{-steps} \rightarrow \begin{cases} \text{For } s = 1, m-2 \\ \quad \boldsymbol{F}'(\boldsymbol{x})\boldsymbol{\phi}_{s+1} = \boldsymbol{F}(\boldsymbol{y}_{s+1}), \\ \quad \boldsymbol{y}_{s+2} = \boldsymbol{y}_{s+1} - \\ \left(\frac{5}{2}I - \frac{3}{2}T\right)\boldsymbol{\phi}_{s+1}, \\ \text{End} \\ \boldsymbol{x} = \boldsymbol{y}_m. \end{cases} \end{cases}$$

5.2 The proposed new multi-step iterative method

We now propose a new multi-step iterative method (MZ_2):

$$MZ_2 = \begin{cases} \text{Number of steps} & = m \geq 2 \\ \text{CO} & = 3m-2 \\ \text{Function evaluations} & = m+2 \\ \text{Inverses} & = 1 \\ \text{Matrix vector multiplications} & = m \\ \text{Number of solutions of systems of linear equations when right hand side is matrix} & = 1 \\ \text{right hand side is vector} & = m \end{cases} \begin{cases} \text{Base-Method} \rightarrow \begin{cases} \boldsymbol{F}'(\boldsymbol{x})\boldsymbol{\phi}_1 = \boldsymbol{F}(\boldsymbol{x}) \\ \boldsymbol{F}'(\boldsymbol{x})\boldsymbol{\phi}_2 = \boldsymbol{F}''\left(\boldsymbol{x} - \frac{4}{9}\boldsymbol{\phi}_1\right)\boldsymbol{\phi}_1^2 \\ \boldsymbol{y}_1 = \boldsymbol{x} - \left(\boldsymbol{\phi}_1 + \frac{3}{2}\boldsymbol{\phi}_2\right) \\ \boldsymbol{F}'(\boldsymbol{x})T = \boldsymbol{F}'(\boldsymbol{y}_1) \\ \boldsymbol{y}_2 = \boldsymbol{x} - \left(\frac{7}{2}I - 6T + \frac{7}{2}T^2\right) \\ \left(\boldsymbol{\phi}_1 + \frac{3}{2}\boldsymbol{\phi}_2\right) \end{cases} \\ (m-2)\text{-steps} \rightarrow \begin{cases} \text{For } s = 1, m-2 \\ \quad \boldsymbol{F}'(\boldsymbol{x})\boldsymbol{\phi}_{s+2} = \boldsymbol{F}(\boldsymbol{y}_{s+1}), \\ \quad \boldsymbol{y}_{s+2} = \boldsymbol{y}_{s+1} - \left(2I - T\right)\boldsymbol{\phi}_{s+2}, \\ \text{End} \\ \boldsymbol{x} = \boldsymbol{y}_m. \end{cases} \end{cases}$$

We claim that the convergence-order of our proposed multi-step iterative method is

$$\mathrm{CO} = \begin{cases} 2 & m = 1, \\ 3m-2 & m \geq 2, \end{cases} \tag{5.16}$$

where m is the number of steps of MZ_2. The computational costs of MZ_1 and FS are high because both methods use two inversions of matrices and hence we will not consider MZ_1 and FS methods in our subsequent analysis and discussion. The multi-step iterative method HM use only one inversion of Jacobian and hence is a good candidate for the performance comparison. We presented comparison between MZ_2 and HM in Table 5.1 and 5.2. If the number of function evaluations and the number of solutions of the system of linear equations are equal, then the performance of MZ_2 in terms of convergence-order is better than HM, when the number of steps of MZ_2 is grater or equal to four (see Table 5.1). When the convergence-orders of both iterative methods are equal then, from Table 5.2, we can see that the computational effort of HM is always higher than that of MZ_2 for $m \geq 2$.

The performance index to measure the efficiency of an iterative method to solve systems of nonlinear equation is defined as

$$\rho = CO^{\frac{1}{flops}}. \tag{5.17}$$

In Table 5.3 we provided the (multiplicative) computational cost of different matrix and matrix-vector operations and Table 5.4 shows the performance index as defined in (5.20) for a particular case, when HM and MZ_2 have the same convergence-order. Clearly the performance index of MZ_2 is better than that of HM.

	MZ_2 $(m \geq 2)$	HM $(m \geq 2)$	MZ_2 $(m = 2)$	HM $(m = 3)$	MZ_2 $(m = m_1)$	HM $(m = m_1 + 1)$	Difference $MZ_2 - HM$
Number of steps	m	m	2	3	m_1	m_1+1	1
Convergence-order	$3m-2$	$2m$	4	6	$3m_1-2$	$2(m_1+1)$	m_1-4
Function evaluations	$m+2$	$m+1$	4	4	m_1+2	m_1+2	0
Solution of system of linear equations when right hand side is vector	m	$m-1$	2	2	m_1	m_1	0
Solution of system of linear equations when right hand side is matrix	1	1	1	1	1	1	0
Matrix vector multiplications	m	m	2	3	m_1	m_1+1	-1

TABLE 5.1: Comparison between multi-steps iterative method MZ_2 and HM if number of function evaluations and solutions of system of linear equations are equal.

	MZ_2 $(m \geq 1)$	HM $(m \geq 1)$	Difference $HM - MZ_2$
Number of steps	$2m$	$3m-1$	$m-1$
Convergence-order	$6m-2$	$6m-2$	0
Function evaluations	$2m+2$	$3m$	$m-2$
Solution of system of linear equations when right hand side is vector	$2m$	$3m$	m
Solution of system of linear equations when right hand side is matrix	1	1	0
Matrix vector multiplications	$2m$	$3m-1$	$m-1$

TABLE 5.2: Comparison between multi-steps iterative method MZ_2 and HM if convergence orders are equal.

LU decomposition		
Multiplications $\frac{n(n-1)(2n-1)}{6}$	Divisions $\frac{n(n-1)}{2}$	Total cost $\frac{n(n-1)(2n-1)}{6}+3\frac{n(n-1)}{2}$
Two triangular systems (if right hand side is a vector)		
Multiplications $n(n-1)$	Divisions n	Total cost $n(n-1)+3n$
Two triangular systems (if right hand side is a matrix)		
Multiplications $n^2(n-1)$	Divisions n^2	Total cost $n^2(n-1)+3n^2$
Matrix vector multiplication		
n^2		

TABLE 5.3: Computational cost of different operations (the computational cost of a division is three times to multiplication).

Iterative methods	HM	MZ_2
Number of steps	5	4
Rate of convergence	10	10
Number of functional evaluations	$6n$	$6n$
The classical efficiency index	$2^{1/(6n)}$	$2^{1/(6n)}$
Number of Lu factorizations	1	1
Cost of Lu factorizations	$\frac{n(n-1)(2n-1)}{6}+3\frac{n(n-1)}{2}$	$\frac{n(n-1)(2n-1)}{6}+3\frac{n(n-1)}{2}$
Cost of linear systems	$4(n(n-1)+3n)+n^2(n-1)+3n^2$	$4(n(n-1)+3n)+n^2(n-1)+3n^2$
Matrix vector multiplications	$5n^2$	$4n^2$
Flops-like efficiency index	$10^{1/\left(\frac{4n^3}{3}+12n^2+\frac{38}{3}n\right)}$	$10^{1/\left(\frac{4n^3}{3}+11n^2+\frac{38}{3}n\right)}$

TABLE 5.4: Comparison of performance index between multi-steps iterative methods MZ_2 and HM.

5.3 Convergence analysis

In this section, we will prove that the local convergence-order of MZ_2 is seven for $m = 3$. Later we will establish a proof for the convergence-order of the multi-step iterative scheme MZ_2, by using mathematical induction.

Theorem 5.1. *Let $F : \Gamma \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be sufficiently Frechet differentiable on an open convex neighborhood Γ of $\boldsymbol{x}^* \in \mathbb{R}^n$ with $F(\boldsymbol{x}^*) = 0$ and $det(F'(\boldsymbol{x}^*)) \neq 0$. Then the sequence $\{\boldsymbol{x}_k\}$ generated by the iterative scheme MZ_2 converges to $\boldsymbol{x}^*$ with local order of convergence seven, and produces the following error equation*

$$\boldsymbol{e}_{k+1} = L\boldsymbol{e}_k{}^7 + O(\boldsymbol{e}_k{}^8), \tag{5.18}$$

where $\boldsymbol{e}_k = \boldsymbol{x_k} - \boldsymbol{x}^*$, $\boldsymbol{e}_k{}^p = \overbrace{(\boldsymbol{e}_k, \boldsymbol{e}_k, \cdots, \boldsymbol{e}_k)}^{p\text{-times}}$ and $L = -2060C_2^6 - 618C_3C_2^4 + 260C_2^3C_4(1/9) + 26C_3C_2C_4(1/3) - 30C_3C_2C_3C_2 - 6C_3C_2^2C_3 - 100C_2^3C_3C_2 - 20C_2^4C_3$ is a p-linear function i.e. $L \in \mathbb{L}\overbrace{(\mathbb{R}^n, \mathbb{R}^n, \cdots, \mathbb{R}^n)}^{p\text{-times}}$ and $L\boldsymbol{e}_k{}^p \in \mathbb{R}^n$.

Proof. Let $F : \Gamma \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be sufficiently Frechet differentiable function in Γ. The qth Frechet derivative of F at $v \in \mathbb{R}^n$, $q \geq 1$, is the $q - linear$ function $F^{(q)}(v) : \overbrace{\mathbb{R}^n\mathbb{R}^n \cdots \mathbb{R}^n}^{q\text{-times}}$ such that $F^{(q)}(v)(u_1, u_2, \cdots, u_q) \in \mathbb{R}^n$. The Taylor's series expansion of $F(\boldsymbol{x}_k)$ around $\boldsymbol{x}^*$ can be written as:

$$\begin{aligned}
F(\boldsymbol{x}_k) &= F(\boldsymbol{x}^* + \boldsymbol{x}_k - \boldsymbol{x}^*) = F(\boldsymbol{x}^* + \boldsymbol{e}_k), && (5.19)\\
&= F(\boldsymbol{x}^*) + F'(\boldsymbol{x}^*)\boldsymbol{e}_k + \frac{1}{2!}F''(\boldsymbol{x}^*)\boldsymbol{e}_k{}^2 + \frac{1}{3!}F^{(3)}(\boldsymbol{x}^*)\boldsymbol{e}_k{}^3 + \cdots, && (5.20)\\
&= F'(\boldsymbol{x}^*)\Big[\boldsymbol{e}_k + \frac{1}{2!}F'(\boldsymbol{x}^*)^{-1}F''(\boldsymbol{x}^*)\boldsymbol{e}_k{}^2 + \frac{1}{3!}F'(\boldsymbol{x}^*)^{-1}F^{(3)}(\boldsymbol{x}^*)\boldsymbol{e}_k{}^3 + \cdots\Big], && (5.21)\\
&= C_1\Big[\boldsymbol{e}_k + C_2\boldsymbol{e}_k{}^2 + C_3\boldsymbol{e}_k{}^3 + \cdots\Big], && (5.22)
\end{aligned}$$

where $C_1 = F'(\boldsymbol{x}^*)$ and $C_s = \frac{1}{s!}F'(\boldsymbol{x}^*)^{-1}F^{(s)}(\boldsymbol{x}^*)$ for $s \geq 2$. From (6.22), we can calculate the Frechet derivative of F:

$$F'(\boldsymbol{x}_k) = C_1\Big[I + 2C_2\boldsymbol{e}_k + 3C_3\boldsymbol{e}_k{}^2 + 4C_3\boldsymbol{e}_k{}^3 + \cdots\Big], \tag{5.23}$$

where I is the identity matrix. Furthermore, we calculate the inverse of the Jacobian matrix

$$\begin{aligned}F'(\boldsymbol{x}_k)^{-1} &= \big[I - 2C_2\boldsymbol{e}_k + (4C_2^2 - 3C_3)\boldsymbol{e}_k^2 + (6C_3C_2 + 6C_2C_3 - 8C_2^3 - 4C_4)\boldsymbol{e}_k^3 \\ &+ (8C_4C_2 + 9C_3^2 + 8C_2C_4 - 5C_5 - 12C_3C_2^2 - 12C_2C_3C_2 - 12C_2^2C_3 \\ &+ 16C_2^4)\boldsymbol{e}_k^4 + (24C_3C_2^3 + 24C_2^3C_3 + 24C_2^2C_3C_2 + 24C_2C_3C_2^2 + 10C_5C_2 \\ &+ 12C_4C_3 + 12C_3C_4 + 10C_2C_5 - 6C_6 - 16C_4C_2^2 - 18C_3^2C_2 - 18C_3C_2C_3 \\ &- 16C_2C_4C_2 - 18C_2(C_3^2) - 16C_2^2C_4 - 32C_2^5)\boldsymbol{e}_k^5 + (32C_4C_2^3 + 64C_2^6 \\ &- 48C_3(C_2^4) + 12C_2C_6 + 16C_4^2 + 15C_3C_5 + 15C_5C_3 + 12C_6C_2 - 24C_4C_2C_3 \\ &- 24C_4C_3C_2 - 20C_2^2C_5 - 24C_2C_3C_4 - 24C_2C_4C_3 + 32C_2^3C_4 - 20C_2C_5C_2 \\ &+ 36C_2^2(C_3^2) - 20C_5C_2^2 + 32C_2^2C_4C_2 + 32C_2C_4C_2^2 + 36C_2(C_3^2)C_2 \\ &+ 36C_2C_3C_2C_3 + 36C_3^2C_2^2 - 7C_7 - 24C_3C_2C_4 - 27C_3^3 - 24C_3C_4C_2 \\ &+ 36C_3C_2C_3C_2 + 36C_3C_2^2C_3 - 48C_2^2C_3C_2^2 - 48C_2^3C_3C_2 - 48C_2^4C_3 \\ &- 48C_2C_3C_2^3)\boldsymbol{e}_k^6 + \cdots\big]C_1^{-1} \end{aligned} \tag{5.24}$$

By multiplying $F'(\boldsymbol{x}_k)^{-1}$ and $F(\boldsymbol{x}_k)$, we obtain $\boldsymbol{\phi}_1$:

$$\begin{aligned}\boldsymbol{\phi}_1 &= \boldsymbol{e}_k - C_2\boldsymbol{e}_k^2 + (2C_2^2 - 2C_3)\boldsymbol{e}_k^3 + (-3C_4 - 4C_2^3 + 3C_3C_2 + 4C_2C_3)\boldsymbol{e}_k^4 \\ &+ (-4C_5 - 6C_3C_2^2 - 6C_2C_3C_2 - 8C_2^2C_3 + 8C_2^4 + 4C_4C_2 + 6C_3^2 + 6C_2C_4)\boldsymbol{e}_k^5 \\ &+ (-5C_6 + 12C_3C_2^3 + 16C_2^3C_3 + 12C_2^2C_3C_2 + 12C_2C_3C_2^2 - 8C_4C_2^2 - 9C_3^2C_2 \\ &- 12C_3C_2C_3 - 8C_2C_4C_2 - 12C_2(C_3^2) - 12C_2^2C_4 - 16C_2^5 + 5C_5C_2 + 8C_4C_3 \\ &+ 9C_3C_4 + 8C_2C_5)\boldsymbol{e}_k^6 + \cdots. \end{aligned} \tag{5.25}$$

The expression for $\boldsymbol{\phi}_2$ is the following:

$$\begin{aligned}\boldsymbol{\phi}_2 &= 2C_2\boldsymbol{e}_k^2 + (-8C_2^2 + 10C_3(1/3))\boldsymbol{e}_k^3 + (26C_2^3 - 38C_3C_2(1/3) - 12C_2C_3 \\ &+ 100C_4(1/27))\boldsymbol{e}_k^4 + (-364C_2C_4(1/27) - 18C_3^2 - 416C_4C_2(1/27) \\ &+ 116C_2^2C_3(1/3) + 36C_2C_3C_2 + 122C_3C_2^2(1/3) + 2500C_5(1/729) - 76C_2^4)\boldsymbol{e}_k^5 \\ &+ (-106C_2C_3C_2^2 - (1/3)(298C_2^2C_3C_2) - 344C_2^3C_3(1/3) + 1282C_2^2C_4(1/27) \\ &+ 140C_2(C_3^2)(1/3) + (1/27)(1106C_2C_4C_2) - 118C_3C_2^3 + 1364C_4C_2^2(1/27) \\ &- 10664C_2C_5(1/729) - 520C_3C_4(1/27) - 544C_4C_3(1/27) - 12290C_5C_2(1/729) \\ &+ 54C_3^2C_2 + (1/3)(184C_3C_2C_3) + 6250C_6(1/2187) + 208C_2^5)\boldsymbol{e}_k^6 + \cdots. \end{aligned} \tag{5.26}$$

The expressions for $\boldsymbol{y}_1$, T, $\boldsymbol{y}_2$ and $\boldsymbol{y}_3$ in order are

$$\begin{aligned}\boldsymbol{y}_1 - \boldsymbol{x}^* = {} & -2C_2\boldsymbol{e}_k^2 + (10C_2^2 - 3C_3)\boldsymbol{e}_k^3 + (-23C_4(1/9) - 35C_2^3 + 16C_3C_2 \\ & + 14C_2C_3)\boldsymbol{e}_k^4 + (-278C_5(1/243) - 55C_3C_2^2 - 48C_2C_3C_2 - 50C_2^2C_3 \\ & + 106C_2^4 + 172C_4C_2(1/9) + 21C_3^2 + 128C_2C_4(1/9))\boldsymbol{e}_k^5 + (147C_2C_3C_2^2 \\ & + 137C_2^2C_3C_2 + 156C_2^3C_3 - 533C_2^2C_4(1/9) - 58C_2(C_3^2) - (1/9)(481C_2C_4C_2) \\ & + 165C_3C_2^3 - 610C_4C_2^2(1/9) + 3388C_2C_5(1/243) + 179C_3C_4(1/9) \\ & + 200C_4C_3(1/9) + 4930C_5C_2(1/243) - 72C_3^2C_2 - 80C_3C_2C_3 \\ & + 520C_6(1/729) - 296C_2^5)\boldsymbol{e}_k^6 + \cdots . \end{aligned} \tag{5.27}$$

$$\begin{aligned} T = {} & I - 2C_2\boldsymbol{e}_k - 3C_3\boldsymbol{e}_k^2 + (6C_3C_2 - 4C_4 + 20C_2^3)\boldsymbol{e}_k^3 + (12C_3C_2^2 + 20C_2C_3C_2 \\ & + 28C_2^2C_3 - 110C_2^4 + 8C_4C_2 + 9C_3^2 + 26C_2C_4(1/9) - 5C_5)\boldsymbol{e}_k^4 + (-180C_3C_2^3 \\ & - 156C_2^3C_3 - 136C_2^2C_3C_2 - 134C_2C_3C_2^2 + 18C_3C_2C_3 + (1/9)(200C_2C_4C_2) \\ & + 24C_2(C_3^2) + 68C_2^2C_4(1/3) + 432C_2^5 + 10C_5C_2 + 12C_4C_3 + 12C_3C_4 \\ & + 1874C_2C_5(1/243) - 6C_6)\boldsymbol{e}_k^5 + (-112C_4C_2^3 - 1456C_2^6 + 1050C_3C_2^4 \\ & + 9788C_2C_6(1/729) + 16C_4^2 + 15C_3C_5 + 15C_5C_3 + 12C_6C_2 - 24C_4C_3C_2 \\ & + 3028C_2^2C_5(1/243) + 142C_2C_3C_4(1/9) + 184C_2C_4C_3(1/9) - 1474C_2^3C_4(1/9) \\ & + (1/243)(5000C_2C_5C_2) - 164C_2^2(C_3^2) - (1/3)(454C_2^2C_4C_2) - (1/9)(1220C_2C_4C_2^2) \\ & - 144C_2(C_3^2)C_2 - 196C_2C_3C_2C_3 - 222C_3^2C_2^2 - 7C_7 + 20C_3C_2C_4(1/3) \\ & - 26C_3C_4C_2(1/3) - 240C_3C_2C_3C_2 - 258C_3C_2^2C_3 + 562C_2^2C_3C_2^2 + 546C_2^3C_3C_2 \\ & + 624C_2^4C_3 + 690C_2C_3C_2^3)\boldsymbol{e}_k^6 + \cdots . \end{aligned} \tag{5.28}$$

$$\begin{aligned}
\boldsymbol{y}_2-\boldsymbol{x}^* &= (-5C_3C_2+13C_4(1/9)-103C_2^3-C_2C_3)\boldsymbol{e}_k^4+(-104C_2C_4(1/9)\\
&-21C_3^2(1/2)-80C_4C_2(1/9)-148C_2^2C_3-100C_2C_3C_2-109C_3C_2^2\\
&+937C_5(1/243)+666C_2^4)\boldsymbol{e}_k^5+(869C_2C_3C_2^2+873C_2^2C_3C_2+954C_2^3C_3\\
&-1133C_2^2C_4(1/9)-124C_2(C_3^2)-(1/9)(895C_2C_4C_2)+1074C_3C_2^3\\
&-1114C_4C_2^2(1/9)-715C_2C_5(1/27)-238C_3C_4(1/9)-178C_4C_3(1/9)\\
&-3575C_5C_2(1/243)-75C_3^2C_2-158C_3C_2C_3+4894C_6(1/729)\\
&-1990C_2^5)\boldsymbol{e}_k^6+(3632C_4C_2^3(1/3)+420C_2^6-4958C_3C_2^4-30616C_2C_6(1/729)\\
&-404C_4^2(1/9)-7343C_3C_5(1/162)-16001C_5C_3(1/486)-15620C_6C_2(1/729)\\
&-1580C_4C_2C_3(1/9)-580C_4C_3C_2(1/9)-18334C_2^2C_5(1/243)\\
&-761C_2C_3C_4(1/9)-847C_2C_4C_3(1/9)+1074C_2^3C_4-(1/243)(19556C_2C_5C_2)\\
&+1118C_2^2(C_3^2)-35410C_5C_2^2(1/243)+(1/9)(8924C_2^2C_4C_2)\\
&+(1/3)(3038C_2C_4C_2^2)+1040C_2(C_3^2)C_2+1262C_2C_3C_2C_3+1390C_3^2C_2^2\\
&+63418C_7(1/6561)-919C_3C_2C_4(1/9)-165C_3^3(1/2)-589C_3C_4C_2(1/9)\\
&+1331C_3C_2C_3C_2+1542C_3C_2^2C_3-2678C_2^2C_3C_2^2-2886C_2^3C_3C_2\\
&-2881C_2^4C_3-3871C_2C_3C_2^3)\boldsymbol{e}_k^7+\cdots. \qquad (5.29)
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{y}_3-\boldsymbol{x}^* &= (-2060C_2^6-618C_3C_2^4+260C_2^3C_4(1/9)+26C_3C_2C_4(1/3)\\
&-30C_3C_2C_3C_2-6C_3C_2^2C_3-100C_2^3C_3C_2-20C_2^4C_3)\boldsymbol{e}_k^7+\cdots. \qquad (5.30)
\end{aligned}$$

□

Theorem 5.2. *The multi-step iterative scheme MZ_2 has the local convergence-order $3m-2$, using $m(\geq 2)$ evaluations of a sufficiently differentiable function F, two first-order Frechet derivatives F' and one second-order Frechet derivate F'' per full-cycle.*

Proof. The proof is established from mathematical induction. For $m = 1,\ 2,\ 3$ the convergence-orders are two, four and seven from (5.27), (5.29), and (5.30), respectively. Consequently our claim concerning the convergence-order $3m-2$ is true for $m = 2,\ 3$.

