ARTICLE IN PRESS

Computers and Mathematics with Applications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs

Fayyaz Ahmad [a], Emran Tohidi [b,*], Malik Zaka Ullah [c,d], Juan A. Carrasco [e]

[a] Department de Física i Enginyeria Nuclear, Universitat Politècnica de Catalunya, Comte d'Urgell 187, 08036 Barcelona, Spain

[b] Department of Mathematics, Kosar University of Bojnord, P. O. Box 9415615458, Bojnord, Iran

[c] Dipartimento di Scienza e Alta Tecnologia, Universita dell'Insubria, Via Valleggio 11, 22100 Como, Italy

[d] Operator Theory and Applications Research Group, Department of Mathematics, Faculty of Science, King Abdulaziz University, P.O. Box 80203, Jeddah 21589, Saudi Arabia

[e] Departament d'Enginyeria Electrònica, Universitat Politècnica de Catalunya, Diagonal 647, planta 9, 08028 Barcelona, Spain

ARTICLE INFO

Article history:
Received 5 November 2014
Received in revised form 8 May 2015
Accepted 9 May 2015
Available online xxxx

Keywords:
Multi-step iterative methods
Systems of nonlinear equations
Newton's method
Computational efficiency
Nonlinear ordinary differential equations
Nonlinear partial differential equations

ABSTRACT

This paper proposes a multi-step iterative method for solving systems of nonlinear equations with a local convergence order of $3m-4$, where m (≥ 2) is the number of steps. The multi-step iterative method includes two parts: the base method and the multi-step part. The base method involves two function evaluations, two Jacobian evaluations, one LU decomposition of a Jacobian, and two matrix–vector multiplications. Every stage of the multi-step part involves the solution of two triangular linear systems and one matrix–vector multiplication. The computational efficiency of the new method is better than those of previously proposed methods. The method is applied to several nonlinear problems resulting from discretizing nonlinear ordinary differential equations and nonlinear partial differential equations.

1. Introduction

A multi-step iterative method includes the base method and the multi-step part. The base method is designed to be computationally efficient. Since matrix inversion is an expensive operation, most base methods involve only one inversion of the Jacobian matrix. In the multi-step part, systems of linear equations should be solved by using the inverse of the Jacobian matrix computed in the base method. For instance, one can use the LU-factors of the Jacobian matrix. Other expensive operations which should be minimized in multi-step iterative methods include functions and Jacobian evaluations, matrix–vector multiplications, vector–vector multiplications, and solutions of systems of linear equations. Recently, Malik et al. [1] have constructed a general class of multi-step iterative methods with two matrix inversions in the base method, making those methods expensive. The method proposed in [2] also involves two matrix inversions in the base method. Independent recent efforts by two different research groups have resulted [3,4] in the same multi-step iterative method. In this paper, we will improve the higher order multi-step Jarratt-like (HJ) method described in [3,4]. That method can be described, taking

* Corresponding author.
E-mail addresses: fayyaz.ahmad@upc.edu (F. Ahmad), emrantohidi@gmail.com (E. Tohidi), malik.zakaullah@uninsubria.it (M.Z. Ullah), juan.a.carrasco@upc.edu (J.A. Carrasco).

http://dx.doi.org/10.1016/j.camwa.2015.05.012
0898-1221/

note of the convergence order and the computational cost, as

$$\mathrm{HJ} = \begin{cases} \text{Number of steps} & = m \geq 2 \\ \text{Convergence order} & = 2m \\ \text{Function evaluations} & = m-1 \\ \text{Jacobian evaluations} & = 2 \\ \text{LU decomposition} & = 1 \\ \text{Matrix–vector multiplications} & = m \\ \text{Vector–vector multiplications} & = 2m \\ \text{Number of solutions of systems of lower and upper triangular systems of equations} & = 2m-1 \end{cases} \begin{cases} \text{Base method} \longrightarrow & \begin{cases} \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_1 = \mathbf{F}(\mathbf{x}_k) \\ \mathbf{y}_1 = \mathbf{x}_k - \dfrac{2}{3}\boldsymbol{\phi}_1 \\ \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_2 = \mathbf{F}'(\mathbf{y}_1)\,\boldsymbol{\phi}_1 \\ \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_3 = \mathbf{F}'(\mathbf{y}_1)\,\boldsymbol{\phi}_2 \\ \mathbf{y}_2 = \mathbf{x}_k - \dfrac{23}{8}\boldsymbol{\phi}_1 + 3\boldsymbol{\phi}_2 - \dfrac{9}{8}\boldsymbol{\phi}_3 \end{cases} \\ \text{Multi-step part} \rightarrow & \begin{cases} \text{For } s = 1, m-2 \\ \quad \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_{2s+2} = \mathbf{F}(\mathbf{y}_{s+1}) \\ \quad \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_{2s+3} = \mathbf{F}'(\mathbf{y}_1)\,\boldsymbol{\phi}_{2s+2} \\ \quad \mathbf{y}_{s+2} = \mathbf{y}_{s+1} - \dfrac{5}{2}\boldsymbol{\phi}_{2s+2} + \dfrac{3}{2}\boldsymbol{\phi}_{2s+3} \\ \text{End} \end{cases} \end{cases}$$

where $\mathbf{F}'(\cdot)$ denotes the Fréchet derivative [5] or the Jacobian of $\mathbf{F}(\cdot)$. The base method in HJ has a convergence order of four and involves one LU decomposition, one function evaluation, and two Jacobian evaluations. Each step of the multi-step part increases the convergence order by two. In 2015, Malik et al. [6] developed another efficient multi-step iterative method (MSF) for solving nonlinear systems arising from particular ODEs which can be described as