We assume that our claim is true for $m = q > 3$, i.e., the convergence-order of MZ_2 is $3q-2$. The qth-step and $(q-1)th$-step of iterative scheme MZ_2 can be written as:

$$Frozen - factor = (2I - T)\boldsymbol{F}'(\boldsymbol{x})^{-1}, \tag{5.31}$$

$$\boldsymbol{y}_{q-1} = \boldsymbol{y}_{q-2} - (Frozen - factor)\boldsymbol{F}(\boldsymbol{y}_{q-2}), \tag{5.32}$$

$$\boldsymbol{y}_q = \boldsymbol{y}_{q-1} - (Frozen - factor)\boldsymbol{F}(\boldsymbol{y}_{q-1}). \tag{5.33}$$

The enhancement in the convergence-order of MZ_2 from $(q-1)th$-step to qth-step is $(3q-2)-(3(q-1)-2) = 3$. Now we write the $(q+1)th$-step of MZ_2:

$$\boldsymbol{y}_{q+1} = \boldsymbol{y}_q - (Frozen - factor)\boldsymbol{F}(\boldsymbol{y}_q). \tag{5.34}$$

The increment in the convergence-order of MZ_2, due to $(q+1)th$-step, is exactly three, because the use of the $Frozen - factor$ adds an additive constant in the convergence-order[19]. Finally the convergence-order after the addition of the $(q+1)th$-step is $3q-2+3 = 3q+1 = 3(q+1)-2$, which completes the proof. □

5.4 Numerical testing

For the verification of convergence-order, we use the following definition for the computational convergence-order (COC):

$$COC \approx \frac{log\left[Max(|\boldsymbol{x}_{q+2} - \boldsymbol{x}^*|)/Max(|\boldsymbol{x}_{q+1} - \boldsymbol{x}^*|)\right]}{log\left[Max(|\boldsymbol{x}_{q+1} - \boldsymbol{x}^*|)/Max(|\boldsymbol{x}_q - \boldsymbol{x}^*|)\right]}, \tag{5.35}$$

where $Max(|\boldsymbol{x}_{q+2} - \boldsymbol{x}^*|)$ is the maximum absolute error. The number of solutions of systems of linear equations are same in both iterative methods when right hand side is a matrix so we will not mention it in comparison tables. The main benefit of multi-step iterative methods is that we invert Jacobian once and then use it repeatedly in multi-steps part to get better convergence-order for a single cycle of iterative method. We have conducted numerical tests for four different problems to show the accuracy and validity of our proposed multi-step iterative method MZ_2. For the purpose of comparison we adopt two ways (i) when both iterative methods have same number of function evaluations and solution of systems of linear equations (ii) when both schemes have same convergence order. Tables 5.5, 5.7 and 5.8 show that when we number of function evaluations and solutions of systems of linear equation are equal and the convergence

order of MZ_2 is higher than ten then our proposed scheme show better accuracy in less execution time. On the other hand if convergence-order of MZ_2 is less than ten then the performance of HM is relatively better. For the second cases when we equate the convergence-orders the execution time of MZ_2 are always less than that of HM because HM performs more steps to achieve the same convergence-order. Tables 5.6, 5.9 and 5.10 shows that MZ_2 achieve better or almost equal accuracy with less execution time. We have also simulated one PDE Klein-Gordon and results are depicted in Table 5.11. As we have commented if the convergence-order is less ten the performance of HM is better and it is clearly evident in Table 5.11 but the accuracy of MZ_2 is comparable with HM. The numerical error in solution due to MZ_2 is shown in Figure 5.1 and Figure 5.2 corresponds to numerical solution of Klein-Gordon PDE. In the case of Klein-Gordon equation by keeping the mesh size fix, if we increase the number of iterations or either number of steps both iterative method can not improve the accuracy.

Iterative methods		MZ_2	HM
Number of iterations		1	1
Size of problem		200	200
Number of steps		32	33
Theoretical convergence-order(CO)		94	66
Number of function evaluations per iteration		34	34
Solutions of system of linear equations per iteration		32	32
Number of matrix vector multiplication per iteration		32	33
	λ		
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$	1	$3.62e-156$	$7.55e-110$
	2	$4.78e-142$	$2.31e-98$
	3	$3.91e-50$	$4.05e-35$
Execution time		23.48	24.0

TABLE 5.5: Comparison of performances for different multi-step methods in the case of the Bratu problem when number of function evaluations and number of solutions of systems of linear equations are equal in both iterative methods.

5.5 Summary

The inversion of Jacobian is computationally expensive and multi-step iterative methods can provide remedy to it, by offering good convergence-order with relatively less computational cost. The best way to construct a multi-step method is to reduce the number of Jacobian and function evaluations, inversion of Jacobian, matrix-vector and vector-vector multiplications. Higher-order Frechet derivatives are computationally expensive

Iterative methods	MZ_2	HM
Number of iterations	1	1
Size of problem	250	250
Number of steps	120	179
Theoretical convergence-order(CO)	358	358
Number of function evaluations per iteration	122	180
Solutions of system of linear equations per iteration	120	178
Number of matrix vector multiplication per iteration	120	179
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$, $(\lambda = 1)$	$3.98e-235$	$3.98e-235$
Execution time	59.67	70.22

TABLE 5.6: Comparison of performances for different multi-step methods in the case of the Bratu problem when convergence orders are equal in both itrative methods.

Iterative methods	MZ_2	HM
Number of iterations	3	3
Size of problem	250	250
Number of steps	3	4
Theoretical convergence-order(CO)	7	8
Computational convergence-order(COC)	6.75	7.81
Number of function evaluations per iteration	5	5
Solutions of system of linear equations per iteration	3	3
Number of matrix vector multiplication per iteration	3	4
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$	$8.44e-150$	$3.92e-161$
Execution time	63.75	64.66

TABLE 5.7: Comparison of performances for different multi-step methods in the case of the Bratu problem when number of function evaluations and number of solutions of systems of linear equations are equal in both iterative methods.

Iterative methods	MZ_2	HM
Number of iterations	3	3
Size of problem	150	150
Number of steps	3	4
Theoretical convergence-order(CO)	7	8
Computational convergence-order(COC)	7.39	8.64
Number of function evaluations per iteration	5	5
Solutions of system of linear equations per iteration	3	3
Number of matrix vector multiplication per iteration	3	4
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$	$4.21e-126$	$3.21e-149$
Execution time	16.10	16.68

TABLE 5.8: Comparison of performances for different multi-step methods in the case of the Frank Kamenetzkii problem when number of function evaluations and number of solutions of systems of linear equations are equal in both iterative methods.

Iterative methods	MZ_2	HM
Number of iterations	1	1
Size of problem	150	150
Number of steps	80	119
Theoretical convergence-order(CO)	238	238
Number of function evaluations per iteration	82	120
Solutions of system of linear equations per iteration	80	118
Number of matrix vector multiplication per iteration	80	119
$Max\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$, $(\lambda = 1)$	$6.46e-116$	$3.95e-99$
Execution time	19.89	28.21

TABLE 5.9: Comparison of performances for different multi-step methods in the case of the Frank Kamenetzkii problem when convergence orders are equal in both iterative methods.

Iterative methods		MZ_2	HM
Number of iterations		1	1
Size of problem		100	100
Number of steps		30	44
Theoretical convergence-order(CO)		88	88
Number of function evaluations per iteration		32	45
Solutions of system of linear equations per iteration		30	43
Number of matrix vector multiplication per iteration		30	44
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$	1	$1.95e-34$	$2.64e-37$
Execution time		3.01	3.53

TABLE 5.10: Comparison of performances for different multi-step methods in the case of the Lane-Emden equation when convergence orders are equal.

when we use them for the solution of general systems of nonlinear equations, but for a particular type of ODEs and PDEs we could use them because they are just diagonal matrices. The numerical accuracy in the solution of nonlinear systems enhances as we increase the number of step in MZ_2 method. The computational convergence-order of MZ_2 is also calculated in some examples and it agrees with the theoretical study of the convergence-order.

Iterative methods		MZ_2	HM
Number of iterations		1	1
Size of problem		4420	4420
Number of steps		4	4
Theoretical convergence-order(CO)		10	8
Number of function evaluations per iteration		6	5
Solutions of system of linear equations per iteration		4	3
Number of matrix vector multiplication per iteration		4	4
	Steps		
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$	1	$3.24e-1$	$4.11e-1$
	2	$7.51e-3$	$2.62e-3$
	3	$2.70e-5$	$2.63e-5$
	4	$5.59e-7$	$4.39e-7$
Execution time		94.13	80.18

TABLE 5.11: Comparison of performances for different multi-step methods in the case of the Klein Gordon equation , initial guess $u(x_i,t_j) = 0$, $u(x,t) = \delta sech(\kappa(x - \nu t)$, $\kappa = \sqrt{\frac{k}{c^2-\nu^2}}$, $\delta = \sqrt{\frac{2k}{\gamma}}$, $c = 1$, $\gamma = 1$, $\nu = 0.5$, $k = 0.5$, $n_x = 170$, $n_t = 26$, $x \in [-22,\ 22]$, $t \in [0,\ 0.5]$.

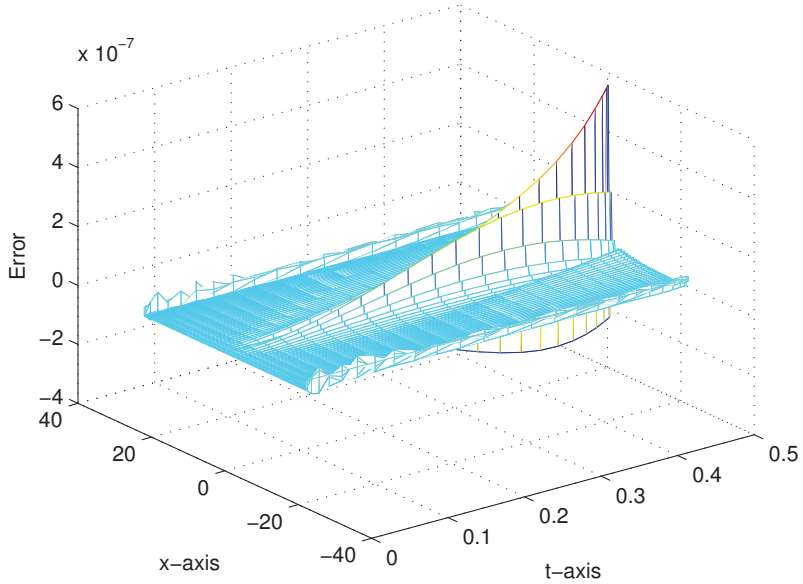

FIGURE 5.1: Absolute error plot for multi-step method MZ_2 in the case of the Klein Gordon equation , initial guess $u(x_i,t_j) = 0$, $u(x,t) = \delta sech(\kappa(x - \nu t)$, $\kappa = \sqrt{\frac{k}{c^2-\nu^2}}$, $\delta = \sqrt{\frac{2k}{\gamma}}$, $c = 1$, $\gamma = 1$, $\nu = 0.5$, $k = 0.5$, $n_x = 170$, $n_t = 26$, $x \in [-22,\ 22]$, $t \in [0,\ 0.5]$.

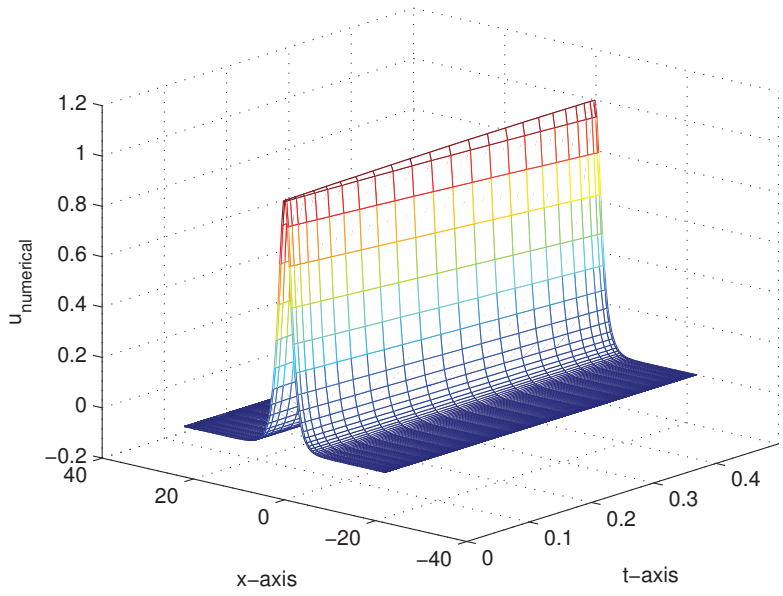

FIGURE 5.2: Numerical solution of the Klein Gordon equation , $x \in [-22,\ 22]$, $t \in [0,\ 0.5]$.

Chapter 6

Higher Order Multi-step Iterative Method for Computing the Numerical Solution of Systems of Nonlinear Equations: Application to Nonlinear PDEs and ODEs

We consider multi-step iterative method to solve systems of nonlinear equations. Since, the Jacobian evaluation and its inversion are expensive, in order to achieve best computational efficiency, we compute the Jacobian and its inverse only once in a single cycle of the proposed multi-step iterative method. Actually the involved systems of linear equations are solved by employing the LU-decomposition, rather than inversion. The primitive iterative method (termed base method) has convergence-order (CO) five and then we describe a matrix polynomial of degree two to design multi-step method. Each inclusion of singlestep in the base method will increase the convergence-order by three. The general expression for CO is $3s-1$, where s is the number of steps of multi-step iterative method. Computational efficiency is also addressed in comparison with other existing methods. The claimed convergence-rates proofs are established. The new contribution in our analysis relies essentially in the increment of CO by three for each added step, with a comparable computational cost in comparison with existing multi-steps iterative methods. Numerical assessments which justify the theoretical results are made: in particular, some systems of nonlinear equations associated with the numerical approximation of partial differential equations (PDEs) and

ordinary differential equations (ODEs) are built up and solved.

6.1 Introduction

Iterative methods for approximating the solution of systems of nonlinear equations represent an important research area, widely investigated in the relevant literature [103–105, 108, 109, 118, 120, 186, 187, 187–190, 193]. Several systems of nonlinear equations are originated from the numerical approximation of PDEs and ODEs. For instance, we may consider the Bratu problem, the Frank-Kamenetzkii problem [142], the Lene-Emden equation [184], the Burgers equation [194], and the Klein-Gordon equation [185], which are stated in order as

$$y''(x)+\lambda e^{y(x)}=0, \quad y(0)=y(1)=0, \tag{6.1}$$

$$y''(x)+\frac{1}{x}y'(x)+\lambda e^{y(x)}=0, \quad y'(0)=y(1)=0, \tag{6.2}$$

$$y''(x)+\frac{2}{x}y'(x)+y(x)^5=0, \quad y'(0)=0,\ y(0)=1, \tag{6.3}$$

$$\begin{aligned} &u_t(x,t)+u(x,t)u_x(x,t)-\gamma u_{xx}(x,t)=0, \quad u(x,0)=g_1(x),\\ &\quad u(0,t)=g_2(t), \quad u(2,t)=g_3(t), \end{aligned} \tag{6.4}$$

$$u_{tt}(x,t)-c^2u_{xx}(x,t)+f(u)=p(x,t), \quad -\infty<x<\infty, \quad t>0, \tag{6.5}$$

where $f(u)$ is odd function of u and $u(x,0)=f_1(x)$ and $u_t(x,0)=f_2(x)$. For the discretization of Equations (6.1), (6.2), (6.3), (6.4), and (6.5) we may use any suitable method in space and time. In this chapter, we employ the Chebyshev pseudo-spectral collocation method for temporal and spatial discretization. As a consequence, we obtain

the following systems of nonlinear equations

$$F(\boldsymbol{y}) = D_x^2\boldsymbol{y} + e^{\boldsymbol{y}} = 0, \tag{6.6}$$

$$F(\boldsymbol{y}) = \left(D_x^2 + diag\left(\frac{1}{\boldsymbol{x}}\right)D_x\right)\boldsymbol{y} + \lambda e^{\boldsymbol{y}} = 0, \tag{6.7}$$

$$F(\boldsymbol{y}) = \left(D_x^2 + diag\left(\frac{2}{\boldsymbol{x}}\right)D_x\right)\boldsymbol{y} + \lambda \boldsymbol{y}^5 = 0, \tag{6.8}$$

$$F(\boldsymbol{u}) = \left(\left(I_x \otimes D_t\right) - \gamma\left(D_x^2 \otimes I_t\right)\right)\boldsymbol{u} + diag\left(D_x\boldsymbol{u}\right)\boldsymbol{u} = 0, \tag{6.9}$$

$$F(\boldsymbol{u}) = \left(\left(I_x \otimes D_t^2\right) - c^2\left(D_x^2 \otimes I_t\right)\right)\boldsymbol{u} + f(\boldsymbol{u}) = 0, \tag{6.10}$$

where D_x, D_t are the differential matrices in spatial and temporal dimension, respectively, $\boldsymbol{y} = [y_1,\ y_2, \cdots,\ y_n]^T$, $\boldsymbol{x} = [x_1,\ x_2, \cdots,\ x_n]^T$, $f(\boldsymbol{y}) = [f(y_1),\ f(y_2), \cdots,\ f(y_n)]^T$ for $f(y) = e^y$ or y^n, $diag\left(\frac{1}{\boldsymbol{x}}\right)$ is a diagonal matrix with main diagonal entries of $\frac{1}{\boldsymbol{x}} = \left[\frac{1}{x_1},\ \frac{1}{x_2},\ \cdots,\ \frac{1}{x_n}\right]^T$, $\boldsymbol{u} = [u_{11},\ u_{12}, \cdots,\ u_{1m},\ u_{21}, \cdots,\ u_{2m}, \cdots,\ u_{n1},\ u_{n2}, \cdots,\ u_{nm}]^T$, I_x and I_t are identity matrices of dimensions n and m. We can write the systems of nonlinear equations (6.6), (6.7), (6.8), (6.9), (6.10) in compact form as

$$F(\boldsymbol{z}) = A\boldsymbol{z} + h(\boldsymbol{z}) = \boldsymbol{0}, \tag{6.11}$$

$$F'(\boldsymbol{z}) = A + diag(h'(\boldsymbol{z})), \tag{6.12}$$

where $A\boldsymbol{z}$ is a linear part , $h(\boldsymbol{z})$ is the nonlinear part of $F(\boldsymbol{z})$ and $F'(\boldsymbol{z})$ is first order Frechet derivative or Jacobian of $F(\boldsymbol{z})$. As we are dealing with general systems of nonlinear equations. Examine $F(\boldsymbol{x}) = [f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x}),\ f_3(\boldsymbol{x}), \cdots,\ f_n(\boldsymbol{x})]^T = \boldsymbol{0}$, with $f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x}),\ f_3(\boldsymbol{x}), \cdots,\ f_n(\boldsymbol{x})$ being coordinate continuously differentiable functions. Newton-Raphon (NR) is a classical iterative method [3, 191] for systems of nonlinear equations and it is defined as

$$\boldsymbol{x}_{q+1} = \boldsymbol{x}_q - F(\boldsymbol{x}_q)^{-1}F(\boldsymbol{x}_q), \quad q = 0,\ 1,\ 2,\ \cdots, \tag{6.13}$$

which is quadratically convergent, under suitable regularity assumptions. The multi-step version of NR can be written as

$$\begin{cases} \boldsymbol{y}_1 = \boldsymbol{x}_q - F(\boldsymbol{x}_q)^{-1}F(\boldsymbol{x}_q), \\ \boldsymbol{y}_2 = \boldsymbol{y}_1 - F(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_1), \\ \boldsymbol{y}_3 = \boldsymbol{y}_2 - F(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_2), \\ \vdots \\ \boldsymbol{x}_{q+1} = \boldsymbol{y}_{s-1} = \boldsymbol{y}_{s-2} - F(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_{s-2}). \end{cases} \tag{6.14}$$

One cycle of $(s-1)$-step NR method (6.14) requires one Jacobian, its inversion and s-function evaluations at the initial point with convergence-order s. However, quite recently, more efficient multi-step iterative methods, with better convergence-order, have been designed, by using the same number of Jacobian and function evaluations. As an example, the multi-step version of H. Montazeri et. al. (HM) [195] is written as

$$\begin{cases} \boldsymbol{y}_1 = \boldsymbol{x}_q - \frac{2}{3}F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{x}_q), \\ W = F'(\boldsymbol{x}_q)^{-1}F'(\boldsymbol{y}_1), \\ \boldsymbol{y}_2 = \boldsymbol{x}_q - \left(\frac{23}{8}I - 3W + \frac{9}{8}W^2\right)F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{x}_q), \\ \boldsymbol{y}_3 = \boldsymbol{y}_2 - \left(\frac{5}{2}I - \frac{3}{2}W\right)F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_2), \\ \boldsymbol{y}_4 = \boldsymbol{y}_3 - \left(\frac{5}{2}I - \frac{3}{2}W\right)F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_3), \\ \vdots \\ \boldsymbol{x}_{q+1} = \boldsymbol{y}_s = \boldsymbol{y}_{s-1} - \left(\frac{5}{2}I - \frac{3}{2}W\right)F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_{s-1}). \end{cases} \tag{6.15}$$

A single cycle of HM s-step iterative method requires $(s-1)$ functions evaluations, two Jacobian at different points, one inversion of Jacobian at initial guess and its

convergence-order is $2s$ for $s \geq 2$ and for $s = 1$, $CO = 1$. A further multi-step iterative method was developed by F. Soleymani et. al. (FS) [37]

$$\begin{cases} \boldsymbol{y}_1 = \boldsymbol{x}_q - \frac{2}{3}F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{x}_q), \\ W = \frac{1}{2}\Big(3F'(\boldsymbol{y}_1) - F'(\boldsymbol{x}_q)\Big)^{-1}(3F'(\boldsymbol{y}_1) + F'(\boldsymbol{x}_q)), \\ \boldsymbol{y}_2 = \boldsymbol{x}_q - WF'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{x}_q), \\ \boldsymbol{y}_3 = \boldsymbol{y}_2 - W^2F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_2), \\ \vdots \\ \boldsymbol{x}_{q+1} = \boldsymbol{y}_s = \boldsymbol{y}_{s-1} - W^2F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_{s-1}). \end{cases} \tag{6.16}$$

The s-step iterative method (FS) has $CO = 2s$ and computationally needs $(s-1)$ function evaluations, two Jacobian at different points, two inversion (solution of two systems of linear equations). Clearly the multi-step FS iterative method is better than the multi-step NS method , even if the multi-step HM method is computationally more efficient than both. In [192] M. Zaka and his co-workers proposed an iterative method to solve systems of nonlinear equations, but computational efficiency is not better than HM method, because of two Jacobian inversions at different points. In the next section, we propose our new multi-step iterative method which has better convergence order than the HM multi-step iterative method, with comparable computational cost.

6.2 New multi-step iterative method

Let us consider a general system of nonlinear equations $F(\boldsymbol{x}) = 0$, with the assumption that $F(\boldsymbol{x})$ is sufficiently differentiable. Our proposal for the new multi-step iterative

method (MZ) is the following

$$\begin{cases} \boldsymbol{y}_1 = \boldsymbol{x}_q - F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{x}_q), \\ V = (F'(\boldsymbol{x}_q))^{-1}F'(\boldsymbol{y}_1), \\ \boldsymbol{y}_2 = \boldsymbol{y}_1 - \left(\frac{13}{4}I - \frac{7}{2}V + \frac{5}{4}V^2\right)F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_1), \\ \boldsymbol{y}_3 = \boldsymbol{y}_2 - \left(\frac{7}{2}I - 4V + \frac{3}{2}V^2\right)F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_2), \\ \boldsymbol{y}_4 = \boldsymbol{y}_3 - \left(\frac{7}{2}I - 4V + \frac{3}{2}V^2\right)F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_3), \\ \vdots \\ \boldsymbol{x}_{q+1} = \boldsymbol{y}_s = \boldsymbol{y}_{s-1} - \left(\frac{7}{2}I - 4V + \frac{3}{2}V^2\right)F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{y}_{s-1}). \end{cases} \tag{6.17}$$

The new MZ procedure uses s function evaluations, two Jacobian and one inversion of Jacobian. We claim that the convergence-order of MZ is

$$CO = 3s - 1, \tag{6.18}$$

where s is step number. The multi-step HM iterative method is the best candidate for comparison, since it requires only a unique Jacobian inversion. For $s = 2$ the

	MZ ($s \geq 2$)	HM ($s \geq 2$)	MZ ($s = 2$)	HM ($s = 3$)	MZ ($s = s_1$)	HM ($s = s_1 + 1$)	Difference $MZ - HM$
Number of steps	s	s	2	3	s_1	$s_1 + 1$	1
Convergence-order	$3s-1$	$2s$	5	6	$3s_1 - 1$	$2(s_1 + 1)$	$s_1 - 3$
Function evaluations	s	$s-1$	2	2	s_1	s_1	0
Solution of system of linear equations when right hand side is vector	s	$s-1$	2	2	s_1	s_1	0
Solution of system of linear equations when right hand side is matrix	1	1	1	1	1	1	0
Jacobian evaluations	2	2	2	2	2	2	0
Matrix vector multiplications	$2(s-1)$	s	2	3	$2(s_1 - 1)$	$s_1 + 1$	$s_1 - 3$

TABLE 6.1: Comparison between multi-steps iterative method MZ and HM if number of function evaluations and solutions of system of nonlinear equations are equal.

convergence-orders of HM and MZ are 5 and 4, respectively, and the number of matrix vector multiplications are equal: however, with MZ, we have one more solution of system of linear equations and function evaluation, when compared with the HM method. The HM($s = 2,\ 3$) is better than MZ($s = 2$) because HM($s = 3$) uses the same number

of function evaluations and solution of system of linear equations as that of MZ($s=2$) with better convergence-order, even though HM($s=3$) requires one more matrix vector multiplication than MZ($s=2$). The computational cost of one matrix vector multiplication is negligible versus an increment in convergence-order. The multi-step iterative methods MZ($s=3$) and HM($s=4$) have the same convergence order with same number of function evaluations, solution of system of linear equations and number of matrix vector multiplications and this can be seen in the last column of Table 6.1. If in Table 6.1 $s_1 > 3$ then the convergence-order of MZ is better than HM but in the case of MZ we have to perform $s_1 - 3$ more matrix vector multiplication. Nevertheless, the enhancement in the convergence-order is also $s_1 - 3$, by keeping fixed the number of function evaluations and solution of systems of linear equations. Table 6.2 shows that, if the convergence-order is equal for MZ and HM, then we have to perform m steps more for HM which means we require $m-1$ more function evaluations and solution of system of linear equations, but $m-1$ less matrix vector multiplications. The computational cost of $m-1$ matrix vector multiplication is not higher than $m-1$ function evaluations and solutions of systems of linear equations. We conclude that our proposal for multi-steps iterative method MZ is better than existing multi-steps iterative methods in our literature review.

	MZ ($m \geq 1$)	HM ($m \geq 1$)	Difference $HM - MZ$
Number of steps	$2m+1$	$3m+1$	m
Convergence-order	$6m+2$	$6m+2$	0
Function evaluations	$2m+1$	$3m$	$m-1$
Solution of system of linear equations when right hand side is vector	$2m+1$	$3m$	$m-1$
Solution of system of linear equations when right hand side is matrix	1	1	0
Jacobian evaluations	2	2	0
Matrix vector multiplications	$4m$	$3m+1$	$-m+1$

TABLE 6.2: Comparison between multi-steps iterative method MZ and HM if number convergence-orders are equal.

If n is the size of system of nonlinear equations and we adopt the definition of performance index as

$$\rho = CO^{\frac{1}{flops}}, \tag{6.19}$$

then Table 6.3 shows the performance index in terms of floating-point operations per second for different number of steps. In Figure 6.1, we compare the performance index between MZ and HM multi-steps iterative methods. The curve of performance index related to MZ is better than that associated with the method HM.

Iterative methods	HM	MZ	HM	MZ
Number of steps	4	3	5	4
Rate of convergence	8	8	10	11
Number of functional evaluations	$3n+2n^2$	$3n+2n^2$	$4n+2n^2$	$4n+2n^2$
The classical efficiency index	$2^{1/(3n+2n^2)}$	$2^{1/(3n+2n^2)}$	$2^{1/(4n+2n^2)}$	$2^{1/(4n+2n^2)}$
Number of Lu factorizations	1	1	1	1
Cost of Lu factorizations	$\frac{2n^3}{3}$	$\frac{2n^3}{3}$	$\frac{2n^3}{3}$	$\frac{2n^3}{3}$
Cost of linear systems	$\frac{5n^3}{3}+6n^2$	$\frac{5n^3}{3}+6n^2$	$\frac{5n^3}{3}+8n^2$	$\frac{5n^3}{3}+8n^2$
Matrix vector multiplications	$4n^2$	$4n^2$	$5n^2$	$6n^2$
Flops-like efficiency index	$8^{1/(\frac{5n^3}{3}+12n^2+3n)}$	$8^{1/(\frac{5n^3}{3}+12n^2+3n)}$	$10^{1/(\frac{5n^3}{3}+15n^2+4n)}$	$11^{1/(\frac{5n^3}{3}+16n^2+4n)}$

TABLE 6.3: Comparison of performance index between multi-steps iterative methods MZ and HM.

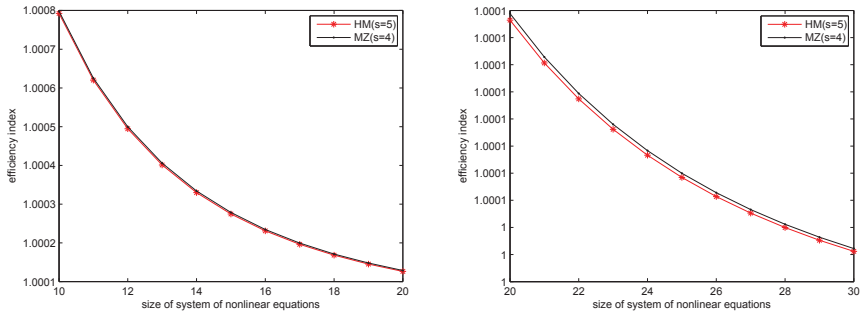

FIGURE 6.1: Comparison between performance index of MZ and HM multi-steps iterative methods.

6.3 Convergence analysis

In this section, we will prove that the local convergence-order of the technique reported in (6.17) is eight for $s = 3$ and later we will establish a proof for the multi-step iterative

scheme (6.17), by using mathematical induction.

Theorem 6.1. *Let $F : \Gamma \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be sufficiently Frechet differentiable on an open convex neighborhood Γ of $\boldsymbol{x}^* \in \mathbb{R}^n$ with $F(\boldsymbol{x}^*) = 0$ and $det(F'(\boldsymbol{x}^*)) \neq 0$. Then the sequence $\{\boldsymbol{x}_q\}$ generated by the iterative scheme (6.17) converges to $\boldsymbol{x}^*$ with local order of convergence eight, and produces the following error equation*

$$\boldsymbol{e}_{q+1} = L\boldsymbol{e}_q{}^8 + O(\boldsymbol{e}_q{}^9), \tag{6.20}$$

where $\boldsymbol{e}_q = \boldsymbol{x}_q - \boldsymbol{x}^*$, $\boldsymbol{e}_q{}^p = \overbrace{(\boldsymbol{e}_q, \boldsymbol{e}_q, \cdots, \boldsymbol{e}_q)}^{p\text{-times}}$ and $L = (280C_2^7 + (1/2)(C_2C_3C_2C_3C_2) - (1/2)(3C_2(C_3^2)(C_2^2)) - 42C_3(C_2^5) - 30C_2^3C_3(C_2^2) + (1/2)(9C_3C_2C_3(C_2^2)) + 14C_2C_3(C_2^4) + 10C_2^4C_3C_2 - (1/2)(3C_3(C_2^2)C_3C_2))$ is a p-linear function i.e. $L \in \mathbb{L}\overbrace{(\mathbb{R}^n, \mathbb{R}^n, \cdots, \mathbb{R}^n)}^{p\text{-times}}$ and $L\boldsymbol{e}_q{}^p \in \mathbb{R}^n$.

Proof. Let $F : \Gamma \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be sufficiently Frechet differentiable function in Γ. The *ith* Frechet derivative of F at $v \in \mathbb{R}^n$, $i \geq 1$, is the $i-linear$ function $F^{(q)}(v) : \overbrace{\mathbb{R}^n\mathbb{R}^n \cdots \mathbb{R}^n}^{\text{i-times}}$ such that $F^{(i)}(v)(u_1, u_2, \cdots, u_i) \in \mathbb{R}^n$ [154]. The Taylor's series expansion of $F(\boldsymbol{x}_q)$ around $\boldsymbol{x}^*$ can be written as:

$$\begin{aligned}
F(\boldsymbol{x}_q) &= F(\boldsymbol{x}^* + \boldsymbol{x}_q - \boldsymbol{x}^*) = F(\boldsymbol{x}^* + \boldsymbol{e}_q), && (6.21)\\
&= F(\boldsymbol{x}^*) + F'(\boldsymbol{x}^*)\boldsymbol{e}_q + \frac{1}{2!}F''(\boldsymbol{x}^*)\boldsymbol{e}_q{}^2 + \frac{1}{3!}F^{(3)}(\boldsymbol{x}^*)\boldsymbol{e}_q{}^3 + \cdots, && (6.22)\\
&= F'(\boldsymbol{x}^*)\Big[\boldsymbol{e}_q + \frac{1}{2!}F'(\boldsymbol{x}^*)^{-1}F''(\boldsymbol{x}^*)\boldsymbol{e}_q{}^2 + \frac{1}{3!}F'(\boldsymbol{x}^*)^{-1}F^{(3)}(\boldsymbol{x}^*)\boldsymbol{e}_q{}^3 + \cdots\Big], && (6.23)\\
&= C_1\Big[\boldsymbol{e}_q + C_2\boldsymbol{e}_q{}^2 + C_3\boldsymbol{e}_q{}^3 + \cdots\Big], && (6.24)
\end{aligned}$$

where $C_1 = F'(\boldsymbol{x}^*)$ and $C_s = \frac{1}{s!}F'(\boldsymbol{x}^*)^{-1}F^{(s)}(\boldsymbol{x}^*)$ for $s \geq 2$. From (6.21), we can compute the Frechet derivative of F:

$$F'(\boldsymbol{x}_q) = C_1\Big[I + 2C_2\boldsymbol{e}_q + 3C_3\boldsymbol{e}_q{}^2 + 4C_3\boldsymbol{e}_q{}^3 + \cdots\Big], \tag{6.25}$$

where I is the identity matrix. Furthermore, we calculate the inverse of the Jacobian matrix

$$\begin{aligned}F'(\boldsymbol{x}_q)^{-1} &= \big[I-2C_2\boldsymbol{e}_q+(4C_2^2-3C_3)\boldsymbol{e}_q^2+(-8C_2^3+6C_3C_2+6C_2C_3-4C_4)\boldsymbol{e}_q^3\\&+(-12C_3C_2^2-12C_2^2C_3-12C_2C_3C_2+8C_4C_2+9C_3^2+8C_2C_4+16C_2^4-5C_5)\boldsymbol{e}_q^4\\&+(-16C_4C_2^2-18C_3C_2C_3-18C_3^2C_2-18C_2C_3^2-16C_2^2C_4-16C_2C_4C_2+24C_2^3C_3\\&+24C_2^2C_3C_2+24C_2C_3C_2^2-32C_2^5+10C_5C_2+12C_4C_3+12C_3C_4+10C_2C_5\\&+24C_3C_2^3-6C_6)\boldsymbol{e}_q^5+(-24C_3C_4C_2-24C_3C_2C_4-27C_3^3-48C_3C_2^4+64C_2^6-24C_4C_3C_2\\&-24C_4C_2C_3-20C_2C_5C_2-24C_2C_4C_3+32C_4C_2^3-24C_2C_3C_4-20C_2^2C_5+36C_2^2C_3^2\\&+32C_2^3C_4-48C_2^2C_3C_2^2+36C_2C_3C_2C_3-48C_2^3C_3C_2-48C_2^4C_3-48C_2C_3C_2^3\\&+36C_3C_2^2C_3+36C_3C_2C_3C_2+12C_2C_6+16C_4^2+15C_3C_5+15C_5C_3+12C_6C_2\\&+36C_2C_3^2C_2+32C_2C_4C_2^2-7C_7+36C_3^2C_2^2+32C_2^2C_4C_2-20C_5C_2^2)\boldsymbol{e}_q^6+\cdots\big]C_1^{-1}\end{aligned} \tag{6.26}$$

By multiplying $F'(\boldsymbol{x}_q)^{-1}$ and $F(\boldsymbol{x}_q)$, we obtain:

$$\begin{aligned}F'(\boldsymbol{x}_q)^{-1}F(\boldsymbol{x}_q) &= \boldsymbol{e}_q-C_2\boldsymbol{e}_q^2+(2C_2^2-2C_3)\boldsymbol{e}_q^3+(4C_2C_3+3C_3C_2-3C_4-4C_2^3)\boldsymbol{e}_q^4\\&+(8C_2^4-6C_3C_2^2+6C_3^2+6C_2C_4+4C_4C_2-4C_5-8C_2^2C_3-6C_2C_3C_2)\boldsymbol{e}_q^5\\&+(-12C_3C_2C_3-8C_2C_4C_2-12C_2^2C_4-12C_2C_3^2+12C_3C_2^3-16C_2^5\\&-9C_3^2C_2+8C_2C_5+9C_3C_4+8C_4C_3+5C_5C_2-5C_6-8C_4C_2^2+16C_2^3C_3\\&+12C_2^2C_3C_2+12C_2C_3C_2^2)\boldsymbol{e}_q^6+\cdots.\end{aligned} \tag{6.27}$$

From (6.27) we have

$$\begin{aligned}\boldsymbol{y}_1-\boldsymbol{x}^* &= C_2\boldsymbol{e}_q^2+(-2C_2^2+2C_3)\boldsymbol{e}_q^3+(3C_4+4C_2^3-3C_3C_2-4C_2C_3)\boldsymbol{e}_q^4+(4C_5\\&+6C_3(C_2^2)+8C_2^2C_3+6C_2C_3C_2-8C_2^4-4C_4C_2-6C_3^2-6C_2C_4)\boldsymbol{e}_q^5\\&+(5C_6+8C_4(C_2^2)+12C_3C_2C_3+9C_3^2C_2+12C_2(C_3^2)+12C_2^2C_4+8C_2C_4C_2\\&-16C_2^3C_3-12C_2^2C_3C_2-12C_2C_3(C_2^2)-12C_3(C_2^3)+16C_2^5-5C_5C_2\\&-8C_4C_3-9C_3C_4-8C_2C_5)\boldsymbol{e}_q^6+\cdots.\end{aligned} \tag{6.28}$$

The expression for V is

$$\begin{aligned}V &= I - 2C_2\boldsymbol{e}_q + (6C_2^2 - 3C_3)\boldsymbol{e}_q^2) + (-16C_2^3 + 10C_2C_3 + 6C_3C_2 - 4C_4)\boldsymbol{e}_q^3 \\ &+ (-5C_5 + 8C_4C_2 + 14C_2C_4 - 18C_2C_3C_2 + 9C_3^2 + 40C_2^4 - 15C_3(C_2^2) \\ &- 28C_2^2C_3)\boldsymbol{e}_q^4 + (-6C_6 + 10C_5C_2 + 12C_4C_3 + 12C_3C_4 + 18C_2C_5 \\ &- 24C_3C_2C_3 - 24C_2C_4C_2 + 72C_2^3C_3 + 48C_2^2C_3C_2 + 42C_2C_3(C_2^2) \\ &+ 36C_3(C_2^3) - 96C_2^5 - 24C_4(C_2^2) - 12C_3^2C_2 - 30C_2(C_3^2) - 40C_2^2C_4)\boldsymbol{e}_q^5 \\ &+ (-52C_2^2C_5 - 42C_2C_3C_4 - 40C_2C_4C_3 - 30C_2C_5C_2 - 15C_3C_4C_2 \\ &- 33C_3C_2C_4 - 15C_3^3 + 224C_2^6 - 24C_4C_3C_2 - 40C_4C_2C_3 - 176C_2^4C_3 \\ &- 120C_2^3C_3C_2 - 108C_2^2C_3(C_2^2) - 96C_2C_3(C_2^3) + 68C_4(C_2^3) - 84C_3(C_2^4) \\ &+ 22C_2C_6 + 16C_4^2 + 15C_3C_5 + 15C_5C_3 + 12C_6C_2 + 24C_3^2(C_2^2) \\ &+ 60C_3(C_2^2)C_3 + 33C_3C_2C_3C_2 + 64C_2^2C_4C_2 - 7C_7 + 104C_2^3C_4 \\ &+ 84C_2^2(C_3^2) + 42C_2(C_3^2)C_2 + 72C_2C_3C_2C_3 + 64C_2C_4(C_2^2) \\ &- 30C_5(C_2^2))\boldsymbol{e}_q^6 + \cdots . \end{aligned} \tag{6.29}$$

Finally, we provide the expression for $\boldsymbol{y}_2 - \boldsymbol{x}^*$ and $\boldsymbol{y}_3 - \boldsymbol{x}^*$, by skipping some steps:

$$\begin{aligned}\boldsymbol{y}_2 - \boldsymbol{x}^* &= (-3C_3(C_2^2)(1/2) + (1/2)(C_2C_3C_2) + 14C_2^4)\boldsymbol{e}_q^5 + (C_2C_4C_2 + C_2(C_3^2) \\ &- 9C_3^2C_2(1/4) - 3C_3C_2C_3 + 55C_3(C_2^3)(1/2) + 19C_2C_3(C_2^2) + (1/2)(41C_2^2C_3C_2) \\ &+ 28C_2^3C_3 - 2C_4(C_2^2) - 140C_2^5)\boldsymbol{e}_q^6 + \cdots . \end{aligned} \tag{6.30}$$

$$\begin{aligned}\boldsymbol{y}_3 - \boldsymbol{x}^* &= (280C_2^7 + (1/2)(C_2C_3C_2C_3C_2) - (1/2)(3C_2(C_3^2)(C_2^2)) - 42C_3(C_2^5) \\ &- 30C_2^3C_3(C_2^2) + (1/2)(9C_3C_2C_3(C_2^2)) + 14C_2C_3(C_2^4) + 10C_2^4C_3C_2 \\ &- (1/2)(3C_3(C_2^2)C_3C_2))\boldsymbol{e}_q^8 + \cdots . \end{aligned} \tag{6.31}$$

□

Theorem 6.2. *The multi-step iterative scheme (6.17) has the local convergence-order* $3s-1$*, using* $s(\geq 1)$ *evaluations of a sufficiently differentiable function* F *and two first-order Frechet derivatives* F' *per full-cycle.*

Proof. The proof is established via mathematical induction. For $s = 1,\ 2,\ 3$ the convergence-orders are two, five and eight from (6.28), (6.30), and (6.31), respectively. Consequently our claim concerning the convergence-order $3s-1$ is true for $s = 2,\ 3$.

We assume that our claim is true for $s = m > 3$, i.e., the convergence-order of (6.17) is $3m-1$. The mth-step and $(m-1)th$-step of iterative scheme(6.17) can be written as:

$$Frozen - factor = \left(\frac{7}{2}I - 4V + \frac{3}{2}V^2\right)(F'(\boldsymbol{x}_q))^{-1}, \tag{6.32}$$

$$\boldsymbol{y}_{m-1} = \boldsymbol{y}_{m-2} - (Frozen - factor)F(\boldsymbol{y}_{m-2}), \tag{6.33}$$

$$\boldsymbol{y}_m = \boldsymbol{y}_{m-1} - (Frozen - factor)F(\boldsymbol{y}_{m-1}). \tag{6.34}$$

The enhancement in the convergence-order of (6.17) from the $(m-1)th$-step to the mth-step is $(3m-1)-(3(m-1)-1) = 3$. Now we write the $(m+1)th$-step of (6.17):

$$\boldsymbol{y}_{m+1} = \boldsymbol{y}_m - (Frozen - factor)F(\boldsymbol{y}_m). \tag{6.35}$$

The increment in the convergence-order of (6.17), due to $(m+1)th$-step, is exactly three, because the use of the $Frozen - factor$ adds an additive constant in the convergence-order [37]. Finally, after the addition of the $(m+1)th$-step, we observe that the convergence-order is $3m-1+3 = 3m+2$, which completes the proof. □

6.4 Dynamics of multi-steps iterative methods

Here we analyze the dynamics of classical and newly developed multi-step iterative methods. Actually dynamics of iterative solvers for nonlinear problems shows the region of convergence and divergence. In order to draw the convergence and divergence regions, we select two simple systems of nonlinear equations

$$P_1 = \begin{cases} \frac{x^2}{16} + \frac{y^2}{4} = 1 \\ y - 4\sin(x) = 0, \end{cases} \tag{6.36}$$

$$P_2 = \begin{cases} y - x^2 - 1 = 0 \\ y + x^2 - 1 = 0. \end{cases} \tag{6.37}$$

The P_1 problem has six roots and P_2 has only two roots. Regarding the dynamics plots, we start with an initial guess and iterates it with a given iterative method. If iterations show convergence to a root we assign a specific color (different from black) to that initial guess. Otherwise, in case of divergence, we employ the black color. Notice that the nonlinear curves of P_1 and P_2 are also plotted in black color in all figures which

has no connection with divergence. In Figures 6.2, 6.3, 6.4, we plotted the convergence and divergence regions of classical Newton Raphson iterative methods, with different convergence orders for problem P_1. The multi-step iterative method MZ dynamics is shown in Figures 6.5, 6.6 for problem P_1 and in Figures 6.7, 6.8 for problem P_2. As we increase the convergence-order of an iterative method, the figures clearly show that its region of convergence around the roots shrink. Generally specking all higher order methods are sensitive to initial guess because of their narrow convergence regions. Figure 6.6 corresponds to multi-step iterative method MZ($s = 3$, $CO = 8$) and has more dark region than that in Figure 6.2. It means that, as expected, our multi-step iterative method is more sensitive to the initial guess than classical Newton Raphson.

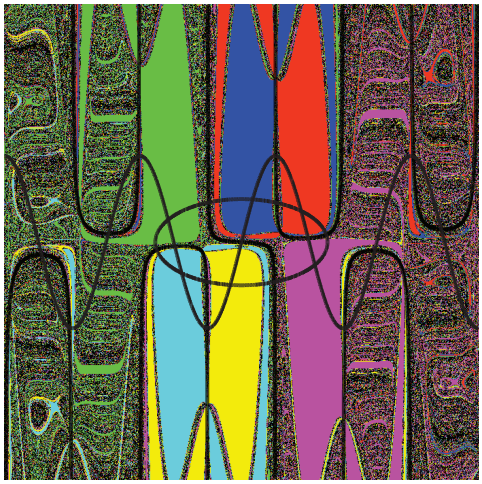

FIGURE 6.2: Newton Raphon with $CO = 2$, Domain$= [-11,\ 11] \times [-11,\ 11]$, Grid$=$ 700×700.

6.5 Numerical tests

For the verification of convergence-order, we use the following definition for the computational convergence-order (COC):

$$COC \approx \frac{log\left[Max(|\boldsymbol{x}_{q+2} - \boldsymbol{x}^*|)/Max(|\boldsymbol{x}_{q+1} - \boldsymbol{x}^*|)\right]}{log\left[Max(|\boldsymbol{x}_{q+1} - \boldsymbol{x}^*|)/Max(|\boldsymbol{x}_q - \boldsymbol{x}^*|)\right]}, \tag{6.38}$$

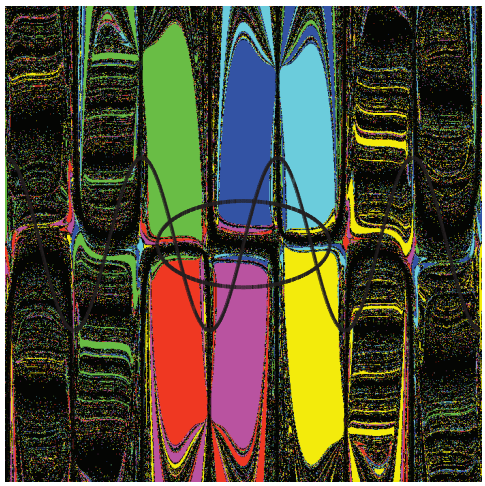

FIGURE 6.3: Multi-step Newton Raphon with $CO = 3$, Domain$= [-11,\ 11] \times [-11,\ 11]$, Grid$= 700 \times 700$.

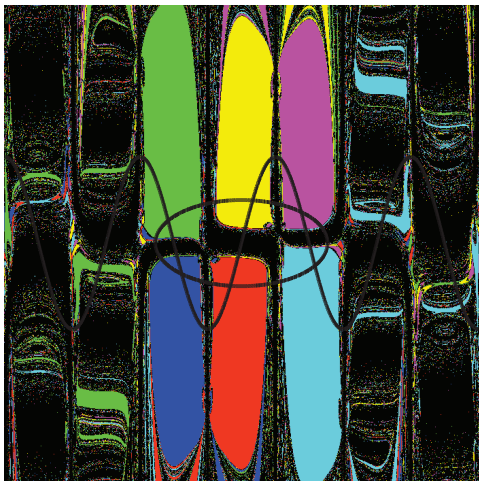

FIGURE 6.4: Multi-step Newton Raphon with $CO = 4$, Domain$= [-11,\ 11] \times [-11,\ 11]$, Grid$= 700 \times 700$.

FIGURE 6.5: Multi-step iterative method MZ with $CO = 5$, Domain$= [-11,\ 11] \times [-11,\ 11]$, Grid$= 700 \times 700$.

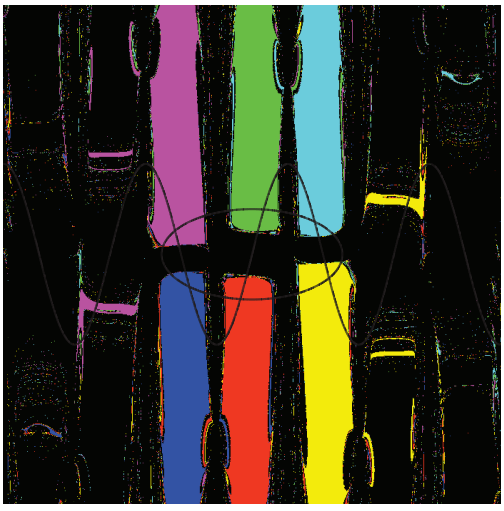

FIGURE 6.6: Multi-step iterative method MZ with $CO = 8$, Domain$= [-11,\ 11] \times [-11,\ 11]$, Grid$= 700 \times 700$.

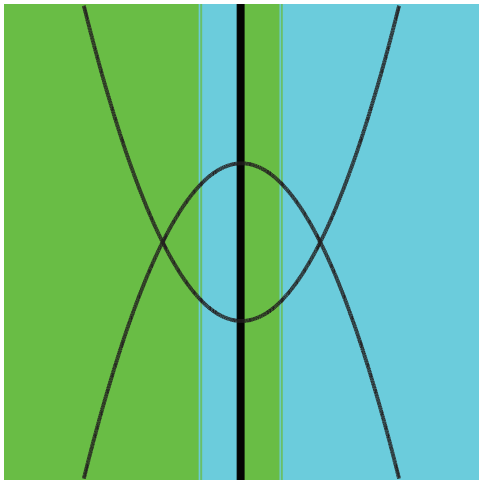

FIGURE 6.7: Multi-step iterative method MZ with $CO = 5$, Domain$= [-3,\ 3] \times [-3,\ 3]$, Grid$= 300 \times 300$.