$$\mathrm{MSF} = \begin{cases} \text{Number of steps} & = m \\ \text{Convergence-order} & = 3m \\ \text{Function evaluations} & = m \\ \text{Jacobian evaluations} & = 2 \\ \text{Second-order Fréchet derivative} & = 1 \\ \text{LU decomposition} & = 1 \\ \text{Matrix–vector multiplications} & = 2m-2 \\ \text{Vector–vector multiplications} & = m+2 \\ \text{Number of solutions of systems of lower and upper triangular systems of equations} & = 3m-1 \end{cases} \begin{cases} \text{Base method} \longrightarrow & \begin{cases} \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_1 = \mathbf{F}(\mathbf{x}_k) \\ \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_2 = \mathbf{F}''(\mathbf{x}_k)\,\boldsymbol{\phi}_1^2 \\ \mathbf{y}_1 = \mathbf{x}_k - \boldsymbol{\phi}_1 - \dfrac{1}{2}\boldsymbol{\phi}_2 \end{cases} \\ \text{Multi-step part} \rightarrow & \begin{cases} \text{For } s = 1, m-1 \\ \quad \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_{3s} = \mathbf{F}(\mathbf{y}_s) \\ \quad \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_{3s+1} = \mathbf{F}'(\mathbf{y}_1)\,\boldsymbol{\phi}_{3s} \\ \quad \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_{3s+2} = \mathbf{F}'(\mathbf{y}_1)\,\boldsymbol{\phi}_{3s+1} \\ \quad \mathbf{y}_{s+1} = \mathbf{y}_s - 3\boldsymbol{\phi}_{3s} + 3\boldsymbol{\phi}_{3s+1} \\ \qquad\qquad -\boldsymbol{\phi}_{3s+2} \\ \text{End.} \end{cases} \end{cases}$$

The limitation of MSF is that it was only constructed for a particular class of ordinary differential equations (ODEs) of the form $L(x(t)) + f(x(t)) = g(t)$. The MSF method uses a second-order Fréchet derivative that is a diagonal matrix. The computational cost of that second-order Fréchet derivative is prohibitive for general systems of nonlinear equations. Interested readers can find information about other multi-step iterative methods for scalar as well as systems of nonlinear equations in [7–18].

The structure of this paper is as follows. In the next section, a new multi-step iterative method will be proposed. The order of convergence of the new method will be studied in Section 3. Section 4 will derive an efficiency index for the method taking into account both order of convergence and computational cost. To confirm theoretical predictions, some test problems resulting from discretizing nonlinear ordinary differential equations and nonlinear partial differential equations (PDEs) will be considered in Section 5. Conclusions will be given in Section 6.

2. New multi-step iterative method

The new multi-step iterative method (FTUC) can be described as

$$\mathrm{FTUC} = \begin{cases} \text{Number of steps} & = m \geq 3 \\ \text{Convergence-order} & = 3m-4 \\ \text{Function evaluations} & = m-1 \\ \text{Jacobian evaluations} & = 2 \\ \text{LU decomposition} & = 1 \\ \text{Matrix–vector multiplications} & = m-1 \\ \text{Vector–vector multiplications} & = m+1 \\ \text{Number of solutions of systems of lower and upper triangular systems of equations} & = 2m-2 \end{cases} \begin{cases} \text{Base method} \longrightarrow & \begin{cases} \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_1 = \mathbf{F}(\mathbf{x}_k) \\ \mathbf{y}_1 = \mathbf{x}_k - \boldsymbol{\phi}_1 \\ \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_2 = \mathbf{F}(\mathbf{y}_1) \\ \mathbf{y}_2 = \mathbf{y}_1 - 3\boldsymbol{\phi}_2 \\ \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_3 = \mathbf{F}'(\mathbf{y}_2)\,\boldsymbol{\phi}_2 \\ \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_4 = \mathbf{F}'(\mathbf{y}_2)\,\boldsymbol{\phi}_3 \\ \mathbf{y}_3 = \mathbf{y}_1 - \dfrac{7}{4}\boldsymbol{\phi}_2 + \dfrac{1}{2}\boldsymbol{\phi}_3 + \dfrac{1}{4}\boldsymbol{\phi}_4 \end{cases} \\ \text{Multi-step part} \rightarrow & \begin{cases} \text{For } s = 1, m-3 \\ \quad \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_{2s+3} = \mathbf{F}(\mathbf{y}_{s+2}) \\ \quad \mathbf{F}'(\mathbf{x}_k)\,\boldsymbol{\phi}_{2s+4} = \mathbf{F}'(\mathbf{y}_2)\,\boldsymbol{\phi}_{2s+3} \\ \quad \mathbf{y}_{s+3} = \mathbf{y}_{s+2} - 2\boldsymbol{\phi}_{2s+3} + \boldsymbol{\phi}_{2s+4} \\ \text{End.} \end{cases} \end{cases}$$

The convergence-order of the base method of FTUC is five, and each step of the multi-step part increases the convergence order by three. We will prove the convergence order for $m = 4$ and, then, will use mathematical induction to obtain the convergence order for any m.

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

3. Convergence analysis

In this section, we will prove that the local convergence-order of FTUC is eight for $m = 4$ and later we will establish the proof for the convergence order for arbitrary m via mathematical induction.

Theorem 3.1. *Let $\mathbf{F} : \Gamma \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be sufficiently Fréchet differentiable on an open convex neighborhood Γ of $\mathbf{x}^* \in \mathbb{R}^n$ with $\mathbf{F}(\mathbf{x}^*) = 0$ and $\det\left(\mathbf{F}'(\mathbf{x}^*)\right) \neq 0$. Then the sequence $\{\mathbf{x}_k\}$ generated by FTUC converges to $\mathbf{x}^*$ with local order of convergence at least eight and the following error equation*

$$\mathbf{e}_{k+1} = \mathbf{L}\mathbf{e}_k{}^8 + O\left(\mathbf{e}_k{}^9\right), \tag{1}$$

where $\mathbf{e}_k = \boldsymbol{x}_k - \mathbf{x}^*$, $\mathbf{e}_k{}^p = \overbrace{(\mathbf{e}_k, \mathbf{e}_k, \ldots, \mathbf{e}_k)}^{p \text{ times}}$ and $\mathbf{L} = -15\,\mathbf{C}_3\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2 + 45\,\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 + 120\,\mathbf{C}_3\mathbf{C}_2^5 + 5\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 - 15\,\mathbf{C}_2\mathbf{C}_3^2\mathbf{C}_2^2 + 150\,\mathbf{C}_2^3\mathbf{C}_3\mathbf{C}_2^2 - 40\,\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^4 - 50\,\mathbf{C}_2^4\mathbf{C}_3\mathbf{C}_2 + 400\,\mathbf{C}_2^7$ is a p-linear function i.e. $\mathbf{L} \in \mathbb{L}\left(\overbrace{\mathbb{R}^n, \mathbb{R}^n, \ldots, \mathbb{R}^n}^{p \text{ times}}\right)$ and $\mathbf{L}\mathbf{e}_k{}^p \in \mathbb{R}^n$.