FIGURE 6.8: Multi-step iterative method MZ with $CO = 8$, Domain$= [-3,\ 3] \times [-3,\ 3]$, Grid$= 300 \times 300$.

where $Max(|\boldsymbol{x}_{q+2} - \boldsymbol{x}^*|)$ is the maximum absolute error. In the introduction section we have listed three ODEs and two PDEs problem. The analytical solution of (6.1), (6.2), (6.3), (6.5) can be found in [154], [183], [184], [185], respectively. The close form solution of (6.4) is not known: in this case we can check the norm of $F(\boldsymbol{x})$. The proposed iterative methods are designed for general systems of nonlinear equations. Hence we choose a small system of nonlinear equations for checking the computational order of convergence and accuracy of multi-step MZ method:

$$\begin{cases} x_2x_3 + x_4(x_2 + x_3) = 0, \\ x_1x_3 + x_4(x_1 + x_3) = 0, \\ x_1x_2 + x_4(x_1 + x_2) = 0, \\ x_1x_2 + x_1x_3 + x_2x_3 - 1 = 0. \end{cases} \tag{6.39}$$

For the purpose of testing accuracy, we give here thirty digits accurate solution of (6.39)

$$\begin{cases} x_1 = 0.57735026918962576450914878050 2, \\ x_2 = 0.57735026918962576450914878050 2, \\ x_3 = 0.57735026918962576450914878050 2, \\ x_4 = -0.28867513459481288225457439025 1. \end{cases} \tag{6.40}$$

The initial and boundary conditions for PDEs (6.4) and (6.5) can be found in the caption of Tables 6.9 and 6.10. For the purpose of comparison we adopted two ways: either (i) both multi-step iterative methods MZ and HM use the same number of function evaluations or (ii) the convergence-order is forced to be equal. In some cases we just perform one iteration and increase the number of steps. Otherwise, we perform more iterations with different number of steps. When we perform a single iteration of multi-steps methods, by increasing the number of steps to achieve better accuracy, we pay minimum computational cost because, in this case, we have to perform just one LU-factorization of a single Jacobian and have to reuse the factors for the solution of system of linear equations. The first problem which we try to solve is the Bratu problem (6.1). In Table 6.4, we use 200 grid points to discretize the domain of the problem $[0,\ 1]$ and each grid point corresponds to a nonlinear equation. In this we way the system of nonlinear equations has size 200. We performed one iteration and select 32, 33 steps for MZ and HM method so that both methods have same number of function

evaluations and solutions of systems of linear equations. For different values of the parameter λ for Bratu problem in Table 6.4, results shows that MZ performs better than HM in terms of execution time and achieved accuracy. In Table 6.4 we solved Bratu problem for different values of parameter λ for fix number of steps. One may observe that if we increase the value of λ there is degradation in accuracy for fixed number of steps. Actually for different values of parameter Bratu problem has different solutions and we solve all the problem with same initial guess and achieve different accuracies in the numerical solutions for fixed number of multi-steps. The dynamics of iterative method for different problems is different so same initial guess for all problems may provide different accuracy. For the better comparison of execution time of the methods MZ and HM, we select different number of steps such that both methods have the same convergence-order. Table 6.5 depicts the execution time of MZ, which is less than that of the HM method. The reason why is that the HM procedure requires a larger number of steps to achieve equal convergence-order as MZ and for each step we solve one system of linear equations and one function evaluation. It is true that MZ uses more matrix vector multiplications than HM, but more function evaluations and solutions of systems of linear equations are performed by the HM solver. The successive iterations of MZ method for Bratu problem is given in Figure 6.9 and the analytical solution is plotted in Figure 6.10. The successive iterations and solution of Frank Kamenetzkii problem is shown in Figures 6.11, 6.12. In Table 6.6 we also calculated the COCs for both MZ and HM methods and they verify the claimed theoretical convergence-orders. The results of Table 6.7 also confirm the fast convergence of the MZ method. The first two problems were boundary value problems, while the third one is the initial value problem Lene-Emden, having infinite. For numerical experimentation we select a reasonable closed interval $[0,\ 8]$ to integrate the problem. In Table 6.8 we execute two iterations by fixing the function evaluations and solution of system of linear equations. The results show that the execution time is more or less the same, but the resulting accuracy of the MZ method is better than that of the HM method. Figure 6.13 and 6.14 present the numerical treatment of Lene-Emden equation. For the numerical solution of Burgers equations we select $n_x = 40$ grid points in spatial dimension and $n_t = 40$ in the temporal dimension so that the size of the resulting system of nonlinear equations becomes 1600. Table 6.9 tells that MZ achieved $O(10^{-14})$ in the $2-$norm of $F(\boldsymbol{x})$ of Burgers equations, while HM got the same accuracy in three iterates and consumed more time. Error in the $||F(x)||_2$ of Burgers equation and approximated numerical solution can be visualized in Figures 6.15 and 6.16. The largest system of nonlinear equations is constructed for the Klein-Gordon equations. In Table 6.10 we iterate MZ and HM methods

one single time and both consume almost the same execution time: however the MZ procedure got better maximum absolute error than HM. The absolute error plot related to the analytical solution of Klien-Gordon can be seen in Figures 6.17 and 6.18. Since we have mentioned that our proposed multi-step iterative method is designed for general systems of nonlinear equations, in (6.39) we picked a general system of nonlinear equations and in (6.40) a thirty digits accurate solution of it is also supplied. The results of Table 6.11 confirms the COCs for both methods and because the convergence-order of MZ is higher than HM, by using the same number of function evaluations, the resulting accuracy is also better than HM. Table 6.12 shows the successive iterations for both methods.

Iterative methods		MZ	HM
Number of iterations		1	1
Size of problem		200	200
Number of steps		32	33
Theoretical convergence-order(CO)		95	66
Number of function evaluations per iteration		32	32
Solutions of system of linear equations per iteration		32	32
Number of Jacobian evaluations per iteration		2	2
Number of Jacobian LU-factorizations per iteration		1	1
Number of matrix vector multiplication per iteration		62	33
	λ		
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$	1	$4.66e-161$	$7.55e-110$
	2	$2.24e-140$	$2.31e-98$
	3	$8.41e-48$	$4.05e-35$
Execution time$\lambda = 1$)		23.72	22.31

TABLE 6.4: Comparison of performances for different multi-step methods in the case of the Bratu problem (6.1) when number of function evaluations and number of solutions of systems of linear equations are equal.

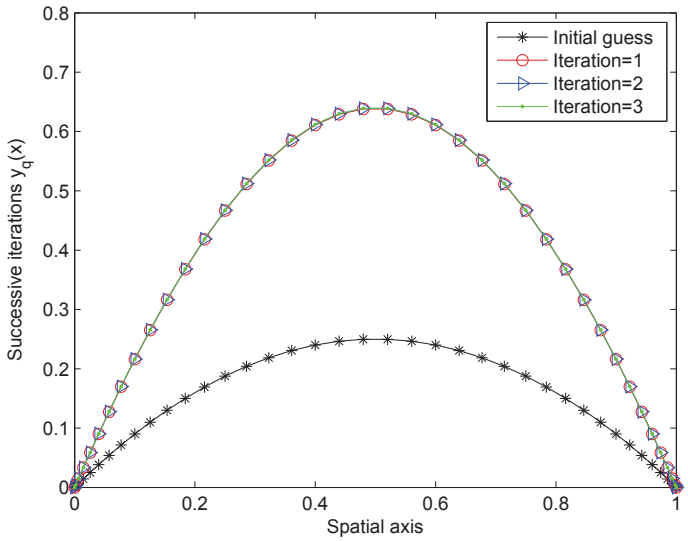

FIGURE 6.9: successive iterations of multi-step method MZ in the case of the Bratu problem (6.1), $\lambda = 3$, $iter = 3$, $step = 2$, size of problem $= 40$.

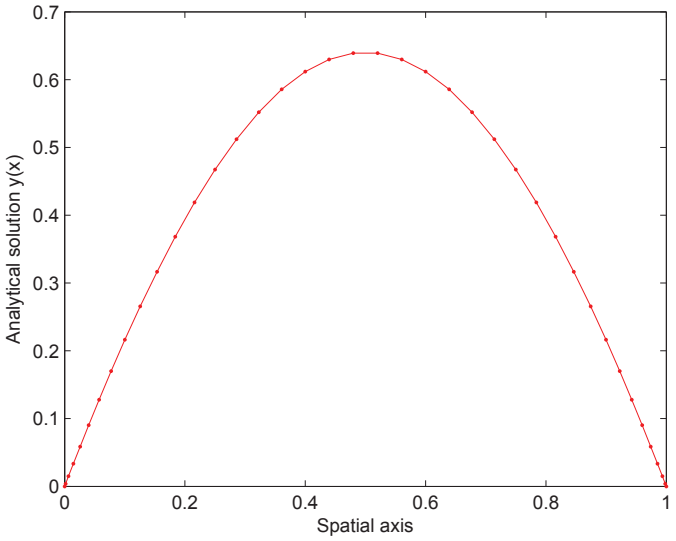

FIGURE 6.10: Analytical solution of Bratu problem (6.1), $\lambda = 3$, $iter = 3$, $step = 2$, size of problem $= 40$.

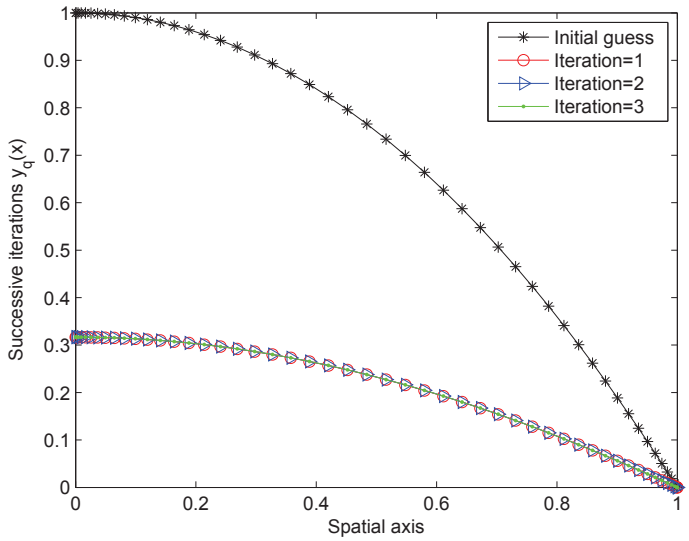

FIGURE 6.11: successive iterations of multi-step method MZ in the case of the Frank Kamenetzkii problem (6.1), $iter = 3$, $step = 2$, size of problem $= 50$.

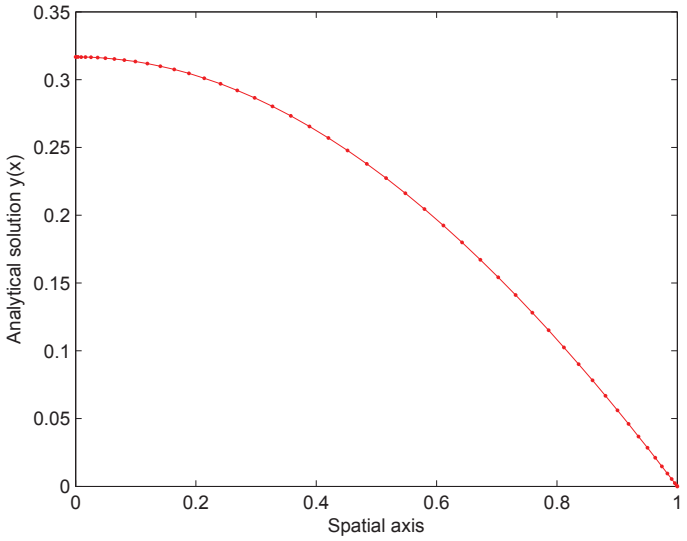

FIGURE 6.12: Analytical solution of Frank Kamenetzkii problem (6.1), $iter = 3$, $step = 2$, size of problem $= 50$.

Iterative methods	MZ	HM
Number of iterations	1	1
Size of problem	250	250
Number of steps	121	181
Theoretical convergence-order(CO)	362	362
Number of function evaluations per iteration	121	180
Solutions of system of linear equations per iteration	121	180
Number of Jacobian evaluations per iteration	2	2
Number of Jacobian LU-factorizations per iteration	1	1
Number of matrix vector multiplication per iteration	240	181
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$, $(\lambda = 1)$	$3.98e-235$	$3.98e-235$
Execution time	65.51	71.00

TABLE 6.5: Comparison of performances for different multi-step methods in the case of the Bratu problem (6.1) when convergence orders are equal.

Iterative methods	MZ	HM
Number of iterations	3	3
Size of problem	150	150
Number of steps	3	4
Theoretical convergence-order(CO)	8	8
Computational convergence-order(COC)	7.77	8.65
Number of function evaluations per iteration	3	3
Solutions of system of linear equations per iteration	3	3
Number of Jacobian evaluations per iteration	2	2
Number of Jacobian LU-factorizations per iteration	1	1
Number of matrix vector multiplication per iteration	4	4
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$	$5.12e-149$	$3.21e-149$
Execution time	16.44	16.81

TABLE 6.6: Comparison of performances for different multi-step methods in the case of the Frank Kamenetzkii problem (6.2) when convergence orders, number of function evaluations, number of solutions of systems of linear equuations are equal.

Iterative methods	MZ	HM
Number of iterations	1	1
Size of problem	150	150
Number of steps	121	181
Theoretical convergence-order(CO)	362	362
Number of function evaluations per iteration	121	180
Solutions of system of linear equations per iteration	121	180
Number of Jacobian evaluations per iteration	2	2
Number of Jacobian LU-factorizations per iteration	1	1
Number of matrix vector multiplication per iteration	240	181
$Max\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$, $(\lambda = 1)$	$6.46e-148$	$3.95e-149$
Execution time	30.15	35.45

TABLE 6.7: Comparison of performances for different multi-step methods in the case of the Frank Kamenetzkii problem (6.2) when convergence orders are equal.

Iterative methods		MZ	HM
Number of iterations		2	2
Size of problem		150	150
Number of steps		7	8
Theoretical convergence-order(CO)		20	16
Number of function evaluations per iteration		7	7
Solutions of system of linear equations per iteration		7	7
Number of Jacobian evaluations per iteration		2	2
Number of Jacobian LU-factorizations per iteration		1	1
Number of matrix vector multiplication per iteration		12	8
	Iter		
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$	1	0.50	0.68
	2	$4.77e-32$	$1.18e-17$
Execution time		13.33	13.12

TABLE 6.8: Comparison of performances for different multi-step methods in the case of the Lene-Emden equation (6.3)

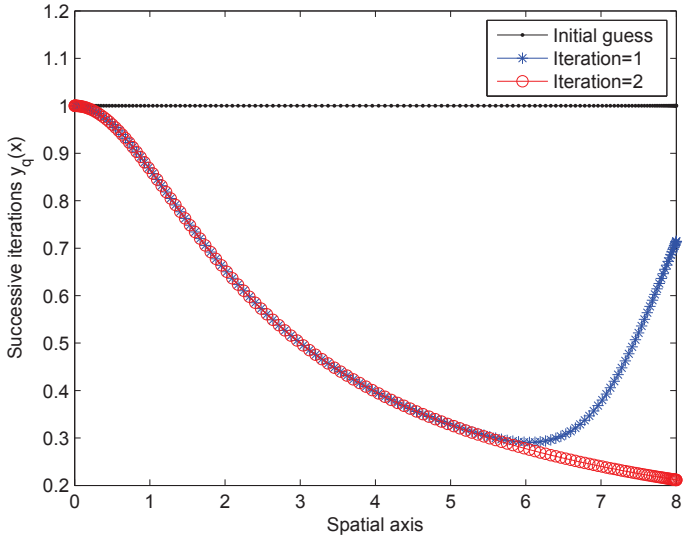

FIGURE 6.13: successive iterations of multi-step method MZ in the case of the Lene-Emden equation (6.3), $x \in [0,\ 8]$.

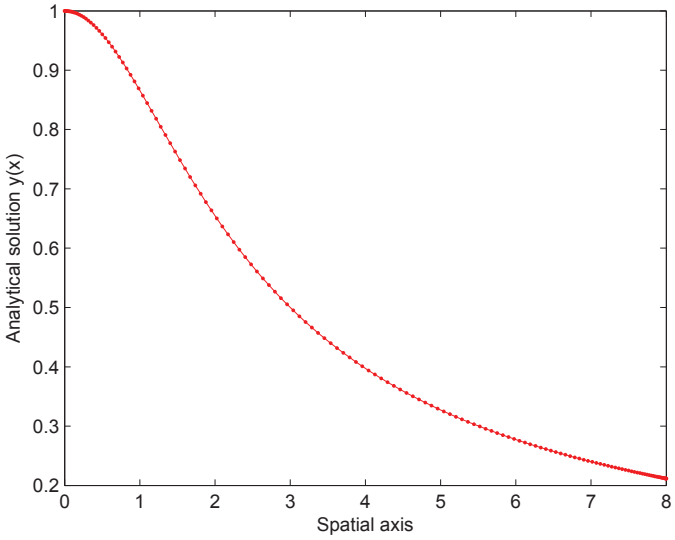

FIGURE 6.14: Analytical solution of Lene-Emden equation (6.3), $x \in [0,\ 8]$.

Iterative methods		MZ	HM
Number of iterations		2	3
Size of problem		1600	1600
Number of steps		4	5
Theoretical convergence-order(CO)		11	10
Number of function evaluations per iteration		4	4
Solutions of system of linear equations per iteration		4	4
Number of Jacobian evaluations per iteration		2	2
Number of Jacobian LU-factorizations per iteration		1	1
Number of matrix vector multiplication per iteration		6	5
	Iter		
$\|\|F(x)\|\|_2$	1	0.47	2.19
	2	$7.86e-14$	$3.50e-11$
	3		$1.02e-13$
Execution time		0.14	0.19

TABLE 6.9: Comparison of performances for different multi-step methods in the case of the Burgers equation (6.4), initial guess $u(x_i,t_j)=0$, $u(x,0)=\frac{2\gamma\pi sin(\pi x)}{\alpha+\beta cos(\pi x)}$, $u(0,t)=u(2,t)=0$, $\alpha=15$, $\beta=14$, $\gamma=0.2$, $n_x=40$, $n_t=40$, $x\in[0,\ 2]$, $t\in[0,\ 100]$.

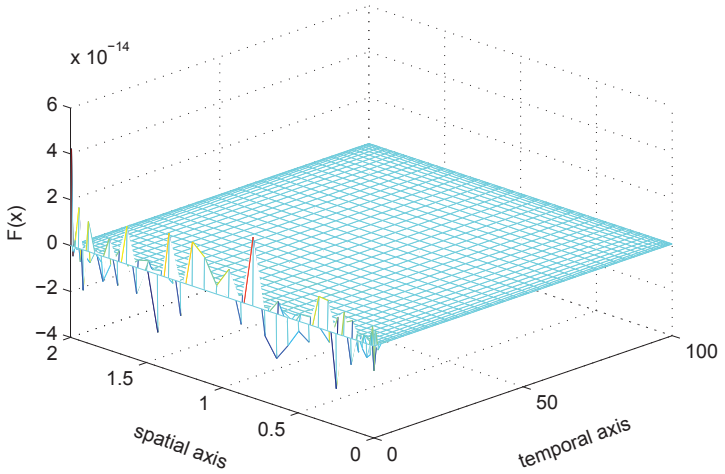

FIGURE 6.15: Comparison of performances for different multi-step methods in the case of the Burgers equation (6.4), initial guess $u(x_i,t_j)=0$, $u(x,0)=\frac{2\gamma\pi sin(\pi x)}{\alpha+\beta cos(\pi x)}$, $u(0,t)=u(2,t)=0$, $\alpha=15$, $\beta=14$, $\gamma=0.2$, $n_x=40$, $n_t=40$, $x\in[0,\ 2]$, $t\in[0,\ 100]$

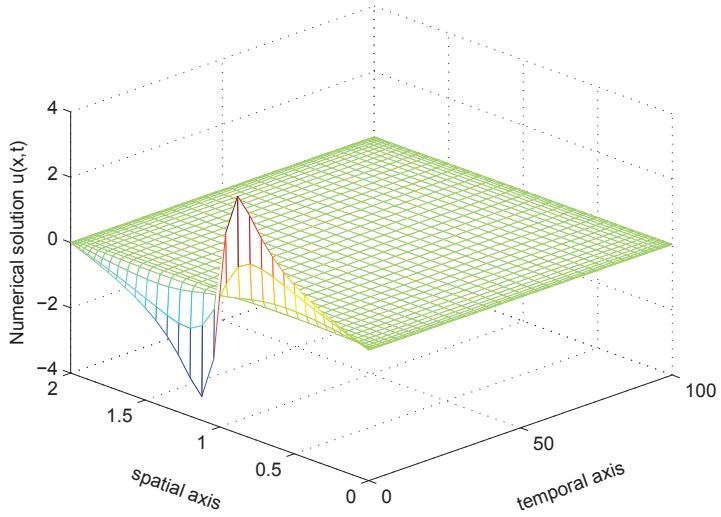

FIGURE 6.16: Comparison of performances for different multi-step methods in the case of the Burgers equation (6.4), initial guess $u(x_i,t_j)=0$, $u(x,0)=\frac{2\gamma\pi sin(\pi x)}{\alpha+\beta cos(\pi x)}$, $u(0,t)=u(2,t)=0$, $\alpha=15$, $\beta=14$, $\gamma=0.2$, $n_x=40$, $n_t=40$, $x\in[0,\ 2]$, $t\in[0,\ 100]$.

Iterative methods		MZ	HM
Number of iterations		1	1
Size of problem		4420	4420
Number of steps		3	4
Theoretical convergence-order(CO)		8	8
Number of function evaluations per iteration		3	3
Solutions of system of linear equations per iteration		3	3
Number of Jacobian evaluations per iteration		2	2
Number of Jacobian LU-factorizations per iteration		1	1
Number of matrix vector multiplication per iteration		4	4
	Steps		
$Max\lvert\boldsymbol{x}_q-\boldsymbol{x}^*\rvert$	1	$11.76e-2$	$4.11e-1$
	2	$3.01e-4$	$2.62e-3$
	3	$7.92e-7$	$2.63e-5$
	4		$4.39e-7$
Execution time		71.69	71.79

TABLE 6.10: Comparison of performances for different multi-step methods in the case of the Klien Gordon equation (6.5), initial guess $u(x_i,t_j)=0$, $u(x,t)=\delta sech(\kappa(x-\nu t)$, $\kappa=\sqrt{\frac{k}{c^2-\nu^2}}$, $\delta=\sqrt{\frac{2k}{\gamma}}$, $c=1$, $\gamma=1$, $\nu=0.5$, $k=0.5$, $n_x=170$, $n_t=26$, $x\in[-22,\ 22]$, $t\in[0,\ 0.5]$.

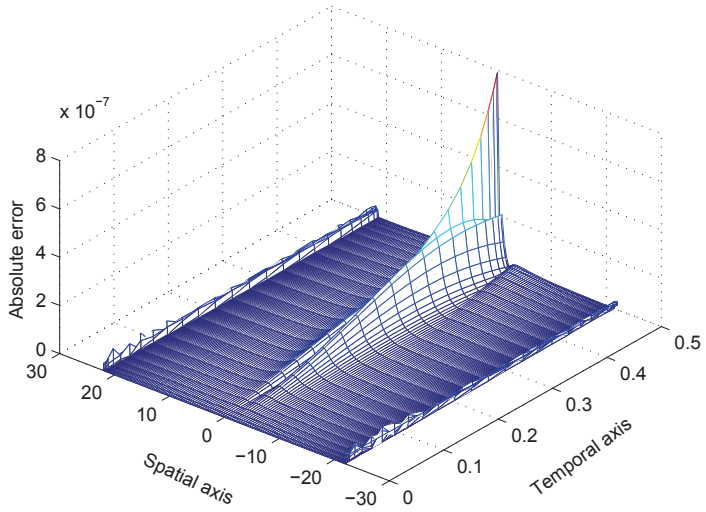

FIGURE 6.17: Absolute error plot for multi-step method MZ in the case of the Klien Gordon equation (6.5), initial guess $u(x_i,t_j) = 0$, $u(x,t) = \delta sech(\kappa(x - \nu t)$, $\kappa = \sqrt{\frac{k}{c^2-\nu^2}}$, $\delta = \sqrt{\frac{2k}{\gamma}}$, $c = 1$, $\gamma = 1$, $\nu = 0.5$, $k = 0.5$, $n_x = 170$, $n_t = 26$, $x \in [-22,\ 22]$, $t \in [0,\ 0.5]$.

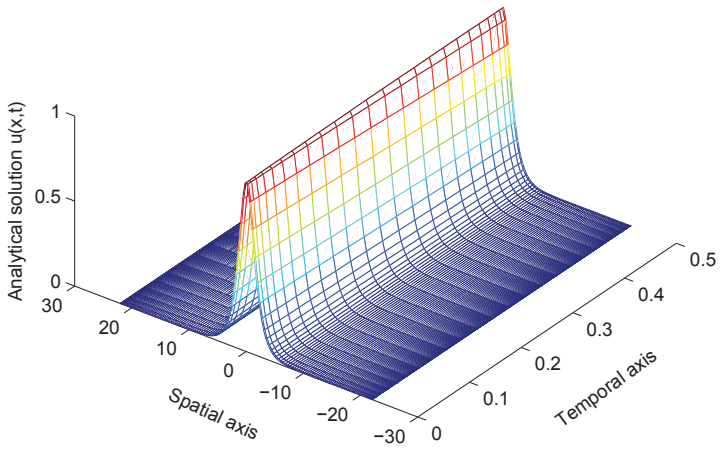

FIGURE 6.18: Analytical solution of the Klien Gordon equation (6.5), $x \in [-22,\ 22]$, $t \in [0,\ 0.5]$.

Iterative methods		MZ	HM
Number of iterations		3	3
Size of problem		4	4
Number of steps		7	8
Theoretical convergence-order(CO)		20	16
Computational convergence-order(COC)		20.1	16.1
Number of function evaluations per iteration		7	7
Solutions of system of linear equations per iteration		7	7
Number of Jacobian evaluations per iteration		2	2
Number of Jacobian LU-factorizations per iteration		1	1
Number of matrix vector multiplication per iteration		12	8
	Iter		
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$	1	$8.51e-142$	$7.13e-1$
	2	$3.56e-277$	$5.61e-190$
	3	$3.32e-5569$	$1.93e-3057$
Execution time		0.02	0.03

TABLE 6.11: Comparison of performances for different multi-step methods in the case of general systems of nonlinear equations (6.39), the initial guess for both of methods is $[0.5,\ 0.5,\ 0.5,\ -0.2]$.

Iterative methods		MZ	HM
Number of iterations		1	1
Size of problem		4	4
Number of steps		30	31
Theoretical convergence-order(CO)		89	62
Number of function evaluations per iteration		30	30
Solutions of system of linear equations per iteration		30	30
Number of Jacobian evaluations per iteration		2	2
Number of Jacobian LU-factorizations per iteration		1	1
Number of matrix vector multiplication per iteration		58	30
	Steps		
$Max\|\boldsymbol{x}_q - \boldsymbol{x}^*\|$	1	0.01	0.02
	2	$1.91e-4$	$6.77e-4$
	3	$3.05e-6$	$3.67e-5$
	5	$5.603-10$	$8.43e-8$
	10	$1.363-19$	$1.26e-14$
	15	$2.463-29$	$1.44e-21$
	20	$3.93e-39$	$1.46e-28$
	25	$5.88e-49$	$1.40e-35$
	30	$8.44e-59$	$1.29e-42$
	31		$5.01e-44$
Execution time		0.02	0.02

TABLE 6.12: Comparison of performances for different multi-step methods in the case of general systems of nonlinear equations (6.39), the initial guess for both of methods is $[0.5,\ 0.5,\ 0.5,\ -0.2]$.

It is worth mentioning that when we solve 1-D problem, we always use high prcesion floating point arithmetic but for 2-D problems we use double precision. In 2-D problems, It is hard to achieve accuracy higher than double.

6.6 Summary

The main benefit of multi-steps methods is to provide computationally efficient iterative methods, which use a minimal number of Jacobian evaluations and related inversions. The single iteration of a good multi-step iterative methods uses only one inversion (in terms of LU-factorization) of the Jacobian, which is computationally efficient: the multi-step idea is employed for enhancing the convergence order. The key fact related to better performances of the MZ method is hidden in the increment of convergence-order by three, for each step, and obeys the formula $3s-1$ where s is the step number.

Three ODEs, two PDEs and one small general system of nonlinear equations are presented and numerical results confirm the convergence-order, accuracy and validity of MZ multi-step iterative method. We also found that spectral collocation methods offer a good accuracy, which helped us to verify our claimed convergence-orders, when using our MZ multi-step iterative method.

Chapter 7

An Efficient Matrix Iteration for Computing Weighted Moore-Penrose Inverse

The goal of this study is threefold. In order to calculate the weighted Moore-Penrose inverse, we first derive a new matrix iteration for computing the inverse of non-singular square matrices. We then analytically extend the obtained results in order to compute the Moore-Penrose generalized inverse of a non-square matrix. Subsequently, these results will again be theoretically extended to find the weighted Moore-Penrose inverse. The computational efficiency of the presented scheme is rigourously studied and compared with the existing matrix iterations, to show its computational efficiency. Some applications are given as well.

7.1 Introduction

The introduction and importance of weighted Moore-Penrose inverse for an arbitrary matrix has been made in [196], [197], and [198]. For an arbitrary matrix $A \in \mathbb{C}^{m\times n}$, and two Hermitian positive definite matrices M and N of orders m and n, respectively, there is a unique matrix X satisfying the relations

$$I)\ AXA = A,\ \ II)\ XAX = X,\ \ III)\ (MAX)^* = MAX,\ \ IV)\ (NXA)^* = NXA. \qquad (7.1)$$

The matrix $X \in \mathbb{C}^{n\times m}$ is known as the weighted Moore-Penrose inverse of A, denoted by $A^{\dagger}_{MN}$ and the relation (7.1) is called as weighted Penrose equations. In particular, when $M = I_{m\times m}$ and $N = I_{n\times n}$, the matrix X is called the Moore-Penrose inverse or the generalized pseudo-inverse and is denoted by $A^{\dagger}$, while (7.1) reduced to the well-known Penrose equations originally attributed to [199] in what follows

$$i)\ AXA = A,\ \ ii)\ XAX = X,\ \ iii)\ (AX)^* = AX,\ \ iv)\ (XA)^* = XA. \tag{7.2}$$

Algorithms for computing the (weighted) Moore-Penrose inverse of a matrix are a subject of current research (see, e.g., [200], [201] and [85]). Greville's partitioning method for numerical computation of generalized inverses was introduced in [202]. Wang in [203] generalized Greville's method to the weighted Moore-Penrose inverse. Many numerical algorithms for computing the (weighted) Moore-Penrose inverse lack of numerical stability. The Greville's algorithm requires more operations and consequently it accumulates more rounding errors. Furthermore, it is widely known that the Moore-Penrose inverse is not necessarily a continuous function of the elements of the matrix. The existence of this discontinuity provides more efforts in its computation [196].

It is therefore clear that cumulative round-off errors should be totally eliminated, which is possible only by means of the symbolic implementation. In this case, variables are stored in the "exact" form or can be left "unassigned", resulting in no loss of accuracy during the calculation. Anyway, by increasing the dimension of the input matrix, the computation of its (weighted) Moore-Penrose inverse by the symbolic implementation will take too much time, This made some numerical analysts to suggest and rely on numerically stable matrix methods.

The fundamental method for finding the weighted Moore-Penrose inverse is based on the weighted singular value decomposition (WSVD) discussed originally in [204] given in what follows. Assume that $A \in \mathbb{C}^{m\times n}$ and $rank(A) = r$. There exist $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$, satisfying $U^*MU = I_{m\times m}$ and $V^*N^{-1}V = I_{n\times n}$, such that

$$A = U \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} V^*. \tag{7.3}$$

Then, the weighted Moore-Penrose inverse $A^{\dagger}_{MN}$ could be expressed by

$$A^{\dagger}_{MN} = N^{-1}V \begin{pmatrix} D^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^{*}M, \tag{7.4}$$

where $D = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, and σ_i^2 is the nonzero eigenvalue of $N^{-1}A^{*}MA$. Furthermore,

$$\|A\|_{MN} = \sigma_1, \quad \|A^{\dagger}_{MN}\|_{NM} = \frac{1}{\sigma_r}. \tag{7.5}$$

We denote throughout this chapter $A^{\#} = N^{-1}A^{*}M$ as the weighted conjugate transpose matrix of A.

The restrictions of computing the weighted Moore-Penrose inverse using the WSVD encouraged some researchers to develop iteration methods for this purpose. In 2006, Huang and Zhang in [205] developed the quadratically Schulz iterative method [71] (sometimes called as Hotelling inverse-finder [206]) as follows

$$X_{k+1} = X_k(2I - AX_k),\ k = 0, 1, 2, \cdots, \tag{7.6}$$

for finding the weighted Moore-Penrose inverse.

This scheme has interesting features of being based exclusively on matrix-matrix operations, which is quite fast in parallel machines. The Schulz iteration has poly-logarithmic complexity and is numerically stable [78]. Note that though (7.6) was first suggested to find the inverse of square matrices, it had successfully discussed in [207] that it is also so interesting in finding the Moore-Penrose inverse.

The idea of matrix iterations (Schulz-type iterative methods) was then developed by Sen and Prabhu in [208] to present matrix iterations of arbitrary orders of convergence for finding the pseudo-inverse.

The perception and reason of constructing higher order matrix iterations is that the low order ones, such as (7.6) are too slow at the beginning of the process and it might take many cycles to arrive at the convergence phase. That is, the scheme (7.6) is even linearly convergent at the beginning of the process. Söderström and Stewart in [78] indicated that (7.6) requires *almost* the following number of iterations in machine precision to converge

$$s \approx 2\log_2 \kappa_2(A), \tag{7.7}$$

where $\kappa_2(A)$, is the condition number of the input matrix A in $\|\cdot\|_2$.

This made the construction of higher order matrix iterations (only the efficient ones) meaningful as will be discussed further in Section 7.3. For example, the cubically Chebyshev's method [209] could be presented by

$$X_{k+1} = X_k(3I - AX_k(3I - AX_k)),\ k = 0,1,2,\cdots, \tag{7.8}$$

while a high-order method [209] to reach the convergence order nine can be deduced as

$$X_{k+1} = X_k(I + Y_k(I + Y_k(I + Y_k(I + Y_k(I + Y_k(I + Y_k(I + Y_k(I + Y_k))))))),\ k = 0,1,2,\cdots, \tag{7.9}$$

where $Y_k = I - AX_k$.

It must be noted that in such constructions and as also discussed in [210], the convergence order p could be attained using p times of matrix-matrix products.

Generally speaking, the construction of different iterative methods for matrix inversion is based on applying a nonlinear equation solver (see e.g. [211]) on the matrix equation

$$AX = I, \tag{7.10}$$

wherein I is the identity matrix of the appropriate dimension.

In this chapter, we seek for a new matrix iteration at which, we obtain a high convergence order p with less number of matrix-by-matrix products than p. This would make the method quite efficient in contrast to the existing iterative methods of the same type for this purpose. Toward this aim, we first in Section 7.2, derive a new method and discuss under which condition, it could converge for non-singular square matrices. Then, in Section 7.3 we infer that the new method is computationally economic. Next, Section 7.4 contains the proof of converegence Moore-Penrose inverse. The main contribution will then be presented by extending the novel method for finding the weighted Moore-Penrose inverse in Section 7.5. Section 7.6 is devoted to the application of the new method to some numerical tests. The chapter ends in Section 7.7, wherein a conclusion will be drawn.

7.2 Derivation

In this section, we contribute and construct a new iterative method for matrix inversion. In fact, we construct a high order method whereas the number of matrix-matrix products is lower than that of the corresponding method (here the scheme (7.9)) produced by the general ways of [210] and [209]. To this purpose, we apply the following new rational iteration function

$$\begin{cases} y_k = x_k - \frac{f(x_k)}{2f'(x_k)}\left(2+\frac{f''(x_k)f(x_k)}{f'(x_k)^2}\right), \\ z_k = y_k - \frac{f(y_k)}{f'(y_k)}, \\ x_{k+1} = y_k - \left(\frac{f(y_k)-\frac{1}{5}f(z_k)}{f'(z_k)}\right)\left(1-\frac{4}{5}\frac{f(z_k)}{f(y_k)}\right), \quad k=0,1,2,\cdots, \end{cases} \tag{7.11}$$

on equation $f(x)=0$. The iterative method (7.11) solves nonlinear equation of single variable and it has local ninth order of convergence for finding the simple zeros of nonlinear equation. In fact, it reads the following error equation: $e_{n+1} = -\frac{1}{25}\left(4c_2^2-25c_3\right)\left(2c_2^2-c_3\right)^3 e_n^9 + O(e_n^{10})$, wherein $c_j = \frac{1}{j!}\frac{f^{(j)}(\alpha)}{f'(\alpha)}$, $j \geq 2$ and α is the simple zero of a nonlinear equation.

The equivalent version of iterative method (7.11) for (7.10) could be obtained

$$\begin{aligned} X_{k+1} &= -\tfrac{1}{25}X_k(3I+AX_k(-3I+AX_k))(-79I+AX_k(3I+AX_k(-3I+AX_k))(87I \\ &\quad +AX_k(3I+AX_k(-3I+AX_k))(-37I+4AX_k(3I+AX_k(-3I+AX_k))))) \\ &= \tfrac{1}{25}X_k(237I-1020AX_k+2644(AX_k)^2-4626(AX_k)^3+5814(AX_k)^4 \\ &\quad -5460(AX_k)^5+3924(AX_k)^6-2169(AX_k)^7+901(AX_k)^8-264(AX_k)^9 \\ &\quad +48(AX_k)^{10}-4(AX_k)^{11}),\ k=0,1,2,\cdots. \end{aligned}$$

We now re-write the obtained iteration as efficiently as possible to reduce the number of matrix-matrix multiplications in what follows

$$\begin{cases} \psi_k = AX_k, \\ \zeta_k = 3I+\psi_k(-3I+\psi_k), \\ \upsilon_k = \psi_k\zeta_k, \\ X_{k+1} = -\frac{1}{25}X_k\zeta_k(-79I+\upsilon_k(87I+\upsilon_k(-37I+4\upsilon_k))),\ k=0,1,2,\cdots. \end{cases} \tag{7.12}$$

The iterative method (7.12) falls within the domain of Schulz-type methods for matrix inversion. It requires an initial matrix to start the process and can rapidly converge,

which is an advantage over the existing methods. Below, we first give a mathematical analysis to observe that under what condition, (7.12) converges.

Theorem 7.1. *Let $A = [a_{i,j}]_{n\times n}$ be a nonsingular complex square matrix. If the initial approximation X_0 satisfies*

$$\|I - AX_0\| < 1, \tag{7.13}$$

then, the iterative method (7.12) converges with ninth order to A^{-1}.

Proof. We assume that (7.13) is true and the matrix $E_0 = I - AX_0$, is the initial residual matrix. Also let that $E_k = I - AX_k = I - \psi_k$. We obtain

$$\begin{aligned}
E_{k+1} &= I - AX_{k+1} \\
&= I - A(-\tfrac{1}{25}X_k(3I + \psi_k(-3I + \psi_k))(-79I + \psi_k(3I + \psi_k(-3I + \psi_k))(87I \\
&\quad + \psi_k(3I + \psi_k(-3I + \psi_k))(-37I + 4\psi_k(3I + \psi_k(-3I + \psi_k))))) \\
&= I - \tfrac{1}{25}[237\psi_k - 1020\psi_k^2 + 2644\psi_k^3 - 4626\psi_k^4 + 5814\psi_k^5 - 5460\psi_k^6 \\
&\quad + 3924\psi_k^7 - 2169\psi_k^8 + 901\psi_k^9 - 264\psi_k^{10} + 48\psi_k^{11} - 4\psi_k^{12}] \\
&= \tfrac{1}{25}(-I + \psi_k)^9(-25I + 4\psi_k(3I + \psi_k(-3I + \psi_k))) \\
&= -\tfrac{1}{25}(I - \psi_k)^9(-21I - 4I + 12\psi_k - 12\psi_k^2 + 4\psi_k^3) \\
&= \tfrac{1}{25}(I - \psi_k)^9(21I + 4(I - \psi_k)^3) \\
&= \tfrac{1}{25}E_k^9(21I + 4E_k^3).
\end{aligned}$$

Subsequently, one has

$$E_{k+1} = \frac{1}{25}[21E_k^9 + 4E_k^{12}]. \tag{7.14}$$

Taking a generic matrix operator norm from both sides of (7.14), we obtain

$$\|E_{k+1}\| \le \frac{1}{25}[21\|E_k\|^9 + 4\|E_k\|^{12}]. \tag{7.15}$$

In addition, since $\|E_0\| < 1$ (due to (7.13)), by relation (7.15) and using mathematical induction, we attain

$$\|E_1\| \le \frac{1}{25}[21\|E_0\|^9 + 4\|E_0\|^{12}] < 1. \tag{7.16}$$

Now, if we take into consideration $\|E_k\| < 1$, then

$$\|E_{k+1}\| \le \frac{1}{25}[21\|E_k\|^9 + 4\|E_k\|^{12}] \le \|E_k\|^9. \tag{7.17}$$

Besides, we get that

$$\|E_{k+1}\| \le \|E_k\|^9 \le \cdots \le \|E_0\|^{9^{k+1}} < 1. \tag{7.18}$$

That is $I - AX_k \to 0$, when $k \to \infty$, and thus $X_k \to A^{-1}$, as $k \to \infty$. Now, one part of the proof is remained. We must manifest that the ninth order of convergence can be obtained for the sequence $\{X_k\}_{k=0}^{k=\infty}$. To this end, we take into consideration that

$$\varepsilon_k = A^{-1} - X_k, \tag{7.19}$$

is the error matrix in the iterative procedure (7.12). Moreover, $A\varepsilon_k = I - AX_k = E_k$. This implies

$$A\varepsilon_{k+1} = E_{k+1} = \frac{1}{25}[21E_k^9 + 4E_k^{12}] = \frac{1}{25}[21(A\varepsilon_k)^9 + 4(A\varepsilon_k)^{12}]. \tag{7.20}$$

One has now that

$$\varepsilon_{k+1} = \frac{1}{25}[21\varepsilon_k(A\varepsilon_k)^8 + 4\varepsilon_k(A\varepsilon_k)^{11}]. \tag{7.21}$$

By taking a generic matrix operator norm, we obtain

$$\|\varepsilon_{k+1}\| \leq (\frac{1}{25}[21\|A\|^8 + 4\|A\|^{11}\|\varepsilon_k\|^3])\|\varepsilon_k\|^9. \tag{7.22}$$

That is, by considering $\xi_k = (\frac{1}{25}[21\|A\|^8 + 4\|A\|^{11}\|\varepsilon_k\|^3])$, the error inequality, which reveals at least local ninth convergence order is

$$\|\varepsilon_{k+1}\| \leq \xi_k\|\varepsilon_k\|^9. \tag{7.23}$$

The proof is finished. □

In the above analysis, we have investigated that the new scheme (7.12) is convergent for nonsingular square complex matrices provided that a good initial approximation is available. Hence, it is of great importance to achieve the convergence by a valid initial value X_0.

Pan and Schreiber in [207] considered that the approximations X_k share singular vectors with A^*, and both the largest (σ_{max}) and the smallest singular values (σ_{min}) of A are available, then for a general matrix A, one can choose

$$X_0 = \frac{2}{\sigma_{min}^2 + \sigma_{max}^2}A^*. \tag{7.24}$$

This is reported as the best general initial choice for X_0, which is also called as the *optimal initial choice*.

Another interesting initial matrix was introduced and developed by Ben-Israel and Greville in [196] as follows

$$X_0 = \alpha A^*, \tag{7.25}$$

where $0 < \alpha < \frac{2}{||A||_2^2}$. Because of the fact that, the computation of matrix norm $||\cdot||_2$, is difficult for large matrices, an alternative bound for α could be considered in what follows

$$0 < \alpha < \frac{2}{\sigma_{max}^2}, \tag{7.26}$$

where σ_{max} can be computed by the Arnoldi algorithm. We refer the reader to [212] for observing more interesting initial choices.

7.3 Efficiency challenge

The performance of an algorithm depends on many aspects. In matrix iterations for finding the inverses, some important aspects are in focus to rate the efficiency of an algorithm by considering the fact that the Schulz-type methods are asymptotically stable. These factors are the local convergence order, number of matrix by matrix multiplications, the stopping criterion, etc. Here, we try to answer that: "is the computational complexity of the new inverse-finder (7.12) reasonable?"

As discussed in Section 7.1, the most general ways for producing higher-order inverse-finders construct the matrix iterations of local convergence order p using p times of matrix-matrix products, while the method (7.12) possesses ninth order of convergence using only seven matrix-matrix multiplications. Traub in the Appendix C of [3] proposed an index, named as the computational efficiency index, by considering all the imposing costs of an algorithm as follows:

$$CEI = p^{\frac{1}{\mathscr{C}}}, \tag{7.27}$$

whereas $\mathscr{C}$ stands for the total computational cost of an algorithm.

It is clear that the governing cost per cycle of each Schulz-type method is the matrix-matrix products. Let us assume that this cost is unity. On the other hand, and similar to (7.7) in the same environment, an iteration method of order p will *almost* require the following number of iterations to converge [84]

$$s \approx 2\log_p \kappa_2(A). \tag{7.28}$$

Therefore, the computational efficiency index of a p-th order matrix iteration with η times of matrix-matrix products per cycle, is

$$CEI \approx p^{\frac{1}{\eta(2\log_p \kappa_2(A))}}. \tag{7.29}$$

The index tries to make a balance between some important factors of an iteration process to give an output indicating the efficiency of an algorithm. Using (7.29) the iterative methods (7.6), (7.8), (7.9) and (7.12), which are denoted throughout this work by "Schulz", "Chebyshev", "KMS" and "PM", respectively, are compared in Figure 7.1 in terms of the computational efficiency index. Figure 7.1, reveals that by increasing the condition number and under the same conditions, the new matrix iteration is more economic than the other competitors in the literature, since the convergence order 9 is attainable using 7 matrix-matrix products. Figure 7.2 also reveals the estimate number of iterates by increasing the condition number for different methods.

Up to now, we have derived a new iteration for matrix inversion when the complex matrices are square and nonsingular, and we have analytically found that it is economic in terms of the computational efficiency index to tackle matrix inversion problems. By considering this as the first contribution of this work, we extend the obtained results for finding the Moore-Penrose and the weighted Moore-Penrose inverses in the forthcoming Sections 7.4 and 7.5, respectively.

7.4 Moore-Penrose inverse

Let us now extend the contributed method (7.12) for calculating the generalized pseudo-inverse $A^{\dagger}$. That is, we must analytically reveal that the sequence $\{X_k\}_{k=0}^{k=\infty}$ generated by the iterative Schulz-type method (7.12), for any $k \geq 0$, tends to the Moore-Penrose inverse as well.

Using mathematical induction, it would be easy to check that the iterates produced at each cycle of (7.12) satisfies the following relation:

$$(AX_k)^* = AX_k, \;\; (X_kA)^* = X_kA, \;\; X_kAA^{\dagger} = X_k, \;\; A^{\dagger}AX_k = X_k. \tag{7.30}$$

The forthcoming theorem best addresses convergence conditions for the high-order iteration (7.12), when dealing with Moore-Penrose inverse.

Theorem 7.2. *For the complex matrix $A \in \mathbb{C}^{m\times n}$, and the sequence $\{X_k\}_{k=0}^{k=\infty}$ generated by (7.12), for any $k \geq 0$, using the initial approximation (7.25), the sequence is converged to the pseudo-inverse $A^{\dagger}$ with at least ninth order of convergence.*

Proof. Considering $\mathbb{E}_k = X_k - A^{\dagger}$, as the error matrix for finding the Moore-Penrose inverse. We have

$$\begin{aligned} A\mathbb{E}_{k+1} &= AX_{k+1} - AA^{\dagger} \\ &= AX_{k+1} - I + I - AA^{\dagger} \\ &= -E_{k+1} + I - AA^{\dagger} \\ &= -\tfrac{1}{25}[21E_k^9 + 4E_k^{12}] + I - AA^{\dagger} \\ &= \tfrac{1}{25}(21(A\mathbb{E}_k)^9 - 4(A\mathbb{E}_k)^{12}), \end{aligned} \tag{7.31}$$

wherein the following identities have been used

$$(I - AA^{\dagger})^t = (I - AA^{\dagger}), \quad t \in \mathbb{N}, \tag{7.32}$$

and

$$(I - AA^{\dagger})A\mathbb{E} = 0. \tag{7.33}$$

Clearly, we have $\|A\mathbb{E}_{k+1}\| \leq \frac{1}{25}(21\|A\mathbb{E}_k\|^9 + 4\|A\mathbb{E}_k\|^{12})$. Let us now denote $P = AA^{\dagger}$, and $S = I - AX_0$. Then, $P^2 = P$ and

$$\begin{aligned} PS &= AA^{\dagger}(I - AX_0) \\ &= AA^{\dagger} - AA^{\dagger}AX_0 \\ &= AA^{\dagger} - AX_0 \\ &= AA^{\dagger} - AX_0AA^{\dagger} \\ &= (I - AX_0)AA^{\dagger} \\ &= SP. \end{aligned} \tag{7.34}$$

On the other hand, Stanimirović and Cvetković-Ilić in [213] showed that for $P \in \mathbb{C}^{n\times n}$ and $S \in \mathbb{C}^{n\times n}$ such that $P = P^2$ and $PS = SP$, one has

$$\rho(PS) \leq \rho(S). \tag{7.35}$$

Consequently, using (7.34) and (7.35), we attain

$$\rho(A(X_0 - A^{\dagger})) = \rho\left(A\left(\alpha A^* - A^{\dagger}\right)\right) \leq \rho\left(I - \alpha AA^*\right) = \max_{1\leq i\leq r} |1 - \lambda_i(\alpha AA^*)|, \tag{7.36}$$

wherein r denotes the number of singular values. Now, by using an appropriate value for α as given in (7.26) (e.g. $\alpha = \frac{1}{\sigma_{max}^2}$), we conclude that

$$\max_{1\leq i\leq r} |1-\lambda_i(\alpha AA^*)| < 1. \tag{7.37}$$

It is also known that there exists a positive constant ε and a matrix norm $\|\cdot\|$, such that

$$\|A(X_0 - A^\dagger)\| \leq \rho\left(A(X_0 - A^\dagger)\right) + \varepsilon < 1. \tag{7.38}$$

(7.38) implies that $\|A\mathbb{E}_0\| < 1$, and by a similar reasoning as in (16)-(19), one may obtain

$$\|A\mathbb{E}_{k+1}\| \leq \frac{1}{25}[21\|A\mathbb{E}_k\|^9 + 4\|A\mathbb{E}_k\|^{12}] \leq \|A\mathbb{E}_k\|^9 \leq \|A\|^9\|\mathbb{E}_k\|^9. \tag{7.39}$$

We now find the error inequality of the new scheme (7.12), when finding the Moore-Penrose inverse, as follows:

$$\|X_{k+1} - A^\dagger\| = \|A^\dagger AX_{k+1} - A^\dagger AA^\dagger\| \leq \|A^\dagger\|\|AX_{k+1} - AA^\dagger\| = \|A^\dagger\|\|A\mathbb{E}_{k+1}\|. \tag{7.40}$$

And subsequently using (7.39) and (7.40), we have

$$\|\mathbb{E}_{k+1}\| \leq \|A^\dagger\|\|A\|^9\|\mathbb{E}_k\|^9. \tag{7.41}$$

Thus, $\|X_k - A^\dagger\| \to 0$, i.e. the obtained sequence of (7.12) converges to the Moore-Penrose inverse as $k \to +\infty$. This ends the proof. □

Note that we used the proof of the square nonsingular case in the proof of the Moore-Penrose inverse. Hence, it would be more appropriate to do the generalization step by step instead of expressing the most general case and then deducing the simple cases.