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

$$\begin{aligned}
\mathbf{F}(\mathbf{x}_k) &= \mathbf{F}\left(\mathbf{x}^* + \mathbf{x}_k - \mathbf{x}^*\right) = \mathbf{F}\left(\mathbf{x}^* + \mathbf{e}_k\right), \\
&= \mathbf{F}\left(\mathbf{x}^*\right) + \mathbf{F}'\left(\mathbf{x}^*\right)\mathbf{e}_k + \frac{1}{2!}\mathbf{F}''\left(\mathbf{x}^*\right)\mathbf{e}_k{}^2 + \frac{1}{3!}\mathbf{F}^{(3)}\left(\mathbf{x}^*\right)\mathbf{e}_k{}^3 + \cdots, \\
&= \mathbf{F}'\left(\mathbf{x}^*\right)\left(\mathbf{e}_k + \frac{1}{2!}\mathbf{F}'\left(\mathbf{x}^*\right)^{-1}\mathbf{F}''\left(\mathbf{x}^*\right)\mathbf{e}_k{}^2 + \frac{1}{3!}\mathbf{F}'\left(\mathbf{x}^*\right)^{-1}\mathbf{F}^{(3)}\left(\mathbf{x}^*\right)\mathbf{e}_k{}^3 + \cdots\right), \\
&= \mathbf{C}_1\left(\mathbf{e}_k + \mathbf{C}_2\,\mathbf{e}_k{}^2 + \mathbf{C}_3\,\mathbf{e}_k{}^3 + O\left(\mathbf{e}_k{}^4\right)\right),
\end{aligned} \tag{2}$$

where $\mathbf{C}_1 = \mathbf{F}'(\mathbf{x}^*)$ and $\mathbf{C}_s = \frac{1}{s!}\mathbf{F}'(\mathbf{x}^*)^{-1}\mathbf{F}^{(s)}(\mathbf{x}^*)$ for $s \geq 2$. From (2), we can calculate the Fréchet derivative of $\mathbf{F}$ as

$$\mathbf{F}'(\mathbf{x}_k) = \mathbf{C}_1\left(\mathbf{I} + 2\mathbf{C}_2\,\mathbf{e}_k + 3\mathbf{C}_3\,\mathbf{e}_k{}^2 + 4\mathbf{C}_3\,\mathbf{e}_k{}^3 + O\left(\mathbf{e}_k{}^4\right)\right), \tag{3}$$

where $\mathbf{I}$ is the identity matrix. Furthermore, using the Maple software package, we obtained the following expression for the inverse of the Fréchet derivative:

$$\begin{aligned}
\mathbf{F}'(\mathbf{x}_k)^{-1} = \Big(&\mathbf{I} - 2\mathbf{C}_2\,\mathbf{e}_k + \left(4\mathbf{C}_2^2 - 3\mathbf{C}_3\right)\mathbf{e}_k^2 + \left(6\mathbf{C}_3\mathbf{C}_2 + 6\mathbf{C}_2\mathbf{C}_3 - 8\mathbf{C}_2^3 - 4\mathbf{C}_4\right)\mathbf{e}_k^3 \\
&+ \Big(8\mathbf{C}_4\mathbf{C}_2 + 9\mathbf{C}_3^2 + 8\mathbf{C}_2\mathbf{C}_4 - 5\mathbf{C}_5 - 12\mathbf{C}_3\mathbf{C}_2^2 - 12\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 - 12\mathbf{C}_2^2\mathbf{C}_3 + 16\mathbf{C}_2^4\Big)\mathbf{e}_k^4 \\
&+ \Big(24\mathbf{C}_3\mathbf{C}_2^3 + 24\mathbf{C}_2^3\mathbf{C}_3 + 24\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2 + 24\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 + 10\mathbf{C}_5\mathbf{C}_2 + 12\mathbf{C}_4\mathbf{C}_3 + 12\mathbf{C}_3\mathbf{C}_4 + 10\mathbf{C}_2\mathbf{C}_5 \\
&- 6\mathbf{C}_6 - 16\mathbf{C}_4\mathbf{C}_2^2 - 18\mathbf{C}_3^2\mathbf{C}_2 - 18\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 - 16\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2 - 18\mathbf{C}_2\mathbf{C}_3^2 - 16\mathbf{C}_2^2\mathbf{C}_4 - 32\mathbf{C}_2^5\Big)\mathbf{e}_k^5 \\
&+ \Big(32\mathbf{C}_4\mathbf{C}_2^3 + 64\mathbf{C}_2^6 - 48\mathbf{C}_3\mathbf{C}_2^4 + 12\mathbf{C}_2\mathbf{C}_6 + 16\mathbf{C}_4^2 + 15\mathbf{C}_3\mathbf{C}_5 + 15\mathbf{C}_5\mathbf{C}_3 + 12\mathbf{C}_6\mathbf{C}_2 - 24\mathbf{C}_4\mathbf{C}_2\mathbf{C}_3 \\
&- 24\mathbf{C}_4\mathbf{C}_3\mathbf{C}_2 - 20\mathbf{C}_2^2\mathbf{C}_5 - 24\mathbf{C}_2\mathbf{C}_3\mathbf{C}_4 - 24\mathbf{C}_2\mathbf{C}_4\mathbf{C}_3 + 32\mathbf{C}_2^3\mathbf{C}_4 - 20\mathbf{C}_2\mathbf{C}_5\mathbf{C}_2 + 36\mathbf{C}_2^2\mathbf{C}_3^2 \\
&- 20\mathbf{C}_5\mathbf{C}_2^2 + 32\mathbf{C}_2^2\mathbf{C}_4\mathbf{C}_2 + 32\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2^2 + 36\mathbf{C}_2\mathbf{C}_3^2\mathbf{C}_2 + 36\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 + 36\mathbf{C}_3^2\mathbf{C}_2^2 - 7\mathbf{C}_7 \\
&- 24\mathbf{C}_3\mathbf{C}_2\mathbf{C}_4 - 27\mathbf{C}_3^3 - 24\mathbf{C}_3\mathbf{C}_4\mathbf{C}_2 + 36\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 + 36\mathbf{C}_3\mathbf{C}_2^2\mathbf{C}_3 - 48\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2^2 \\
&- 48\mathbf{C}_2^3\mathbf{C}_3\mathbf{C}_2 - 48\mathbf{C}_2^4\mathbf{C}_3 - 48\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^3\Big)\mathbf{e}_k^6 + O\left(\mathbf{e}_k{}^7\right)\Big)\mathbf{C}_1^{-1}.
\end{aligned} \tag{4}$$