It must be remarked that the order of convergence and the matrix-matrix products are not the only factors to govern the efficiency of an algorithm in matrix iterations. Generally speaking, the stopping criterion (or in other words the number of full steps) could be reported as one of the important factors, which could indirectly affect the computational time of an algorithm in implementations, specially when trying to find the (weighted) Moore-Penrose inverse.

To be more precise, in the floating point arithmetic, the following stopping criterion might be used

$$\max\{\|AX_kA-A\|_\circ, \|X_kAX_k-X_k\|_\circ, \|(AX_k)^*-AX_k\|_\circ, \|(X_kA)^*-X_kA\|_\circ\} \leq \varepsilon, \quad (7.42)$$

where $\|\cdot\|_\circ$, denotes the appropriate norm of a matrix. This is a safe strategy to tackle Moore-Penrose inverse numerically, because it guarantees that the prescribed tolerance (ε) of the user has been achieved.

This shows the importance of reduction in the number of iterations as well. Since, the computation of (7.42) is too much burdensome due to further matrix multiplications and matrix norms per computing steps, which makes the method of lower orders with large number of iterations, to be not economic in terms of the computational time, while the higher order methods such as (7.12) would be better, since fewer number of iterations must be computed to achieve the prescribed tolerance, and consequently fewer matrix-matrix products.

Note that an alternative remedy could be the following stop termination, which is somewhat unsafe, though it significantly has lower burden than that of (7.42):

$$||X_{k+1}-X_k||_\circ \leq \varepsilon. \quad (7.43)$$

7.5 Weighted Moore-Penrose inverse

This section contains the third contribution of the present study by showing that how the new method (7.12) and under what conditions it could be applied for finding the weighted Moore-Penrose inverse $A^\dagger_{MN}$.

The most important change when applying the new iterative method (7.12) for the weighted Moore-Penrose inverse is related to the the initial matrix. Here, the initial matrix X_0 plays a very crucial significance to provide convergence, since it must be chosen as if the convergence to the weighted Moore-Penrose inverse happens. Accordingly, we must apply the following initial matrix

$$X_0 = \beta A^\#, \quad (7.44)$$

where $A^{\#} = N^{-1}A^{*}M$ is the weighted conjugate transpose matrix of A and

$$\beta = \frac{1}{\sigma_1^2}. \tag{7.45}$$

The reason of selection β in this way will be proven in Theorem 7.5. Here, σ_1 stands for the largest eigenvalue of the matrix $N^{-1}A^{*}MA$.

In order to validate the applicability of the new scheme for the weighted Moore-Penrose inverse, we now first show that how the iterates produced by (7.12) satisfy some certain equations and then show a relation between (7.12) and (7.4).

Lemma 7.3. *For the sequence $\{X_k\}_{k=0}^{k=\infty}$ generated by (7.12) with the initial matrix (7.44), for any $k \geq 0$, it holds that*

$$(MAX_k)^{*} = MAX_k, \ \ (NX_kA)^{*} = NX_kA, \ \ X_kAA^{\dagger}_{MN} = X_k, \ \ A^{\dagger}_{MN}AX_k = X_k. \tag{7.46}$$

Proof. We will prove the conclusion by induction on k. For $k = 0$ and X_0, as in (7.44), the first two equations can be verified easily, and we only give a verification to the last two equations using the facts that $(AA^{\dagger}_{MN})^{\#} = AA^{\dagger}_{MN}$ and $(A^{\dagger}_{MN}A)^{\#} = A^{\dagger}_{MN}A$, in what follows

$$X_0AA^{\dagger}_{MN} = \beta A^{\#}AA^{\dagger}_{MN} = \beta A^{\#}(AA^{\dagger}_{MN})^{\#} = \beta A^{\#}(A^{\dagger}_{MN})^{\#}A^{\#} = \beta (AA^{\dagger}_{MN}A)^{\#} = \beta A^{\#} = X_0, \tag{7.47}$$

$$A^{\dagger}_{MN}AX_0 = \beta A^{\dagger}_{MN}AA^{\#} = \beta (A^{\dagger}_{MN}A)^{\#}A^{\#} = \beta (A^{\#}(A^{\dagger}_{MN})^{\#}A^{\#}) = \beta (A(A^{\dagger}_{MN}A)^{\#} = \beta A^{\#} = X_0. \tag{7.48}$$

Assume now that the conclusion holds for some $k > 0$. We now show that it continues to hold for $k+1$. Using the iterative method (7.12), one has

$$\begin{aligned}
(MAX_{k+1})^* &= (MA(-\tfrac{1}{25}X_k(3I+\psi_k(-3I+\psi_k))(-79I+\psi_k(3I+\psi_k(-3I+\psi_k))(87I \\
&\quad +\psi_k(3I+\psi_k(-3I+\psi_k))(-37I+4\psi_k(3I+\psi_k(-3I+\psi_k))))))^* \\
&= \tfrac{1}{25}[237(M\psi_k)^* - 1020(M\psi_k^2)^* + 2644(M\psi_k^3)^* - 4626(M\psi_k^4)^* + 5814(M\psi_k^5)^* \\
&\quad -5460(M\psi_k^6)^* + 3924(M\psi_k^7)^* - 2169(M\psi_k^8)^* + 901(M\psi_k^9)^* - 264(M\psi_k^{10})^* \\
&\quad +48(M\psi_k^{11})^* - 4(M\psi_k^{12})^*] \\
&= \tfrac{1}{25}[237M\psi_k - 1020\psi_k^{*1}\psi_k^*M^* + 2644\psi_k^{*2}\psi_k^*M^* - 4626\psi_k^{*3}\psi_k^*M^* \\
&\quad +5814\psi_k^{*4}\psi_k^*M^* - 5460\psi_k^{*5}\psi_k^*M^* + 3924\psi_k^{*6}\psi_k^*M^* - 2169\psi_k^{*7}\psi_k^*M^* \\
&= \tfrac{1}{25}[237M\psi_k - 1020M\psi_k^2 + 2644M\psi_k^3 - 4626M\psi_k^4 + 5814M\psi_k^5 - 5460M\psi_k^6 \\
&\quad +3924M\psi_k^7 - 2169M\psi_k^8 + 901M\psi_k^9 - 264M\psi_k^{10} + 48M\psi_k^{11} - 4M\psi_k^{12}] \\
&= MA(-\tfrac{1}{25}X_k(3I+\psi_k(-3I+\psi_k))(-79I+\psi_k(3I+\psi_k(-3I+\psi_k))(87I \\
&\quad +\psi_k(3I+\psi_k(-3I+\psi_k))(-37I+4\psi_k(3I+\psi_k(-3I+\psi_k))))) \\
&= MAX_{k+1},
\end{aligned}$$

which uses the fact that $(M\psi_k)^* = M\psi_k$, M is Hermitian positive definite ($M^* = M$), and also e.g. $(M\psi_k^2)^* = (M\psi_k\psi_k)^* = \psi_k^*(M\psi_k)^* = \psi_k^*(M\psi_k) = \psi_k^*M^*\psi_k = (M\psi_k)^*\psi_k = M\psi_k\psi_k = M\psi_k^2$. Thus, the first equality in (7.46) holds for $k+1$, and the second equality can be proved in a similar way. For the third equality in (7.46), using the assumption that $X_kAA^\dagger_{MN} = X_k$ and the iterative method (7.12), we could write down

$$\begin{aligned}
X_{k+1}AA^\dagger_{MN} &= \tfrac{1}{25}X_k(237 - 1020\psi_k + 2644\psi_k^2 - 4626\psi_k^3 + 5814\psi_k^4 - 5460\psi_k^5 \\
&\quad +3924\psi_k^6 - 2169\psi_k^7 + 901\psi_k^8 - 264\psi_k^9 + 48\psi_k^{10} - 4\psi_k^{11})AA^\dagger_{MN} \\
&= \tfrac{1}{25}(237X_kAA^\dagger_{MN} - 1020X_k\psi_kAA^\dagger_{MN} + 2644X_k\psi_k^1\psi_kAA^\dagger_{MN} \\
&\quad -4626X_k\psi_k^2\psi_kAA^\dagger_{MN} + 5814X_k\psi_k^3\psi_kAA^\dagger_{MN} - 5460X_k\psi_k^4\psi_kAA^\dagger_{MN} \\
&\quad +3924X_k\psi_k^5\psi_kAA^\dagger_{MN} - 2169X_k\psi_k^6\psi_kAA^\dagger_{MN} + 901X_k\psi_k^7\psi_kAA^\dagger_{MN} \\
&\quad -264X_k\psi_k^8\psi_kAA^\dagger_{MN} + 48X_k\psi_k^9\psi_kAA^\dagger_{MN} - 4X_k\psi_k^{10}\psi_kAA^\dagger_{MN}) \\
&= \tfrac{1}{25}(237X_k - 1020X_k\psi_k + 2644X_k\psi_k^1\psi_k - 4626X_k\psi_k^2\psi_k \\
&\quad +5814X_k\psi_k^3\psi_k - 5460X_k\psi_k^4\psi_k + 3924X_k\psi_k^5\psi_k - 2169X_k\psi_k^6\psi_k \\
&\quad +901X_k\psi_k^7\psi_k - 264X_k\psi_k^8\psi_k + 48X_k\psi_k^9\psi_k - 4X_k\psi_k^{10}\psi_k) \\
&= \tfrac{1}{25}X_k(237 - 1020\psi_k + 2644\psi_k^2 - 4626\psi_k^3 + 5814\psi_k^4 - 5460\psi_k^5 \\
&\quad +3924\psi_k^6 - 2169\psi_k^7 + 901\psi_k^8 - 264\psi_k^9 + 48\psi_k^{10} - 4\psi_k^{11}) \\
&= X_{k+1}.
\end{aligned}$$

Consequently, the third equality in (7.46) holds for $k+1$. The fourth equality can similarly be proved, and the desired result follows. □

Lemma 7.4. *Considering the condition of Lemma 5.1, it holds that*

$$(V^{-1}N)X_k(M^{-1}(U^*)^{-1}) = \begin{pmatrix} T_k & 0 \\ 0 & 0 \end{pmatrix}, \tag{7.49}$$

where T_k is diagonal, and $U \in \mathbb{C}^{m\times m}$, $V \in \mathbb{C}^{n\times n}$, $U^*MU = I_{m\times m}$, $V^*N^{-1}V = I_{n\times n}$, and $A = U\begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}V^* = U\Sigma V^*$.

Proof. Let $T_0 = \beta D$, where $D = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, and σ_i^2 is the nonzero eigenvalue of $N^{-1}A^*MA$. Hence

$$\begin{aligned} T_{k+1} := \varphi(T_k) = \frac{1}{25}T_k(237I - 1020DT_k + 2644(DT_k)^2 - 4626(DT_k)^3 + 5814(DT_k)^4 - 5460(DT_k)^5 \\ + 3924(DT_k)^6 - 2169(DT_k)^7 + 901(DT_k)^8 - 264(DT_k)^9 + 48(DT_k)^{10} - 4(DT_k)^{11}). \end{aligned} \tag{7.50}$$

We now prove this lemma using mathematical induction. For the initial case, we have

$$\begin{aligned} (V^{-1}N)X_0(M^{-1}(U^*)^{-1}) =& \beta(V^{-1}N)A^{\#}(M^{-1}(U^*)^{-1}) \\ =& \beta(V^{-1}N)N^{-1}A^*(MM^{-1}(U^*)^{-1}) \\ =& \beta(V^{-1}N)N^{-1}V\begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}U^*(MM^{-1}(U^*)^{-1}) \\ =& \begin{pmatrix} \beta D & 0 \\ 0 & 0 \end{pmatrix}. \end{aligned}$$

Moreover, if (7.53) is satisfied, then by (7.12), we have

$$\begin{aligned} (V^{-1}N)X_{k+1}(M^{-1}(U^*)^{-1}) =& -\tfrac{1}{25}[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})](3I + A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})] \\ & (-3I + A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})]))(-79I \\ & + A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})](3I + A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})](-3I \\ & + A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})]))(87I + A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})](3I \\ & + A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})](-3I + A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})]))(-37I \\ & + 4A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})](3I + A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})](-3I \\ & + A[(V^{-1}N)X_k(M^{-1}(U^*)^{-1})])))). \end{aligned}$$

And now, by considering

$$A = U^*MU \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} V^*NV, \tag{7.51}$$

we obtain

$$(V^{-1}N)X_{k+1}(M^{-1}(U^*)^{-1}) = \begin{pmatrix} \varphi(T_k) & 0 \\ 0 & 0 \end{pmatrix}, \tag{7.52}$$

which establishes that (7.50) is diagonal. The proof is ended. □

Theorem 7.5. *Assume that A is an $m \times n$ matrix whose weighted singular value decomposition is given by (7.4). Let furthermore that the initial matrix could be constructed by (7.44). Define the sequence of matrices $X_1, X_2, \ldots$, using (7.12). Then, this sequence of iterates converges to $A^{\dagger}_{MN}$.*

Proof. In view of (7.4), to establish this result, we only now need to verify that

$$\lim_{k\to\infty}(V^{-1}N)X_k(M^{-1}(U^*)^{-1}) = \begin{pmatrix} D^{-1} & 0 \\ 0 & 0 \end{pmatrix}. \tag{7.53}$$

It follows from Lemmas 7.3 and 7.4 that $\{T_k\} = diag(\tau_1^{(k)}, \tau_2^{(k)}, \ldots, \tau_r^{(k)})$, where

$$\tau_i^{(0)} = \beta\sigma_i, \tag{7.54}$$

and

$$\begin{aligned} \tau_i^{(k+1)} \;=\; & -\tfrac{1}{25}\tau_i^{(k)}(3I+\sigma_i\tau_i^{(k)}(-3I+\sigma_i\tau_i^{(k)}))(-79I \\ & +\sigma_i\tau_i^{(k)}(3I+\sigma_i\tau_i^{(k)}(-3I+\sigma_i\tau_i^{(k)}))(87I \\ & +\sigma_i\tau_i^{(k)}(3I+\sigma_i\tau_i^{(k)}(-3I+\sigma_i\tau_i^{(k)}))(-37I \\ & +4\sigma_i\tau_i^{(k)}(3I+\sigma_i\tau_i^{(k)}(-3I+\sigma_i\tau_i^{(k)}))))). \end{aligned} \tag{7.55}$$

Now, the sequence generated by the above formula is the result for applying (7.12) for computing the zero σ_i^{-1} of the function $\phi(\tau) = \sigma_i - \tau^{-1}$, with the initial value $\tau_i^{(0)}$. It is seen that this iteration converge to σ_i^{-1} provided $0 < \tau_i^{(0)} < \frac{2}{\sigma_i}$, which results the condition on β (so the choice in formula (7.45) has been proved). Thus, $\{T_k\} \to \Sigma^{-1}$, and the relation (7.53) is satisfied. Clearly, $\{X_k\}_{k=0}^{k=\infty} \to A^{\dagger}_{MN}$, when $k \to \infty$. The proof is complete. □

7.6 Applications

This section addresses issues related to the numerical precision of the matrix inverse finders, using Mathematica 8 built-in precision, [214]. For numerical comparisons in this section, we have used the methods (7.6) denoted by "Schulz", (7.8) denoted by "Chebyshev", (7.9) denoted by "KSM", and (7.12) denoted by "PM". As the programs were running, we measured the running time using the command `AbsoluteTiming[]` to report the elapsed CPU time (in second) for the experiments. The computer specifications are Microsoft Windows XP Intel(R), Pentium(R) 4, CPU 3.20GHz, with 4GB of RAM.

We present three different types of tests. Test 1 dedicates to the application of such methods in providing approximate inverse preconditioners for square matrices. Test 2 is devoted to compare the schemes for finding the Moore-Penrose inverse of some randomly generated large sparse matrices. And Test 3 gives some comparison for finding the weighted Moore-Penrose inverse of some randomly generated dense matrices.

Test 1. *In order to compare the preconditioners obtained from the new method with the preconditioners of the literature resulted from Incomplete LU factorizations [215], we pay heed of solving the linear sparse systems $Ax = b$, of the dimension 841 using GMRES. The matrix A has been chosen from MatrixMarket database as* `A = ExampleData[ "Matrix","YOUNG1C"]`*, while the right hand side vector is $b = (1,1,...,1)^T$. The solution would then be $(-0.0177027 - 0.00693171I,...,-0.0228083 - 0.00589176I)^T$. Figure 7.3 denotes the plot of the matrix A (note that this matrix is not tridiagonal).*

The left preconditioned system using X_5 of (7.6), X_2 of (7.8), and X_1 of (7.9) and (7.12), along with the well-known preconditioned techniques `ILUT` and `ILUTP` have been tested, while the initial vector has been chosen for all the cases automatically by the command of `LinearSolve[]` in Mathematica 8. The results of time comparisons for different tolerances (residual norms) have been listed in Figure 7.4. The numerical results reveal that by increasing the tolerance the consuming time increase, however the preconditioner X_1 attained from the method (7.12) has mostly the best feedbacks. For this test, we used the initial matrix due to Grosz [216] as follows

$$X_0 = diag(1/a_{11}, 1/a_{22}, \cdots, 1/a_{nn}), \tag{7.56}$$

where a_{ii} is the ith diagonal entry of A.

Remark 1. Note that after a few iterations, the computed preconditioner of the Schulz-type methods may be dense. We must choose a strategy to control the sparsity of the preconditioner. This here can be done by setting the Mathematica command $\texttt{Chop}[\texttt{V}_\texttt{i}, 10^{-5}]$, at the end of each cycle for these matrices. Also notice that for high order methods such as (7.12), mostly one full cycle is enough to be used as an approximate inverse preconditioner.

The most important application of Schulz-type methods is in finding the (pseudo-)inverse of large sparse matrices which possess sparse inverses, [217]. This is the content of the next test problem.

Test 2. *This experiment evaluates the applicability of the new method for finding Moore-Penrose inverse of 15 random sparse complex matrices (possessing sparse pseudo-inverses) of the size* $m \times n = 2200 \times 2500$ *as follows:*

```
m = 2200; n = 2500; number = 15; SeedRandom[123456];
Table[A[l] = SparseArray[{Band[{400, 1}, {m, n}] -> Random[] - I,
       Band[{1, 400}, {m, n}] -> {3.1, -Random[]},
       Band[{-60, 1000}] -> -0.02, Band[{-100, 50}] -> +1.},
        {m, n}, 0.];, {l, number}];
threshold = 10^-10;
```

To save memory space and obtain acceptable computational times, there is no need in full saving of the matrix entries, and we must apply the command `SparseArray[]`*, when working with sparse matrices. Note that herein* $I = \sqrt{-1}$.

In this example, the initial approximate for the Moore-Penrose inverse is constructed by $\texttt{X}_0 = \texttt{Conjugate Transpose}[\texttt{A[j]}] * (1./((\texttt{SingularValueList[A[j],1]}[[1]])^2))$ in our written Mathematica codes, i.e. using (7.26), for each random test matrix. We also defined the identity matrix by $\texttt{Id} = \texttt{SparseArray}[\{\{\texttt{i\_},\texttt{i\_}\} -> 1.\}, \{\texttt{m},\texttt{m}\}, 0.]$. We consider the stopping criterion $||X_{k+1} - X_k||_\infty \leq 10^{-10}$, and a threshold to keep the sparsity features of the output inverses using the command $\texttt{Chop}[\texttt{exp}, \texttt{threshold}]$, in our written codes. Figure 7.5 gives the plots of the sparsity pattern of these test random matrices and the approximate Moore-Penrose inverses.

The results are compared in Figure 7.6 in terms of the number of iterations and Figure 7.7 in terms of the computational time. They show a clear advantage of the new scheme in finding their Moore-Penrose inverse.

Test 3. *In this test, we compute the weighted Moore-penrose inverse of 10 randomly generated dense* $m \times n = 200 \times 210$ *matrices as follows*

```
m = 200; n = 210; number = 10; SeedRandom[12];
Table[A[l] = RandomReal[{1}, {m, n}];, {l, number}];
```

where the 10 different Hermitian positive definite matrices M and N (which have also been constructed randomly) are in what follows

```
Table[MM[l] = RandomReal[{2}, {m, m}];, {l, number}];
Table[MM[l] = Transpose[MM[l]].MM[l];, {l, number}];
Table[NN[l] = RandomReal[{3}, {n, n}];, {l, number}];
Table[NN[l] = Transpose[NN[l]].NN[l];, {l, number}];
```

The results of comparisons using the stopping criterion $||X_{k+1} - X_k||_\infty \leq 10^{-10}$ are reported in Table 1, wherein IT stands for the number of iterations. Although the methods perform slightly the same in terms of the computational time, the proposed method (7.12) is mostly again superior to its competitors.

Using a different stop termination as $||X_{k+1} - X_k||_2 \leq 10^{-10}$, we report the results of comparisons for the test matrices of Test 3, in Table 2. The results fully show that the new method is superior than its competitors.

7.7 Summary

In this chapter, we have presented a new method for matrix inversion. We have also discussed how the proposed method could converge for Moore-Penrose inverse and extended the attained results for finding the weighted Moore-Penrose inverse.

We have also theoretically discussed the computational efficiency of the method. The new scheme reaches a local convergence order equal to nine, by using only seven matrix-matrix products, which makes it efficient in finding the generalized inverses.

Some numerical experiments have also been carried out for showing the efficacy of the contribution for three different types of tests. The numerical results upheld the theoretical conclusions.

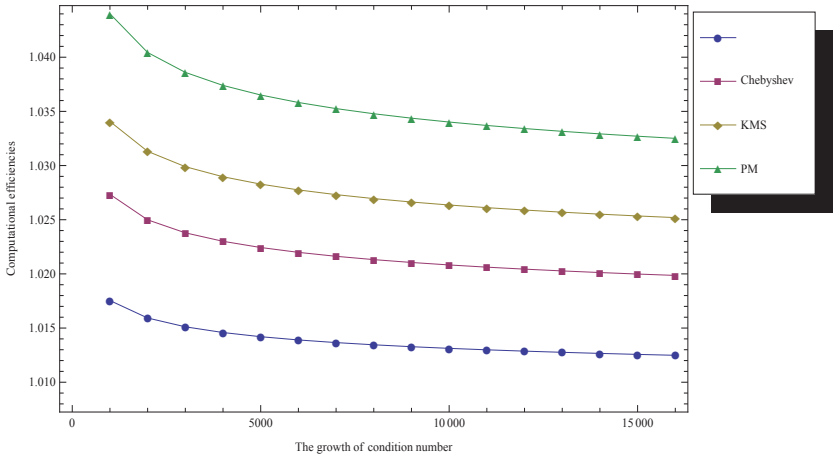

FIGURE 7.1: The comparison of computational efficiency indices for different methods.

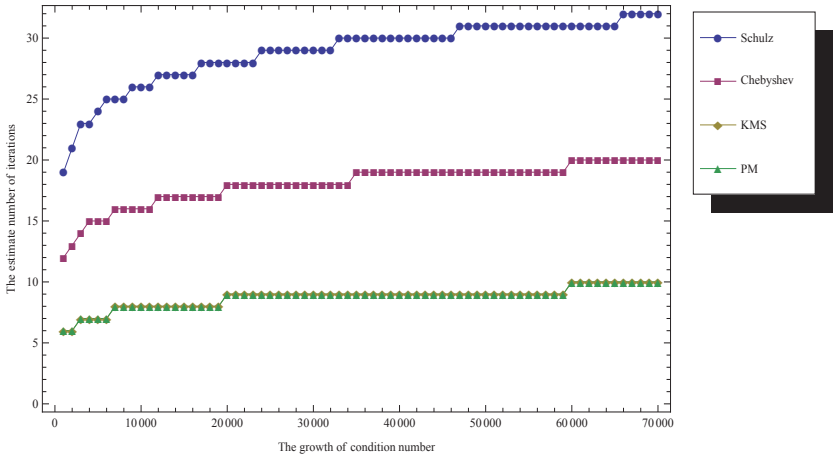

FIGURE 7.2: The comparison of the estimate number of iterations for different methods.

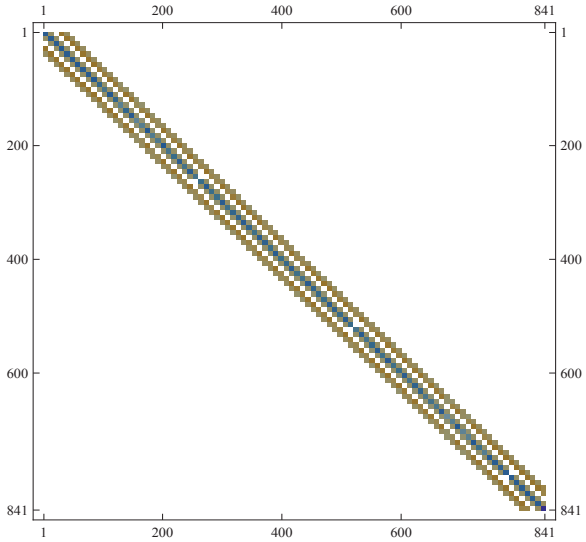

FIGURE 7.3: The plot of the matrix A in Test 1.

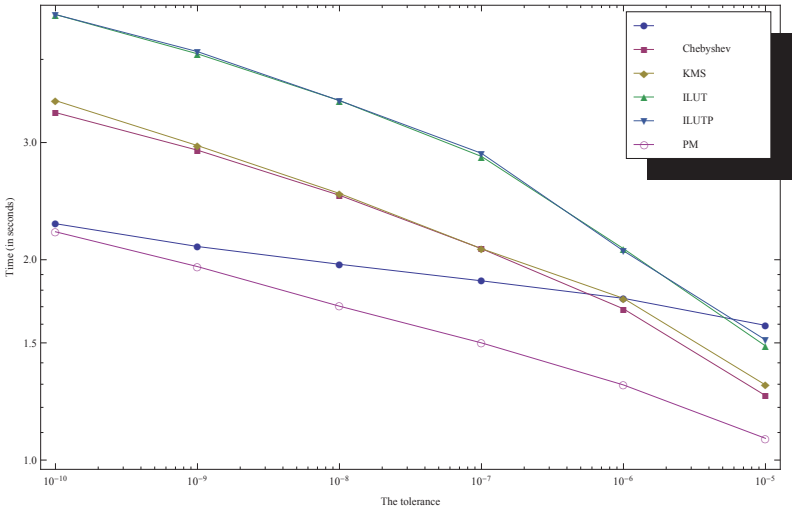

FIGURE 7.4: The results of comparisons in terms of the computational time (right).

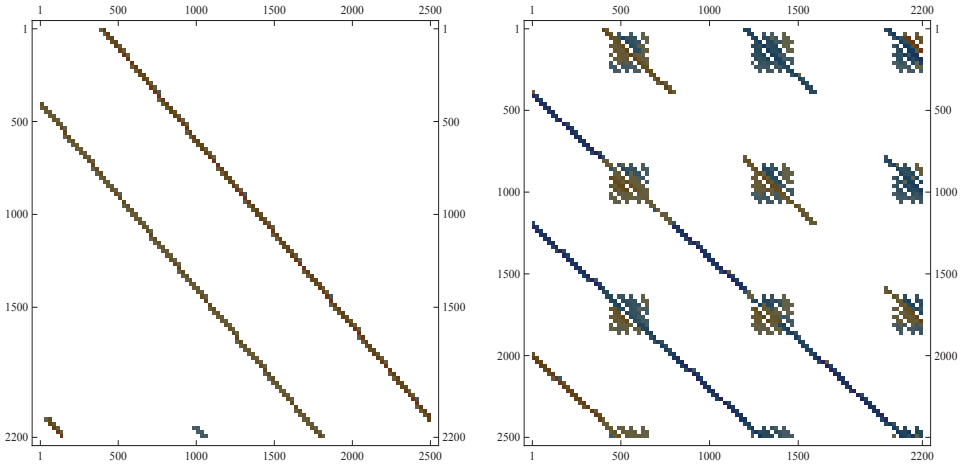

FIGURE 7.5: The general sparsity pattern of the matrices in Test 2 (left) and their approximate Moore-Penrose inverse (right).

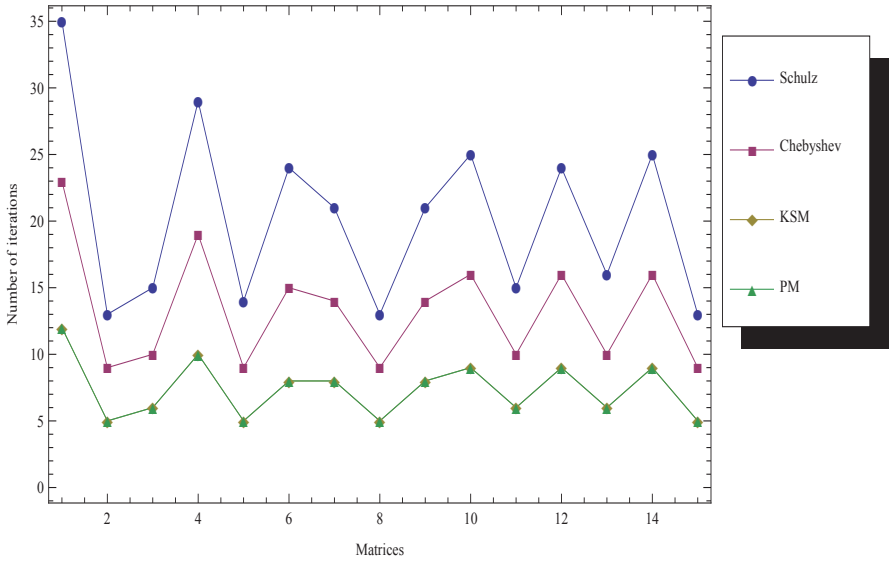

FIGURE 7.6: The results of comparisons for Test 2 in terms of the number of iterations.

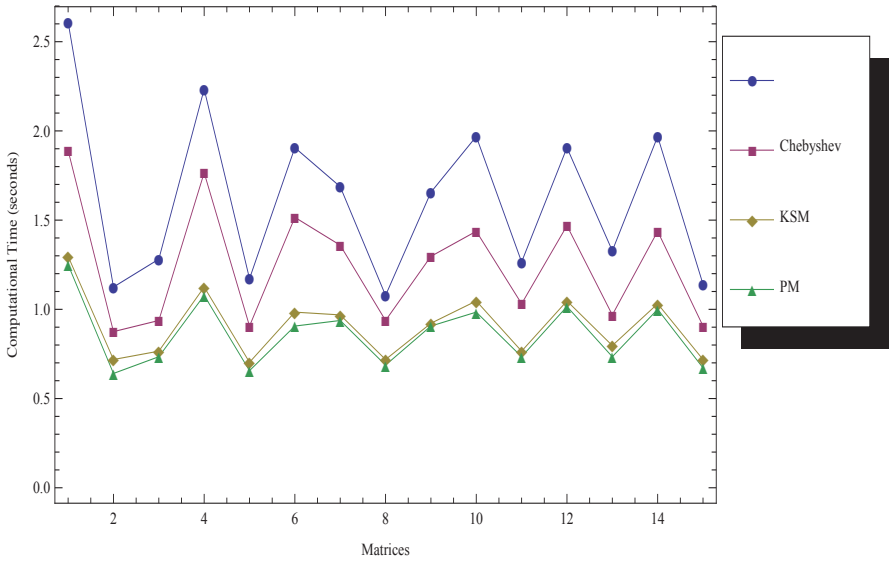

FIGURE 7.7: The results of comparisons for Test 2 in terms of the computational time.

Table 1. Results of comparisons for Test 3 using the stopping criterion $||X_{k+1} - X_k||_\infty \leq 10^{-10}$.

Matrices		Schulz	Chebyshev	KMS	PM
#1	IT	68	43	22	22
	Time	1.093	1.031	1.281	1.031
#2	IT	69	44	23	22
	Time	1.109	1.046	1.343	1.015
#3	IT	67	43	22	22
	Time	1.062	1.015	1.265	1.015
#4	IT	71	46	24	23
	Time	1.140	1.093	1.390	1.062
#5	IT	72	46	24	23
	Time	1.140	1.093	1.406	1.078
#6	IT	72	46	24	23
	Time	1.140	1.093	1.390	1.062
#7	IT	66	42	22	21
	Time	1.062	1.000	1.281	0.968
#8	IT	78	50	26	25
	Time	1.234	1.171	1.500	1.140
#9	IT	64	41	21	21
	Time	1.015	0.968	1.218	0.964
#10	IT	69	44	23	22
	Time	1.109	1.046	1.328	1.015

Table 2. Results of comparisons for Test 3 using the stopping criterion $||X_{k+1} - X_k||_2 \leq 10^{-10}$.

Matrices		Schulz	Chebyshev	KMS	PM
#1	IT	68	43	22	22
	Time	2.546	1.984	1.734	1.609
#2	IT	69	44	23	22
	Time	2.484	1.937	1.953	1.468
#3	IT	67	43	22	22
	Time	2.359	1.890	1.906	1.609
#4	IT	71	46	24	23
	Time	2.578	2.062	1.875	1.546
#5	IT	72	46	24	23
	Time	2.609	2.000	2.031	1.671
#6	IT	72	46	24	23
	Time	2.593	2.015	2.031	1.531
#7	IT	66	42	23	21
	Time	2.406	2.015	1.734	1.531
#8	IT	78	50	26	25
	Time	2.799	2.218	2.187	1.656
#9	IT	63	41	21	21
	Time	2.140	1.718	1.796	1.640
#10	IT	69	44	23	22
	Time	2.515	2.000	1.812	1.468

8. *Conclusions and Future Work*

Higher-order iterative methods to solve nonlinear problems play a significant role in the solution of complex problems. The iterative schemes for a scalar nonlinear equation could help to construct the matrix iterative scheme to compute pseudo-inverses, for example Moore-Penrose inverse and weighted Moore-Penrose inverse of a matrix. Not all but some nonlinear iterative schemes have a natural generalization to construct iterative methods for the systems of nonlinear equations, but it is not always true that they are efficient too.

The multi-step iterative methods are computationally efficient because the LU factors of Jacobian at the initial guess are used in the multi-step part to solve system of linear equations. The multi-step iterative methods are not only computationally efficient, but they also enhance the rate of convergence that in turn offer fast convergence towards a root of a system of nonlinear equations. Usually, higher order Frechet derivatives are prohibited for a general class of systems of nonlinear equations, but there exist some particular cases where we can use them efficiently. The systems of nonlinear equations stemming from ODEs and PDEs have higher-order Frechet derivatives that show the same computational cost as for the Jacobian, owing to the specific structure of the problem. Thus we use higher-order Frechet derivatives to construct higher-order multi-step iterative methods. In the last published article, we achieved a higher order multi-step iterative method for a general class of systems of nonlinear equations that is very efficient. In our future work, we would like to construct iterative methods for systems of nonlinear equations associated with ODEs and PDEs, showing discontinuous nonlinearities. The derivative-free iterative methods with-memory for systems of nonlinear equations represent in our opinion good candidates for these future researches.

Bibliography

[1] N. H. Abel. Beweis der unmöglichkeit, algebraische gleichungen von höheren graden als dem vierten allgemein aufzulösen. *Journal fur die reine und angewandte Mathematik*, 1826:65–84, 2009.

[2] A. M. Ostrowski. Solution of equations and systems of equations, academic press, new york. 1966.

[3] J. F. Traub. Iterative methods for the solution of equations, prentice hall, new york. 1964.

[4] F. I. Chicharro, A. Cordero, and J. R. Torregrosa. Drawing dynamical and parameters planes of iterative families and methods. *The Scientific World Journal*, 2013:11 pages, 2013.

[5] A. Douady and J. H. Hubbard. On the dynamics of polynomials-like mappings. *Annales Scientifiques de l'École Normale Supérieure*, 18:287–343, 1985.

[6] J. Curry, L. Garnet, and D. Sullivan. On the iteration of a rational function: computer experiments with newton's method. *Communications in Mathematical Physics*, 91:267–277, 1983.

[7] P. Blanchard. The dynamics of newton's method. *In Proceedings of the Symposia in Applied Mathematics*, 49:139–154, 1994.

[8] N. Fagella. Invariants in dinàmica complexa. *Butlletí de la Societat Catalana de Matemàtiques*, 23:29–51, 2008.

[9] J. L. Varona. Graphic and numerical comparison between iterative methods. *The Mathematical Intelligencer*, 24:37–46, 2002.

[10] S. Amat, S. Busquier, and S. Plaza. Review of some iterative root-finding methods from a dynamical point of view. *Scientia*, 10:3–35, 2004.

[11] J. M. Gutiérrez, M. A. Hernández, and N. Romero. Dynamics of a new family of iterative processes for quadratic polynomials. *Journal of Computational and Applied Mathematics*, 233:2688–2695, 2010.

[12] G. Honorato, S. Plaza, and N. Romero. Dynamics of a higher-order family of iterative methods. *Journal of Complexity*, 27:221–229, 2011.

[13] F. Chicharro, A. Cordero, J. M. Gutiérrez, and J. R. Torregrosa. Complex dynamics of derivative-free methods for nonlinear equations. *Applied Mathematics and Computation*, 219:7023–7035, 2013.

[14] S. Artidiello, F. Chicharro, A. Cordero, and J. R. Torregrosa. Local convergence and dynamical analysis of a new family of optimal fourth-order iterative methods. *International Journal of Computer Mathematics*, 90:2049–2060, 2013.

[15] M. Scott, B. Neta, and C. Chun. Basin attractors for various methods. *Applied Mathematics and Computation*, 218:2584–2599, 2011.

[16] C. Chun, M. Y. Lee, B. Neta, and J. Dzunic. On optimal fourth-order iterative methods free from second derivative and their dynamics. *Applied Mathematics and Computation*, 218:6427–6438, 2012.

[17] B. Neta, M. Scott, and C. Chun. Basin attractors for various methods for multiple roots. *Applied Mathematics and Computation*, 218:5043–5066, 2011.

[18] A. Cordero, J. G. Maimo, J. R. Torregrosa, M. P. Vassileva, and P. Vindel. Chaos in king's iterative family. *Applied Mathematics Letters*, 26:842–848, 2013.

[19] R. L. Devaney. The mandelbrot set, the farey tree, and the fibonacci sequence. *The American Mathematical Monthly*, 106:289–302, 1999.

[20] Y. I. Kim. A triparametric family of three-step optimal eighth-order methods for solving nonlinear equations. *International Journal of Computer Mathematics*, 89:1051–1059, 2012.

[21] A. Cordero, J. R. Torregrosa, and P. Vindel. Dynamics of a family of chebyshev-halley-type method. *Applied Mathematics and Computation*, 219:8568–8583, 2013.

[22] A. Melman. Geometry and convergence of euler's and halley's methods. *SIAM Review*, 39:728–735, 1997.

[23] M. Davies and B. Dawson. On the global convergence of halley's iteration formula. *Numerische Mathematik*, 24:133–135, 1975.

[24] E. Halley. A new exact and easy method of finding the roots of equations generally and without any previous reduction. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 18:136–148, 1694.

[25] F. Soleymani, M. Sharifi, and B. S. Mousavi. An improvement of ostrowski's and king's techniques with optimal convergence order eight. *Journal of Optimization Theory and Applications*, 153:225–236, 2011.

[26] F Soleymani P Sargolzaei. Accurate fourteenth-order methods for solving nonlinear equations. *Numerical Algorithms*, 58:513–527, 2011.

[27] F. Soleymani, S. K. Khattri, and S. K. Vanani. Two new classes of optimal jarratt-type fourth-order methods. *Applied Mathematics Letters*, 25:847–853, 2012.

[28] F. Soleymani. Regarding the accuracy of optimal eighth-order methods. *Mathematical and Computer Modelling*, 53:1351–1357, 2011.

[29] N. Razmjooy, B. S. Mousavi, and F. Soleymani. A real-time mathematical computer method for potato inspection using machine vision. *Computers and Mathematics with Applications*, 63:268–279, 2012.

[30] F. Soleymani and S. K. Vanani. Optimal steffensen-type methods with eighth order of convergence. *Computers and Mathematics with Applications*, 62:4619–4626, 2011.

[31] F. Soleymani, S. K. Vanani, M. Khan, and M. Sharifi. Some modifications of king's family with optimal eighth order of convergence. *Mathematical and Computer Modelling*, 55:1373–1380, 2012.

[32] F. Soleymani and V. Hosseinabadi. New third-and sixth-order derivative-free techniques for nonlinear equations. *Journal of Mathematics Research*, 3:107–112, 2011.

[33] M. Sharifi, D. K. R. Babajee, and F. Soleymani. Finding the solution of nonlinear equations by a class of optimal methods. *Computers and Mathematics with Applications*, 63:764–774, 2012.

[34] F. Soleymani, S. K. Vanani, and M. J. Paghaleh. A class of three-step derivative-free root solvers with optimal convergence order. *Journal of Applied Mathematics*, 2012:15 pages, 2012.

[35] F. Soleimani and S. Shateyi. Some iterative methods free from derivatives and their basins of attraction for nonlinear equations. *Discrete Dynamics in Nature and Society*, 2013:10 pages, 2013.

[36] F. Soleymani. Revisit of jarratt method for solving nonlinear equations. *Numerical Algorithms*, 57:377–388, 2010.

[37] F. Soleymani, T. Lotfi, and P. Bakhtiari. A multi-step class of iterative methods for nonlinear systems. *Optimization Letters*, 8:1001–1015, 2013.

[38] F. Soleymani, S. K. Vanani, and A. Afghani. A general three-step class of optimal iterations for nonlinear equations. *Mathematical Problems in Engineering*, 2011, 2011.

[39] F. Soleymani and M. Sharifi. On a general efficient class of four-step root-finding methods. *International Journal of Mathematics and Computers in Simulation*, 5: 181–189, 2011.

[40] D. K. R. Babajee, A. Cordero, F. Soleymani, and J. R. Torregrosa. On a novel fourth-order algorithm for solving systems of nonlinear equations. *Journal of Applied Mathematics*, 2012:12 pages, 2012.

[41] F. Soleymani. On a bi-parametric class of optimal eighth-order derivative-free methods. *International Journal of Pure and Applied Mathematics*, 72:27–37, 2011.

[42] F. Soleymani. Concerning some sixth-order iterative methods for finding the simple roots of nonlinear equations. *Bulletin of Mathematical Analysis and Applications*, 2:146–151, 2010.

[43] A. Cordero, T. Lotfi, P. Bakhtiari, and J. R. Torregrosa. An efficient two-parametric family with memory for nonlinear equations. *Numerical Algorithm*, 68:323–335, 2014.

[44] M. S. Petkovi, J. Dzuni, and L. D. Petkovi. A family of two-point methods with memory for solving nonlinear equations. *Applicable Analysis and Discrete Mathematics*, 5:298–317, 2011.

[45] J. R. Sharma, R. K. Guha, and P. Gupta. Some efficient derivative free methods with memory for solving nonlinear equations. *Applied Mathematics and Computation*, 219:699–707, 2012.

[46] J. Dzunic, M. S. Petkovic, and L. D. Petkovic. Three-point methods with and without memory for solving nonlinear equations. *Applied Mathematics and Computation*, 218:4917–4927, 2012.

[47] J. Dzuni. Modified newton's method with memory. *Facta Universitatis, Series Mathematics and Informatics*, 28:429–441, 2013.

[48] M. S. Petkovic, B. Neta, L. D. Petkovic, and J. Dzunic. Multipoint methods for solving nonlinear equations. *A survey, Applied Mathematics and Computation*, 226:635–660, 2014.

[49] T. Lotfi, F. Soleymani, S. Shateyi, P. Assari, and F. K. Haghani. New mono- and biaccelerator iterative methods with memory for nonlinear equations. *Abstract and Applied Analysis*, 2014:8 pages, 2014.

[50] X. Wang, J. Dzunic, and T. Zhang. On an efficient family of derivative free three-point methods for solving nonlinear equations. *Applied Mathematics and Computation*, 219:1749–1760, 2012.

[51] T. Lotfi and E. Tavakoli. n a new efficient steffensen-like iterative class by applying a suitable self-accelerator parameter. *The Scientific World Journal*, 2014: 9 pages, 2014.

[52] J. A. Ezquerro, M. Grau-Sánchez, and M. A. Hernández. Solving non-differentiable equations by a new one-point iterative method with memory. *Journal of Complexity*, 28:48–58, 2012.

[53] J. Dzunic, M. S. Petkovic, and L. D. Petkovic. Three-point methods with and without memory for solving nonlinear equations. *Applied Mathematics and Computation*, 218:4917–4927, 2012.

[54] A. Feldstein and J. F. Traub. Asymptotic behavior of vector recurrences with applications. *Mathematics of Computation*, 31(137 pages):180–192, 1977.

[55] X. Wang and T. Zhang. A new family of newton-type iterative methods with and without memory for solving nonlinear equations. *Calcolo*, 51:1–15, 2014.

[56] X. Wang and T. Zhang. High-order newton-type iterative methods with memory for solving nonlinear equations. *Applied Mathematics and Computation*, 19:91–109, 2014.

[57] M. S. Petkovi, J. Dzuni, and B. Neta. Interpolatory multipoint methods with memory for solving nonlinear equations. *Applied Mathematics and Computation*, 218:2533–2541, 2011.

[58] H. T. Kung and J. F. Traub. Optimal order of one-point and multipoint iteration. *Journal of the Association for Computing Machinery*, 21:643–651, 1994.

[59] F. Soleymani1 and S. Shateyi. Two optimal eighth-order derivative-free classes of iterative methods. *Abstract and Applied Analysis*, 2012:14 pages, 2012.

[60] F. Soleymani. On a new class of optimal eighth-order derivative-free methods. *Acta Universitatis Sapientiae, Mathematica*, 3:169–180, 2011.

[61] M. Matinfar and M. Aminzadeh. A family of optimal derivative free iterative methods with eighth-order convergence for solving nonlinear equations. *Journal of Mathematical Extension*, 6:49–61, 2012.

[62] S. K. Khattri and R. P. Agarwal. Derivative-free optimal iterative methods. *Computational Methods in Applied Mathematics*, 10:368–375, 2010.

[63] R. Thukral. New sixteenth-order derivative-free methods for solving nonlinear equations. *American Journal of Computational and Applied Mathematics*, 2: 112–118, 2010.

[64] F. Ahmad and M. Z. Ullah. Eighth-order derivative-free family of iterative methods for nonlinear equations. *Journal of modern methods in numerical mathematics*, 4:26–33, 2013.

[65] R. Thukral and M. S. Petkovic. A family of three-point methods of optimal order for solving nonlinear equations. *Journal of Computational and Applied Mathematics*, 233:2278–2284, 2010.

[66] F. Soleymani. Novel computational iterative methods with optimal order for nonlinear equations. *Advances in Numerical Analysis*, 2011:10 pages, 2011.

[67] F. Soleymani, S. K. Vanani, H. I. Siyyam, and I. A. Al-Subaihi. Numerical solution of nonlinear equations by an optimal eighth-order class of iterative methods. *Annali Dell'Universita' DI Ferrara*, 59:159–171, 2013.

[68] J. P. Jaiswal and S. panday. An efficient optimal eight-order iterative method for solving nonlinear equations. *Universal Journal of Computational Mathematics*, 1:83–95, 2013.

[69] I. A. Al-Subaihi. Optimal fourth-order iterative methods for solving nonlinear equations. *International Journal of Mathematical Trends and Technology*, 13: 13–18, 2014.

[70] M. Z. Ullah, A. S. Al-Fahid, and F. Ahmad. Four-point optimal sixteenth-order iterative method for solving nonlinear equations. *Journal of applied mathematics*, 2013:5 pages, 2013.

[71] G. Schulz. Iterative berechnung der reziproken matrix. *ZAMM - Journal of Applied Mathematics and Mechanics*, 13:57–59, 1933.

[72] H. Chen and Y. Wang. A family of higher-order convergent iterative methods for computing the moore-penrose inverse. *Applied Mathematics and Computation*, 218:4012–4016, 2011.

[73] D. S. Djordjevic and P. S. Stanimirovic. Iterative methods for computing generalized inverses related with optimization methods. *Journal of the Australian Mathematical Society*, 78:257–272, 2005.

[74] F. Huang and X. Zhang. An improved newton iteration for the weighted moore-penrose inverse. *Applied Mathematics and Computation*, 174:1460–1486, 2006.