Substituting (2) and (4) in $\boldsymbol{\phi}_1 = \mathbf{F}'(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{x}_k)$, we get

$$\begin{aligned}
\boldsymbol{\phi}_1 = \mathbf{e}_k &- \mathbf{C}_2\,\mathbf{e}_k^2 + \left(2\mathbf{C}_2^2 - 2\mathbf{C}_3\right)\mathbf{e}_k^3 + \left(-3\mathbf{C}_4 - 4\mathbf{C}_2^3 + 3\mathbf{C}_3\mathbf{C}_2 + 4\mathbf{C}_2\mathbf{C}_3\right)\mathbf{e}_k^4 \\
&+ \left(-4\mathbf{C}_5 - 6\mathbf{C}_3\mathbf{C}_2^2 - 6\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 - 8\mathbf{C}_2^2\mathbf{C}_3 + 8\mathbf{C}_2^4 + 4\mathbf{C}_4\mathbf{C}_2 + 6\mathbf{C}_3^2 + 6\mathbf{C}_2\mathbf{C}_4\right)\mathbf{e}_k^5 \\
&+ \Big(-5\mathbf{C}_6 + 12\mathbf{C}_3\mathbf{C}_2^3 + 16\mathbf{C}_2^3\mathbf{C}_3 + 12\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2 + 12\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 - 8\mathbf{C}_4\mathbf{C}_2^2 - 9\mathbf{C}_3^2\mathbf{C}_2 - 12\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 \\
&- 8\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2 - 12\mathbf{C}_2\mathbf{C}_3^2 - 12\mathbf{C}_2^2\mathbf{C}_4 - 16\mathbf{C}_2^5 + 5\mathbf{C}_5\mathbf{C}_2 + 8\mathbf{C}_4\mathbf{C}_3 + 9\mathbf{C}_3\mathbf{C}_4 + 8\mathbf{C}_2\mathbf{C}_5\Big)\mathbf{e}_k^6 + O\left(\mathbf{e}_k{}^7\right).
\end{aligned} \tag{5}$$

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

Using $\mathbf{y}_1 = \mathbf{x} - \boldsymbol{\phi}_1$, we obtain

$$\mathbf{y}_1 - \mathbf{x}^* = \mathbf{x} - \mathbf{x}^* - \boldsymbol{\phi}_1 = \mathbf{e}_k - \boldsymbol{\phi}_1,$$

and substituting (5), we get

$$\begin{aligned}\mathbf{y}_1 - \mathbf{x}^* = {} & \mathbf{C}_2\,\mathbf{e}_k^2 + \left(-2\mathbf{C}_2^2 + 2\mathbf{C}_3\right)\mathbf{e}_k^3 + \left(-3\mathbf{C}_3\mathbf{C}_2 - 4\mathbf{C}_2\mathbf{C}_3 + 4\mathbf{C}_2^3 + 3\mathbf{C}_4\right)\mathbf{e}_k^4 \\ & + \left(-4\mathbf{C}_4\mathbf{C}_2 - 6\mathbf{C}_3^2 - 6\mathbf{C}_2\mathbf{C}_4 + 6\mathbf{C}_3\mathbf{C}_2^2 + 8\mathbf{C}_2^2\mathbf{C}_3 + 6\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 - 8\mathbf{C}_2^4 + 4\mathbf{C}_5\right)\mathbf{e}_k^5 \\ & + \left(16\mathbf{C}_2^5 - 5\mathbf{C}_5\mathbf{C}_2 - 8\mathbf{C}_4\mathbf{C}_3 - 9\mathbf{C}_3\mathbf{C}_4 - 8\mathbf{C}_2\mathbf{C}_5 + 8\mathbf{C}_4\mathbf{C}_2^2 + 12\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 + 9\mathbf{C}_3^2\mathbf{C}_2 + 12\mathbf{C}_2\mathbf{C}_3^2\right. \\ & \left. + 12\mathbf{C}_2^2\mathbf{C}_4 + 8\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2 - 12\mathbf{C}_3\mathbf{C}_2^3 - 12\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2 - 16\mathbf{C}_2^3\mathbf{C}_3 - 12\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 + 5\mathbf{C}_6\right)\mathbf{e}_k^6 + O\left(\mathbf{e}_k{}^7\right). \end{aligned} \tag{6}$$

Substituting (4) and

$$\mathbf{F}(\mathbf{y}_1) = \mathbf{C}_1\left(\mathbf{y}_1 + \mathbf{C}_2\,\mathbf{y}_1{}^2 + \mathbf{C}_3\,\mathbf{y}_1{}^3 + \cdots\right)$$

in $\boldsymbol{\phi}_2 = \mathbf{F}'(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{y}_1)$, we get

$$\begin{aligned}\boldsymbol{\phi}_2 = {} & \mathbf{C}_2\mathbf{e}_k^2 + \left(-4\mathbf{C}_2^2 + 2\mathbf{C}_3\right)\mathbf{e}_k^3 + \left(3\mathbf{C}_4 - 8\mathbf{C}_2\mathbf{C}_3 - 6\mathbf{C}_3\mathbf{C}_2 + 13\mathbf{C}_2^3\right)\mathbf{e}_k^4 \\ & + \left(4\mathbf{C}_5 - 38\mathbf{C}_2^4 + 20\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 + 26\mathbf{C}_2^2\mathbf{C}_3 + 18\mathbf{C}_3\mathbf{C}_2^2 - 12\mathbf{C}_2\mathbf{C}_4 - 12\mathbf{C}_3^2 - 8\mathbf{C}_4\mathbf{C}_2\right)\mathbf{e}_k^5 \\ & + \left(5\mathbf{C}_6 + 104\mathbf{C}_2^5 + 27\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2 + 40\mathbf{C}_2\mathbf{C}_3^2 + 39\mathbf{C}_2^2\mathbf{C}_4 + 27\mathbf{C}_3^2\mathbf{C}_2 + 36\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 - 50\mathbf{C}_3\mathbf{C}_2^3\right. \\ & \left. - 16\mathbf{C}_2\mathbf{C}_5 - 18\mathbf{C}_3\mathbf{C}_4 - 16\mathbf{C}_4\mathbf{C}_3 - 10\mathbf{C}_5\mathbf{C}_2 + 24\mathbf{C}_4\mathbf{C}_2^2 - 55\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 - 76\mathbf{C}_2^3\mathbf{C}_3 - 59\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2\right)\mathbf{e}_k^6 + O\left(\mathbf{e}_k{}^7\right). \end{aligned} \tag{7}$$

Using $\mathbf{y}_2 = \mathbf{y}_1 - 3\boldsymbol{\phi}_2$ and substituting (6) and (7), we get

$$\begin{aligned}\mathbf{y}_2 - \mathbf{x}^* = {} & -2\mathbf{C}_2\,\mathbf{e}_k^2 + \left(10\mathbf{C}_2^2 - 4\mathbf{C}_3\right)\mathbf{e}_k^3 + \left(-6\mathbf{C}_4 + 15\mathbf{C}_3\mathbf{C}_2 + 20\mathbf{C}_2\mathbf{C}_3 - 35\mathbf{C}_2^3\right)\mathbf{e}_k^4 \\ & + \left(-8\mathbf{C}_5 + 106\mathbf{C}_2^4 + 20\mathbf{C}_4\mathbf{C}_2 + 30\mathbf{C}_3^2 + 30\mathbf{C}_2\mathbf{C}_4 - 48\mathbf{C}_3\mathbf{C}_2^2 - 70\mathbf{C}_2^2\mathbf{C}_3 - 54\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2\right)\mathbf{e}_k^5 \\ & + \left(-296\mathbf{C}_2^5 - 73\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2 - 108\mathbf{C}_2\mathbf{C}_3^2 - 105\mathbf{C}_2^2\mathbf{C}_4 - 72\mathbf{C}_3^2\mathbf{C}_2 - 96\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 + 138\mathbf{C}_3\mathbf{C}_2^3\right. \\ & + 40\mathbf{C}_2\mathbf{C}_5 + 45\mathbf{C}_3\mathbf{C}_4 + 40\mathbf{C}_4\mathbf{C}_3 + 25\mathbf{C}_5\mathbf{C}_2 - 64\mathbf{C}_4\mathbf{C}_2^2 + 153\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 + 212\mathbf{C}_2^3\mathbf{C}_3 \\ & \left. + 165\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2 - 10\mathbf{C}_6\right)\mathbf{e}_k^6 + O\left(\mathbf{e}_k{}^7\right). \end{aligned} \tag{8}$$

Substituting (4), (7) and (8) in $\boldsymbol{\phi}_3 = \mathbf{F}'(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{y}_2)\,\boldsymbol{\phi}_2$, we get

$$\begin{aligned}\boldsymbol{\phi}_3 = {} & \mathbf{C}_2\,\mathbf{e}_k^2 + \left(-6\mathbf{C}_2^2 + 2\mathbf{C}_3\right)\mathbf{e}_k^3 + \left(3\mathbf{C}_4 - 9\mathbf{C}_3\mathbf{C}_2 - 12\mathbf{C}_2\mathbf{C}_3 + 21\mathbf{C}_2^3\right)\mathbf{e}_k^4 \\ & + \left(4\mathbf{C}_5 - 44\mathbf{C}_2^4 + 36\mathbf{C}_3\mathbf{C}_2^2 + 42\mathbf{C}_2^2\mathbf{C}_3 + 30\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 - 12\mathbf{C}_4\mathbf{C}_2 - 18\mathbf{C}_3^2 - 18\mathbf{C}_2\mathbf{C}_4\right)\mathbf{e}_k^5 \\ & + \left(5\mathbf{C}_6 - 10\mathbf{C}_2^5 - 101\mathbf{C}_3\mathbf{C}_2^3 - 55\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2 - 88\mathbf{C}_2^3\mathbf{C}_3 - 65\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 + 48\mathbf{C}_4\mathbf{C}_2^2 + 72\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3\right. \\ & \left. + 54\mathbf{C}_3^2\mathbf{C}_2 + 63\mathbf{C}_2^2\mathbf{C}_4 + 60\mathbf{C}_2\mathbf{C}_3^2 + 39\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2 - 15\mathbf{C}_5\mathbf{C}_2 - 24\mathbf{C}_4\mathbf{C}_3 - 27\mathbf{C}_3\mathbf{C}_4 - 24\mathbf{C}_2\mathbf{C}_5\right)\mathbf{e}_k^6 + O\left(\mathbf{e}_k{}^7\right). \end{aligned} \tag{9}$$