[75] M. Z. Nashed and X. Chen. Convergence of newton-like methods for singular operator equations using outer inverses. *Numerische Mathematik*, 66:235–257, 1993.

[76] V. Y. Pan and R. Schreiber. An improved newton iteration for the generalized inverse of a matrix, with applications. *SIAM Journal on Scientific and Statistical Computing*, 12:1109–1131, 1991.

[77] X. Sheng and G. Chen. The generalized weighted moore-penrose inverse. *Journal of Applied Mathematics and Computing*, 25:407–413, 2007.

[78] T. Söderström and G. W. Stewart. On the numerical properties of an iterative method for computing the moore-penrose generalized inverse. *SIAM Journal on Numerical Analysis*, 11:61–74, 1974.

[79] Y. Wei, H. Wu, and J. Wei. Successive matrix squaring algorithm for parallel computing the weighted generalized inverse. *Applied Mathematics and Computation*, 116:289–296, 2000.

[80] W. Li and Z. Li. A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix. *Applied Mathematics and Computation*, 215:3433–3442, 2010.

[81] H. S. Najafi and M. S. Solary. Computational algorithms for computing the inverse of a square matrix, quasi-inverse of a nonsquare matrix and block matrices. *Applied Mathematics and Computation*, 183:539–550, 2006.

[82] F. Toutounian and A. Ataei. A new method for computing moore-penrose inverse matrices. *Journal of Computational and Applied Mathematics*, 228:412–417, 2009.

[83] R. Penrose. A generalized inverses for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51:406–413, 1955.

[84] F. Soleymani. On finding robust approximate inverses for large sparse matrices. *Linear Multilinear Algebra*, 62:1314–1334, 2014.

[85] L. Weiguo, L. Juan, and Q. Tiantian. A family of iterative methods for computing moore-penrose inverse of a matrix. *Linear Algebra and its Applications*, 438:47–56, 2013.

[86] Y. Wei. Recurrent neural networks for computing weighted moore-penrose inverse. *Applied Mathematics and Computation*, 116:279–287, 2000.

[87] P. Stanimirovic and M. Stankovic. Determinantal representation of weighted moore-penrose inverse. *Matematicki Vesnik*, 46:41–50, 1994.

[88] F. Soleymani, E. Tohidi, S. Shateyi, and F. K. Haghani. Some matrix iterations for computing matrix sign function. *Journal of Applied Mathematics*, 2014:9 pages, 2014.

[89] F. Soleymani, P. S. Stanimirovic, S. Shateyi, and F. K. Haghani. Approximating the matrix sign function using a novel iterative method. *Abstract and Applied Analysis*, 2014:9 pages, 2014.

[90] M. Z. Ullah, F. Soleymani, and A. S. Al-Fhaid. An efficient matrix iteration for computing weighted moore-penrose inverse. *Applied Mathematics and Computation*, 226:441–454, 2002.

[91] F. Soleymani. A rapid numerical algorithm to compute matrix inversion. *International Journal of Mathematics and Mathematical Sciences*, 2012, Article ID 134653:11 pages, 2012.

[92] A. R. Soheili, F. Soleymani, and M. D. Petković. On the computation of weighted moore-penrose inverse using a high-order matrix method. *Computers and Mathematics with Applications*, 66:2344–2351, 2013.

[93] F. K. Haghani and F. Soleymani. On a fourth-order matrix method for computing polar decomposition. *Computational and Applied Mathematics*, pages 1–11, 2014.

[94] F. Soleymani, M. Sharifi, and S. Shateyi. Approximating the inverse of a square matrix with application in computation of the moore-penrose inverse. *Journal of Applied Mathematics*, 2014:8 pages, 2014.

[95] A. S. Al-Fhaid, S. Shateyi, M. Z. Ullah, and F. Soleymani. A matrix iteration for finding drazin inverse with ninth-order convergence. *Abstract and Applied Analysis*, 2014:7, 2014.

[96] F. Soleymani and F. K. Haghani. A new high-order stable numerical method for matrix inversion. *The Scientific World Journal*, 2014:10, 2014.

[97] F. Soleymani, M. Sharifi, S. K. Vanani, and F. K. Haghani. An inversion-free method for finding positive definite solution of a rational matrix equation. *The Scientific World Journal*, 2014:5, 2014.

[98] F. Soleymani, M. Sharifi, S. Shateyi, and F. K. Haghani. An algorithm for computing geometric mean of two hermitian positive definite matrices via matrix sign. *Abstract and Applied Analysis*, 2014:6, 2014.

[99] F. Soleymani, H. Salmani, and M. Rasouli. Finding the moore-penrose inverse by a new matrix iteration. *Journal of Applied Mathematics and Computing*, 24: 1–16, 2014.

[100] A. R. Soheili, F. Toutounian, and F. Soleymani. A fast convergent numerical method for matrix sign function with application in sdes. *Mathematical Intelligencer*, 282:167–178, 2015.

[101] F. Soleymani, S. Shateyi, and F. K. Haghani. A numerical method for computing the principal square root of a matrix. *Abstract and Applied Analysis*, 2014:7, 2014.

[102] F. K. Haghani and F. Soleymani. An improved schulz-type iterative method for matrix inversion with application transactions of the institute of measurement and control. *Transactions of the Institute of Measurement and Control*, 36(8): 983–991, 2014.

[103] F. Soleymani. An efficient and stable newton-type iterative method for computing generalized inverse. *Numerical Algorithms*, pages 1–10, 2014.

[104] P. Jarratt. Some fourth order multi-point iterative methods for solving equations. *Mathematics of Computation*, 20:434–437, 1996.

[105] H. Montazeri, F. Soleymani, S. Shateyi, and S. S. Motsa. On a new method for computing the numerical solution of systems of nonlinear equations. *Journal of Applied Mathematics*, 2012:15 pages, 2002.

[106] A. M. Ostrowski. Solutions of equations and systems of equations, academic press, new york-london. 1966.

[107] F. Soleymani, M. Sharifi, S. Shateyi, and F. K. Haghani. A class of steffensen-type iterative methods for nonlinear systems. *Journal of Applied Mathematics*, 2014:9 pages, 2014.

[108] M. A. Noor and M. Waseem. Some iterative methods for solving a system of nonlinear equations. *Computers and Mathematics with Applications*, 57:101–106, 2009.

[109] J. R. Sharma, R. K. Guha, and R. Sharma. An efficient fourth-order weighted-newton method for systems of nonlinear equations. *Numerical Algorithms*, 62: 307–323, 2013.

[110] F. Awawdeh. On new iterative method for solving systems of nonlinear equations. *Numerical Algorithms*, 54:395–409, 2010.

[111] D. K. R. Babajee, M. Z. Dauhooa, M. T. Darvishi, A. Karamib, and A. Barati. Analysis of two chebyshev-like third order methods free from second derivatives for solving systems of nonlinear equations. *Journal of Computational and Applied Mathematics*, 233:2002–2012, 2010.

[112] M. N. Bahrami and R. Oftadeh. Graphic and numerical comparison between iterative methods. *Applied Mathematics and Computation*, 215:37–46, 2009.

[113] J. Biazar and B. Ghanbary. A new technique for solving systems of nonlinear equations. *Applied Mathematical Sciences*, 2:2699–2703, 2008.

[114] A. Cordero, E. Martínez, and J. R. Torregrosa. Iterative methods of order four and .ve for systems of nonlinear equations. *Journal of Computational and Applied Mathematics*, 231:541–551, 2009.

[115] M. T. Darvishi. A two-step high order newton-like method for solving systems of nonlinear equations. *International Journal of Pure and Applied Mathematics*, 57:543–555, 2009.

[116] M. T. Darvishi. Some three-step iterative methods free from second order derivative for finding solutions of systems of nonlinear equations. *International Journal of Pure and Applied Mathematics*, 57:557–573, 2009.

[117] M. T. Darvishi and A. Barati. A third-order newton-type method to solve systems of nonlinear equations. *Applied Mathematics and Computation*, 187:630–635, 2007.

[118] M. T. Darvishi and A. Barati. A forth-order method from quadrature formu-lae to solve systems of nonlinear equations. *Applied Mathematics and Computation*, 188:257–261, 2007.

[119] M. T. Darvishi and A. Barati. Super cubic iterative methods to solve systems of nonlinear equations. *Applied Mathematics and Computation*, 188:1678–1685, 2007.

[120] M. Frontini and E. Sormani. Third-order methods from quadrature formulae for solving systems of nonlinear equations. *Applied Mathematics and Computation*, 149:771–782, 2004.

[121] M. E. Gordji, A. Ebadian, M. B. Ghaemi, and J. Shokri. On systems of nonlinear equations. *arXiv:0904.3460v1 math.DS*, 2009.

[122] C. Grosan and A. Abraham. A new approach for solving nonlinear equations systems. *IEEE Transactions on Systems, Man, and Cybernetics Society*, 38:698–714, 2008.

[123] J. L. Hueso, E. Martínez, and J. R. Torregrosa. Third and fourth order it-erative methods free from second derivative for nonlinear systems. *Applied Mathematics and Computation*, 211:190–197, 2009.

[124] J. L. Hueso, E. Martínez, and J. R. Torregrosa. Third order iterative methods free from second derivative for nonlinear systems. *Applied Mathematics and Computation*, 215:58–65, 2009.

[125] D. Kaya and S. M. El-Sayed. Adomian.s decomposition method applied to systems of nonlinear algebraic equations. *Applied Mathematics and Computation*, 154:487–493, 2004.

[126] J. Kou. A third-order modification of newton method for systems of nonlinear equations. *Applied Mathematics and Computation*, 191:117–121, 2007.

[127] J. Kou, Y. Li, and X. Wang. Efficient continuation newton-like method for solving systems of non-linear equations. *Applied Mathematics and Computation*, 174:846–853, 2006.

[128] J. J. Moré. A collection of nonlinear model problems, in: E.l. allgower, k. georg (eds.), computational solution of nonlinear systems of equations, in: Lectures in applied mathematics. *American Mathematical Society*, 26:723–762, 1990.

[129] G. H. Nedzhibov. A family of multi-point iterative methods for solving systems of nonlinear equations. *Journal of Computational and Applied Mathematics*, 222:244–250, 2008.

[130] M. A. Noor, K. I. Noor, and M. Waseem. Applications of quadrature formula for solving systems of nonlinear equations. *International Journal of Modern Physics B*, 2012.

[131] W. C. Rheinboldt. Methods for solving systems of nonlinear equations, siam, philadelphia. 1998.

[132] B. C. Shin, M. T. Darvishi, and C. H. Kim. A comparison of the newton krylov method with high order newton-like methods to solve nonlinear systems. *Applied Mathematics and Computation*, 217:3190–3198, 2010.

[133] I. G. Tsoulos and A. Stavrakoudis. On locating all roots of systems of nonlinear equations inside bounded domain using global optimization methods. *Nonlinear Analysis: Real World Applications*, 11:2465–2471, 2010.

[134] Vahidi A.R., Babobian E., Cordshooli G.A., and Mirzaie M. Restarted adomian decomposition method to systems of nonlinear algebraic equations. *Applied Mathematical Sciences*, 3:883–889, 2009.

[135] R. E. Bellman and R. E. Kalaba. Quasilinearization and nonlinear boundary-value problems, elsevier, new york, ny, usa. 1965.

[136] R. Krivec and V. B. Mandelzweig. Numerical investigation of quasilinearization method in quantum mechanics. *Computer Physics Communications*, 138:69–79, 2001.

[137] V. B. Mandelzweig. "quasilinearization method and its verification on exactly solvable models in quantum mechanics. *Journal of Mathematical Physics*, 40: 6266–6291, 1999.

[138] V. B. Mandelzweig and F. Tabakin. Quasilinearization approach to nonlinear problems in physics with application to nonlinear odes. *Computer Physics Communications*, 141:268–281, 2001.

[139] V. B. Mandelzweig. Quasilinearization method: nonperturbative approach to physical problems. *Physics of Atomic Nuclei*, 68:1227–1258, 2005.

[140] S. S.Motsa and P. Sibanda. Some modification of the quasilinearization method with higher-order convergence for solving nonlinear bvps. *Numerical Algorithms*, 63:399–417, 2013.

[141] S. S. Motsaa, P. Sibandab, and S. Shateyi. On a new quasi-linearization method for systems of nonlinear boundary value problems. *Mathematical Methods in the Applied Sciences*, 34:1406–1413, 2011.

[142] E. S. Alaidarous, M. Z. Ullah, F. Ahmad, and A.S. Al-Fhaid. An efficient higher-order quasilinearization method for solving nonlinear bvps. *Journal of Applied Mathematics*, 2013, Article ID 259371:11 pages, 2013.

[143] W. Bi, H. Ren, and Q. Wu. Three-step iterative methods with eighth-order convergence for solving nonlinear equations. *Journal of Computational and Applied Mathematics*, 225:105–112, 2009.

[144] W. Bi, Q. Wu, and H. Ren. A new family of eighth-order iterative methods for solving nonlinear equations. *Applied Mathematics and Computation*, 214:236–245, 2009.

[145] Y. H. Geum and Y. I. Kim. A multi-parameter family of three-step eighth-order iterative methods locating a simple root. *Applied Mathematics and Computation*, 215:3375–3382, 2010.

[146] L. Liu and X. Wang. Eighth-order methods with high efficiency index for solving nonlinear equations. *Applied Mathematics and Computation*, 215:3449–3454, 2010.

[147] X. Wang and L. Liu. Modified ostrowski's method with eighth-order convergence and high efficiency index. *Applied Mathematics Letters*, 23:549–554, 2010.

[148] F. Soleymani. On a fast iterative method for approximate inverse of matrices. *Communications of the Korean Mathematical Society*, 28:407–418, 2013.

[149] D. K. R. Babajee and R. Thukral. On a 4-point sixteenth-order king family of iterative methods for solving nonlinear equations. *International Journal of Mathematics and Mathematical Sciences*, 2012, Article ID 979245:13 pages, 2012.

[150] F. Soleymani. A new method for solving ill-conditioned linear systems. *Opuscula Mathematica*, 33:337–344, 2013.

[151] F. Soleymani. On a new class of optimal eight-order derivative-free methods. *Acta Universitatis Sapientiae, Mathematica*, 3:169–180, 2011.

[152] Y. H. Geun and Y. I. Kim. A family of optimal sixteenth-order multipoint methods with a linear fraction plus a trivariate polynomial as the fourth-step weighting function. *Computers and Mathematics with Applications*, 61:3278–3287, 2011.

[153] Y. H. Geum and Y. I. Kim. A biparametic family of optimally convergent sixteenth-order multipoint methods with their fourth-step weighting function as a sum of rational and a generic two-variable function. *Journal of Computational and Applied Mathematics*, 235:3178–3188, 2011.

[154] A. Cordero, J. L. Hueso, E. Martinez, and J. R. Torregrosa. A modified newton-jarratt's composition. *Numerical Algorithms*, 55:87–99, 2010.

[155] E. Pourjafari and H. Mojallali. Solving nonlinear equations systems with a new approach based on invasive weed optimization algorithm and clustering. *Swarm and Evolutionary Computation*, 4:33–43, 2012.

[156] M. Y. Waziri, W. J. Leong, M. A. Hassan, and M. Monsi. A low memory solver for integral equations of chandrasekhar type in the radiative transfer problems. *Mathematical Problems in Engineering*, 2011:12 pages, 2011.

[157] B. H. Dayton, T. Y. Li, and Z. Zeng. Multiple zeros of nonlinear systems. *Mathematics of Computation*, 80:2143–2168, 2011.

[158] M. J. Hirsch, P. M. Pardalos, and M. G. C. Resende. Solving systems of nonlinear equations with continuous grasp. *Nonlinear Analysis: Real World Applications*, 10:2000–2006, 2009.

[159] A. R. Soheili, F. Soleymani, and M. D. Petković. On the computation of weighted moore-penrose inverse using a high-order matrix method. *Computers and Mathematics with Applications*, 69:2344–2351, 2013.

[160] J. M. Ortega and W. C. Rheinboldt. Iterative solution of nonlinear equations in several variables, academic press, new york. 1970.

[161] D. K. R. Babajee. Analysis of higher order variants of newton's method and their applications to differential and integral equations and in ocean acidification, ph.d. thesis, university of mauritius. 2010.

[162] D. W. Decker and C. T. Kelley. Sublinear convergence of the chord method at singular points. *Numerische Mathematik*, 42:147–154, 1983.

[163] S. Wagon. Mathematica in action, third edition, springer, berlin, germany. 2010.

[164] D. K. R. Babajee and M. Z. Dauhoo. Convergence and spectral analysis of the Frontini-Sormani family of multipoint third order methods from Quadrature Rule. *Numerical Algorithms*, 53:467–484, 2010.

[165] http://www.wolfram.com/mathematica/new-in-8/new-and-improved-core-algorithms /index.html.

[166] C. Tadonki. High performance computing as a combination of machines and methods and programming, universite paris sud. 2013.

[167] V. S. Semenov. The method of determining all real non-multiple roots of systems of nonlinear equations. *Computational Mathematics and Mathematical Physics*, 47:1428–1434, 2007.

[168] I. G. Tsoulos and A. Stavrakoudis. On locating all roots of systems of nonlinear equations inside bounded domain using global optimization methods. *Nonlinear Analysis: Real World Applications*, 11:2465–2471, 2010.

[169] F. Soleymani, P. S. Stanimirović, and M. Z. Ullah. On an accelerated iterative method for weighted moore-penrose inverse. *Applied Mathematics and Computation*, 222:365–371, 2013.

[170] F. Soleymani and P. S. Stanimirović. A higher order iterative method for computing the drazin inverse. *The Scientific World Journal*, 2013:11 pages, 1998.

[171] F. Toutounian and F. Soleymani. An iterative method for computing the approximate inverse of a square matrix and the moore-penrose inverse of a non-square matrix. *Applied Mathematics and Computation*, 224:671–680, 2013.

[172] T. Sauer. Numerical analysis, pearson, 2nd edition, usa. 2012.

[173] M. Z. Ullah, A. S. Al-Fhaid, and F. Ahmad. Four-point optimal sixteenth-order iterative method for solving nonlinear equations. *Journal of Applied Mathematics*, 2013:5 pages, 2013.

[174] A. R. Sharma and P. Gupta. An efficient fifth order method for solving systems of nonlinear equations. *Computers and Mathematics with Applications*, 67:591–601, 2014.

[175] A. R. Sharma and R. Sharma. Some third order methods for solving systems of nonlinear equations. *World Academy of Science, Engineering and Technology*, 60, 2011.

[176] J. R. Sharma and H. Arora. Efficient jarratt-like methods for solving systems of nonlinear equations. *Calcolo*, 51:193–210, 2014.

[177] H. Arora J. R. Sharma. A novel derivative free algorithm with seventh order convergence for solving systems of nonlinear equations. *Numerical Algorithms*, 67:917–933, 2014.

[178] V. Candela and A. Marquina. Recurrence relations for rational cubic methods ii: The chebyshev method. *Computing*, 45:355–367, 1990.

[179] V. Kanwar, S. Kumar, and R. Behl. Several new families of jarratt's method for solving systems of nonlinear equations. *Applications and Applied Mathematics: An International Journal*, 8:701–716, 2013.

[180] J. R. Torregrosa, I. K. Argyros, C. Chun, A. Cordero, and F. Soleymani. Iterative methods for nonlinear equations or systems and their applications. *Journal of Applied Mathematics*, 2013:2, 2013.

[181] G. Bratu. Sur les equations integrales non-lineaires. *Bulletins of the Mathematical Society of France*, 42:113–142, 1914.

[182] D. A. Frank-Kamenetzkii. Diffusion and heat transfer in chemical kinetics. plenum press, new york. 1969.

[183] G. Bratu. Sur les equations integrales non-lineaires. *Bulletins of the Mathematical Society of France*, 42:113–142, 1914.

[184] S. S. Motsa and S. Shateyi. New analytic solution to the lane-emden equation of index 2. *Mathematical Problems in Engineering*, 2012:19 pages, 2012.

[185] T. S. Jang. An integral equation formalism for solving the nonlinear klein–gordon equation. *Applied Mathematics and Computation*, 243:322–338, 2014.

[186] J. M. Gutiérrez and M. A. Hernández. A family of chebyshev-halley type methods in banach spaces. *Bulletin of the Australian Mathematical Society*, 55:113–130, 1997.

[187] M. Frontini and E. Sormani. Some variant of newton's method with third-order convergence. *Applied Mathematics and Computation*, 140:419–426, 2003.

[188] H. H. H. Homeier. A modified newton method with cubic convergence: the multivariable case. *Journal of Computational and Applied Mathematics*, 169: 161–169, 2004.

[189] A. Cordero and J. R. Torregrosa. Variants of newton's method using fifth-order quadrature formulas. *Applied Mathematics and Computation*, 190:686–698, 2007.

[190] M. G. Sánchez, À. Grau, and M. Noguera. Ostrowski type methods for solving systems of nonlinear equations. *Applied Mathematics and Computation*, 218: 2377–2385, 2011.

[191] C. T. Kelley. Solving nonlinear equations with newton's method, siam, philadelphia. 2003.

[192] A. S. Al-Fhaid M. Z. ullah, F. Soleymani. Numerical solution of nonlinear systems by a general class of iterative methods with application to nonlinear pdes. *Numerical Algorithms*, 67:223–242, 2014.

[193] A. Cordero, J. L. Hueso, E. Martínez, and J. R. Torregrosa. Increasing the convergence order of an iterative method for nonlinear systems. *Applied Mathematics Letters*, 25:2369–2374, 2012.

[194] N. M. L. Romeiro C. A. Ladeia. Numerical solutions of the 1d convection-diffusion-reaction and the burgers equation using implicit multi-stage and finite element methods. *Integral Methods in Science and Engineering*, pages 205–216, 2013.

[195] S. Shateyi H. Montazeri, F. Soleymani and S. S. Motsa. On a new method for computing the numerical solution of systems of nonlinear equations. *Journal of Applied Mathematics*, 2012:15 pages, 2012.

[196] A. Ben-Israel and T.N.E. Greville. Generalized inverses, springer, 2nd edition. 2003.

[197] M. E. Gulliksson, P. A. Wedin, and Y. Wei. Perturbation identities for regularized tikhonov inverse and weighted pseudo inverse. *BIT*, 40:513–523, 2000.

[198] W. Sun and Y.Wei. Inverse order rule for weighted generalized inverses. *SIAM Journal on Matrix Analysis and Applications*, 19:772–775, 1998.

[199] R. Penrose. A generalized inverses for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51:406–413, 1955.

[200] M. D. Petković, P. S. Stanimirović, and M. B. Tasić. Effective partitioning method for computing weighted moore-penrose inverse. *Computers and Mathematics with Applications*, 55:1720–1734, 2008.

[201] S. K. Sen, H. Agarwal, and S. Sen. Chemical equation balancing: An integer programming approach. *Mathematical and Computer Modelling*, 44:678–691, 2006.

[202] T. N. E. Grevile. Some applications of the pseudo-inverse of matrix. *SIAM Review*, 3:15–22, 1960.

[203] G. R. Wang. A new proof of grevile's method for computing the weighted mp inverse. *Journal of Shangai Normal University*, 3, 1985.

[204] C. F. Van Loan. Generalizing the singular value decomposition. *SIAM Journal on Numerical Analysis*, 13:76–83, 1976.

[205] F. Huang and X. Zhang. An improved newton iteration for the weighted moore-penrose inverse. *Applied Mathematics and Computation*, 174:1460–1486, 2006.

[206] H. Hotelling. Analysis of a complex statistical variable into principal components. *Journal of Educational Psychology*, 24:498–520, 1933.

[207] V. Y. Pan and R. Schreiber. An improved newton iteration for the generalized inverse of a matrix, with applications. *SIAM Journal on Scientific and Statistical Computing*, 91:1109–1131, 2012.

[208] S. K. Sen and S. S. Prabhu. Optimal iterative schemes for computing moore-penrose matrix inverse. *International Journal of Systems Science*, 8:748–753, 1976.

[209] E. V. Krishnamurthy and S. K. Sen. Numerical algorithms - computations in science and engineering, affiliated east-west press, new delhi. 2007.

[210] E. Isaacson and H. B. Keller. Analysis of numerical methods, wiley, new york. 1966.

[211] J. M. McNamee and V. Y. Pan. Efficient polynomial root-refiners: A survey and new record efficiency estimates. *Computers and Mathematics with Applications*, 63:239–254, 2012.

[212] V. Y . Pan. Newton's iteration for matrix inversion, advances and extensions. matrix methods: Theory, algorithms and applications. pages 364–381, 2010.

[213] P. S. Stanimirovic and D. S. Cvetkovic-wllic. Successive matrix squaring algorithm for computing outer inverse. *Applied Mathematics and Computation*, 203: 19–29, 2008.

[214] M. Trott. The mathematica guide-book for numerics, springer-verlag. 2006.

[215] Y. Saad. Iterative methods for sparse linear systems, 2ed., siam, usa. 2003.

[216] L. Grosz. Preconditioning by incomplete block elimination. *Numerical Linear Algebra with Applications*, 7:527–541, 2000.

[217] B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin. Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM Journal on Scientific Computing*, 14:159–184, 1993.