Using $\boldsymbol{\phi}_4 = \mathbf{F}'(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{y}_2)\,\boldsymbol{\phi}_3$ and substituting (4), (8) and (9), we get

$$\begin{aligned}\boldsymbol{\phi}_4 = {} & \mathbf{C}_2\mathbf{e}_k^2 + \left(-8\mathbf{C}_2^2 + 2\mathbf{C}_3\right)\mathbf{e}_k^3 + \left(-12\mathbf{C}_3\mathbf{C}_2 - 16\mathbf{C}_2\mathbf{C}_3 + 33\mathbf{C}_2^3 + 3\mathbf{C}_4\right)\mathbf{e}_k^4 \\ & + \left(60\mathbf{C}_3\mathbf{C}_2^2 + 66\mathbf{C}_2^2\mathbf{C}_3 + 46\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 - 16\mathbf{C}_4\mathbf{C}_2 - 24\mathbf{C}_3^2 - 24\mathbf{C}_2\mathbf{C}_4 - 66\mathbf{C}_2^4 + 4\mathbf{C}_5\right)\mathbf{e}_k^5 \\ & + \left(80\mathbf{C}_4\mathbf{C}_2^2 + 120\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 + 90\mathbf{C}_3^2\mathbf{C}_2 + 99\mathbf{C}_2^2\mathbf{C}_4 + 92\mathbf{C}_2\mathbf{C}_3^2 + 59\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2 - 20\mathbf{C}_5\mathbf{C}_2 - 32\mathbf{C}_4\mathbf{C}_3\right. \\ & \left. - 36\mathbf{C}_3\mathbf{C}_4 - 32\mathbf{C}_2\mathbf{C}_5 - 152\mathbf{C}_2^5 - 188\mathbf{C}_3\mathbf{C}_2^3 - 71\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2 - 132\mathbf{C}_2^3\mathbf{C}_3 - 107\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 + 5\mathbf{C}_6\right)\mathbf{e}_k^6 + O\left(\mathbf{e}_k{}^7\right). \end{aligned} \tag{10}$$

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

Using $\mathbf{y}_3 = \mathbf{y}_1 - (7/4)\,\boldsymbol{\phi}_2 + (1/2)\,\boldsymbol{\phi}_3 + (1/4)\,\boldsymbol{\phi}_4$ and substituting (6), (7), (9), and (10), we get

$$\begin{aligned}\mathbf{y}_3 - \mathbf{x}^* = {} & \left(20\mathbf{C}_2^4 - (5/2)\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 + (15/2)\mathbf{C}_3\mathbf{C}_2^2\right)\mathbf{e}_k^5 + \Big(-209\mathbf{C}_2^5 - 5\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2 - 5\mathbf{C}_2\mathbf{C}_3^2 + (45/4)\mathbf{C}_3^2\mathbf{C}_2 \\ & + 15\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 - 22\mathbf{C}_3\mathbf{C}_2^3 + 10\mathbf{C}_4\mathbf{C}_2^2 + 25\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 + 40\mathbf{C}_2^3\mathbf{C}_3 + 46\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2\Big)\mathbf{e}_k^6 \\ & + \Big(-(267/2)\mathbf{C}_3\mathbf{C}_2^4 - (15/2)\mathbf{C}_2\mathbf{C}_3\mathbf{C}_4 - 10\mathbf{C}_2\mathbf{C}_4\mathbf{C}_3 - (15/2)\mathbf{C}_2\mathbf{C}_5\mathbf{C}_2 + 15\mathbf{C}_3\mathbf{C}_4\mathbf{C}_2 + (45/2)\mathbf{C}_3^3 \\ & + (45/2)\mathbf{C}_3\mathbf{C}_2\mathbf{C}_4 + 15\mathbf{C}_4\mathbf{C}_3\mathbf{C}_2 + 20\mathbf{C}_4\mathbf{C}_2\mathbf{C}_3 - 44\mathbf{C}_4\mathbf{C}_2^3 + 72\mathbf{C}_2^2\mathbf{C}_4\mathbf{C}_2 + 92\mathbf{C}_2^2\mathbf{C}_3^2 + 60\mathbf{C}_2^3\mathbf{C}_4 \\ & + 54\mathbf{C}_2\mathbf{C}_3^2\mathbf{C}_2 + 50\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 + 40\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2^2 - 50\mathbf{C}_3^2\mathbf{C}_2^2 - 44\mathbf{C}_3\mathbf{C}_2^2\mathbf{C}_3 - 20\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 \\ & - (451/2)\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^3 - 357\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2^2 - 418\mathbf{C}_2^4\mathbf{C}_3 - 385\mathbf{C}_2^3\mathbf{C}_3\mathbf{C}_2 + 25/2\mathbf{C}_5\mathbf{C}_2^2 + 1316\mathbf{C}_2^6\Big)\mathbf{e}_k^7 + O\left(\mathbf{e}_k{}^8\right). \end{aligned} \tag{11}$$

Substituting (4) and

$$\mathbf{F}(\mathbf{y}_3) = \mathbf{C}_1\left(\mathbf{y}_3 + \mathbf{C}_2\,\mathbf{y}_3{}^2 + \mathbf{C}_3\,\mathbf{y}_3{}^3 + \cdots\right),$$

in $\boldsymbol{\phi}_5 = \mathbf{F}'(\mathbf{x}_k)^{-1}\mathbf{F}(\mathbf{y}_3)$, we get

$$\begin{aligned}\boldsymbol{\phi}_5 = {} & \left(20\mathbf{C}_2^4 - (5/2)\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 + (15/2)\mathbf{C}_3\mathbf{C}_2^2\right)\mathbf{e}_k^5 + \Big(-22\mathbf{C}_3\mathbf{C}_2^3 + 51\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2 + 40\mathbf{C}_2^3\mathbf{C}_3 \\ & + 10\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 - 249\mathbf{C}_2^5 + 10\mathbf{C}_4\mathbf{C}_2^2 + 15\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 + (45/4)\mathbf{C}_3^2\mathbf{C}_2 - 5\mathbf{C}_2\mathbf{C}_3^2 - 5\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2\Big)\mathbf{e}_k^6 \\ & + \Big(-(387/2)\mathbf{C}_3\mathbf{C}_2^4 - (15/2)\mathbf{C}_2\mathbf{C}_3\mathbf{C}_4 - 10\mathbf{C}_2\mathbf{C}_4\mathbf{C}_3 - (15/2)\mathbf{C}_2\mathbf{C}_5\mathbf{C}_2 + 15\mathbf{C}_3\mathbf{C}_4\mathbf{C}_2 + (45/2)\mathbf{C}_3^3 \\ & + (45/2)\mathbf{C}_3\mathbf{C}_2\mathbf{C}_4 + 15\mathbf{C}_4\mathbf{C}_3\mathbf{C}_2 + 20\mathbf{C}_4\mathbf{C}_2\mathbf{C}_3 - 44\mathbf{C}_4\mathbf{C}_2^3 + 82\mathbf{C}_2^2\mathbf{C}_4\mathbf{C}_2 + 102\mathbf{C}_2^2\mathbf{C}_3^2 + 60\mathbf{C}_2^3\mathbf{C}_4 \\ & + (63/2)\mathbf{C}_2\mathbf{C}_3^2\mathbf{C}_2 + 20\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 + 20\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2^2 - (145/2)\mathbf{C}_3^2\mathbf{C}_2^2 - 44\mathbf{C}_3\mathbf{C}_2^2\mathbf{C}_3 - (25/2)\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 \\ & - (363/2)\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^3 - 377\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2^2 - 498\mathbf{C}_2^4\mathbf{C}_3 - 487\mathbf{C}_2^3\mathbf{C}_3\mathbf{C}_2 + 25/2\mathbf{C}_5\mathbf{C}_2^2 + 1814\mathbf{C}_2^6\Big)\mathbf{e}_k^7 + O\left(\mathbf{e}_k{}^8\right). \end{aligned} \tag{12}$$

Substituting (4), (8) and (12) in $\boldsymbol{\phi}_6 = \mathbf{F}'(\mathbf{x}_k)^{-1}\mathbf{F}'(\mathbf{y}_2)\,\boldsymbol{\phi}_5$, we get

$$\begin{aligned}\boldsymbol{\phi}_6 = {} & \left(20\mathbf{C}_2^4 - (5/2)\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 + (15/2)\mathbf{C}_3\mathbf{C}_2^2\right)\mathbf{e}_k^5 + \Big(-289\mathbf{C}_2^5 + 10\mathbf{C}_4\mathbf{C}_2^2 + 15\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3 \\ & + (45/4)\mathbf{C}_3^2\mathbf{C}_2 - 5\mathbf{C}_2\mathbf{C}_3^2 - 5\mathbf{C}_2\mathbf{C}_4\mathbf{C}_2 - 22\mathbf{C}_3\mathbf{C}_2^3 + 56\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2 + 40\mathbf{C}_2^3\mathbf{C}_3 - 5\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2\Big)\mathbf{e}_k^6 \\ & + \Big(2312\mathbf{C}_2^6 - (507/2)\mathbf{C}_3\mathbf{C}_2^4 - (275/2)\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^3 - 397\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2^2 - 578\mathbf{C}_2^4\mathbf{C}_3 - 589\mathbf{C}_2^3\mathbf{C}_3\mathbf{C}_2 + (25/2)\mathbf{C}_5\mathbf{C}_2^2 \\ & + 20\mathbf{C}_4\mathbf{C}_2\mathbf{C}_3 + 15\mathbf{C}_4\mathbf{C}_3\mathbf{C}_2 + (45/2)\mathbf{C}_3\mathbf{C}_2\mathbf{C}_4 + (45/2)\mathbf{C}_3^3 + 15\mathbf{C}_3\mathbf{C}_4\mathbf{C}_2 - (15/2)\mathbf{C}_2\mathbf{C}_5\mathbf{C}_2 - 10\mathbf{C}_2\mathbf{C}_4\mathbf{C}_3 \\ & - (15/2)\mathbf{C}_2\mathbf{C}_3\mathbf{C}_4 - 44\mathbf{C}_4\mathbf{C}_2^3 - 44\mathbf{C}_3\mathbf{C}_2^2\mathbf{C}_3 - 95\mathbf{C}_3^2\mathbf{C}_2^2 + 92\mathbf{C}_2^2\mathbf{C}_4\mathbf{C}_2 + 112\mathbf{C}_2^2\mathbf{C}_3^2 + 60\mathbf{C}_2^3\mathbf{C}_4 \\ & + 9\mathbf{C}_2\mathbf{C}_3^2\mathbf{C}_2 - 5\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 - 10\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3\Big)\mathbf{e}_k^7 + O\left(\mathbf{e}_k{}^8\right). \end{aligned} \tag{13}$$

Finally, using $\mathbf{y}_4 - \mathbf{x}^* = \mathbf{y}_3 - \mathbf{x}^* - 2\,\boldsymbol{\phi}_5 + 2\,\boldsymbol{\phi}_6$ and substituting (11)–(13), we get

$$\begin{aligned}\mathbf{y}_4 - \mathbf{x}^* = {} & \Big(-15\mathbf{C}_3\mathbf{C}_2^2\mathbf{C}_3\mathbf{C}_2 + 45\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^2 + 120\mathbf{C}_3\mathbf{C}_2^5 + 5\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2 - 15\mathbf{C}_2\mathbf{C}_3^2\mathbf{C}_2^2 + 150\mathbf{C}_2^3\mathbf{C}_3\mathbf{C}_2^2 \\ & - 40\mathbf{C}_2\mathbf{C}_3\mathbf{C}_2^4 - 50\mathbf{C}_2^4\mathbf{C}_3\mathbf{C}_2 + 400\mathbf{C}_2^7\Big)\mathbf{e}_k^8 + O\left(\mathbf{e}_k{}^9\right). \quad \square \end{aligned} \tag{14}$$

Theorem 3.2. *The multi-step iterative method FTUC has local convergence order at least* $3m - 4$ *for* $m \geq 3$.

Proof. The proof is established by mathematical induction. For $m = 3, 4$ the convergence orders are according to (11) and (14) five and eight, respectively. Consequently, our claim concerning the convergence-order $3m - 4$ is true for $m = 3, 4$.

Assume our claim is true for $m = q \geq 4$, i.e., that the convergence order of FTUC for $m = q$ is $3q - 4$. The qth and $(q-1)$th steps FTUC can be written as

$$\begin{aligned} & \mathbf{F}'(\mathbf{x}_k)\,\mathbf{T} = \mathbf{F}'(\mathbf{y}_2) \\ & \text{Frozen factor} = (2\mathbf{I} - \mathbf{T})\,\mathbf{F}'(\mathbf{x}_k)^{-1}, \\ & \mathbf{y}_{q-1} = \mathbf{y}_{q-2} - (\text{Frozen factor})\,\mathbf{F}\left(\mathbf{y}_{q-2}\right), \\ & \mathbf{y}_q = \mathbf{y}_{q-1} - (\text{Frozen factor})\,\mathbf{F}\left(\mathbf{y}_{q-1}\right). \end{aligned}$$

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

The enhancement in the convergence order of FTUC from the $(q-1)$th step to the qth step is $(3q-4)-(3(q-1)-4)=3$. Now, we write the $(q+1)$th step of FTUC as

$$\mathbf{y}_{q+1} = \mathbf{y}_q - (\text{Frozen factor})\,\mathbf{F}\left(\mathbf{y}_q\right).$$

The increment in the convergence-order of FTUC, due to $(q+1)$th-step, is precisely three because the use of the Frozen factor adds an additive constant to the convergence order [2]. Therefore, the convergence order after the addition of the $(q+1)$th step is $3q-4+3=3q-1=3(q+1)-4$, completing the induction step. □

4. Efficiency index

In the literature, the computational efficiency of an iterative method is evaluated by the efficiency index defined as

$$E = p^{1/C}, \tag{15}$$

where p is the order of convergence of the method and C is the computational cost of the method defined as

$$C(\mu_0,\ \mu_1,\ n) = P_0(n)\,\mu_0 + P_1(n)\,\mu_1 + P(n), \tag{16}$$

where $P_0(n)$ is the number of scalar function f_i evaluations in $\mathbf{F}(\cdot)$, $P_1(n)$ is the number of scalar function $\frac{\partial f_i}{\partial x_j}$ evaluations in the Jacobian, $P(n)$ is the number of products/divisions, and μ_i are coefficients which are required to express the computational cost in terms of products/divisions. Table 1 gives the number of products, divisions and computational cost of some operations in terms on the number of variables n. In the table and in the sequel, l denotes the cost of a division relative to the cost of a multiplication.
The computational costs of HJ and FTUC, C_{HJ} and C_{FTUC}, are

$$C_{HJ} = (m-1)\,\mu_0 n + 2n^2\mu_1 + mn^2 + 2mn + (2m-1)\,(n(n-1)+ln) + \frac{n(n-1)(2n-1)}{6} + l\,\frac{n(n-1)}{2}, \tag{17}$$

$$C_{FTUC} = (m-1)\,\mu_0 n + 2n^2\mu_1 + (m-1)\,n^2 + (m+1)\,n + (2m-2) \times (n(n-1)+ln) + \frac{n(n-1)(2n-1)}{6} + l\frac{n(n-1)}{2}. \tag{18}$$

To compare the efficiencies of HJ and FTUC we define the quotient

$$R = \frac{\log\left((3m_1-4)^{1/C_{FTUC}}\right)}{\log\left((2m_2)^{1/C_{HJ}}\right)} = \frac{C_{HJ}}{C_{FTUC}}\,\frac{\log(3m_1-4)}{2m_2}, \tag{19}$$

where m_1 is the number of steps of FTUC and m_2 is the number of steps of HJ. Clearly, if $R>1$ the efficiency of FTUC is higher than that of HJ.

4.1. Comparison of the efficiencies of FTUC and HJ for convergence orders five and four, respectively

When FTUC and HJ have convergence orders five and four, respectively, the value of R is

$$R = \frac{ln(5)\left(3ln+12\mu_1 n+2n^2+15l+6\mu_0+27n+7\right)}{2\left(3ln+12\mu_1 n+2n^2+21l+12\mu_0+33n+1\right)ln(2)}. \tag{20}$$

R is >1 if

$$\mu_0 < 0.06394801344n^2 + (0.09592202004l + 0.3836880802\mu_1 - 0.3285458591)\,n - 0.7122339393l + 1.415662087. \tag{21}$$

In that case the base method of FTUC will be more efficient than the base method of HJ.

4.2. Comparison of the efficiencies of FTUC and HJ for the same convergence order

FTUC and HJ have convergence order $6s-4$ for numbers of steps $m_1=2s$ and $m_2=3s-2$, $s\geq 2$, respectively. In that case,

$$R = 1 + \frac{6(2ls+\mu_0 s+3ns-3l-2\mu_0-4n+2s-2)}{3ln+24ls+12\mu_0 s+12\mu_1 n+2n^2+36ns-15l-6\mu_0-21n-12s+19}. \tag{22}$$

The value of l depends on the computer system where the methods are run. Reasonable values for l fall between 2.5 and 3. Replacing s and l by $s+2$ and $l+1$ in (22) gives

$$R = 1 + \frac{6(2ls+\mu_0 s+3ns+l+2n+4s+3)}{3ln+24ls+12\mu_0 s+12\mu_1 n+2n^2+36ns+33l+18\mu_0+54n+12s+28}, \tag{23}$$

which is >1 for $s\geq 0$. Since $l>1$, this shows that $R>1$ and FTUC is more efficient than HJ when $s\geq 2$, i.e. with a convergence order ≥ 8.

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

Table 1
Computational cost of different operations.

	Multiplications	Divisions	Computational cost
LU decomposition	$\frac{n(n-1)(2n-1)}{6}$	$\frac{n(n-1)}{2}$	$\frac{n(n-1)(2n-1)}{6} + l\,\frac{n(n-1)}{2}$
Solution of two triangular systems	$n\,(n-1)$	n	$n\,(n-1) + ln$
Matrix–vector multiplication	n^2		n^2
Vector–vector multiplication	n		n

Table 2
Comparison between computational efficiencies when FTUC and HJ make the same number of function evaluations.

Number of steps	Function evaluations	Convergence order of (FTUC, HJ)	R
4	3	(8, 8)	$1 + \frac{6(l+2n+3)}{12n\mu_1+3ln+2n^2+18\mu_0+33l+54n+28}$
5	4	(11, 10)	$1 + \frac{0.16\mu_0+0.08n\mu_1+2.57n+4.40+1.35l+0.01n^2+0.02ln}{4.0\mu_0+2.0n\mu_1+12.0n+5.66+7.50l+0.33n^2+0.50ln}$
6	5	(14, 12)	$1 + \frac{0.31\mu_0+0.12n\mu_1+3.05n+5.72+1.65l+0.02n^2+0.03ln}{5.0\mu_0+2.0n\mu_1+15.0n+6.66+9.50l+0.33n^2+0.50ln}$
100	99	(296, 200)	$1 + \frac{7.32\mu_0+0.14n\mu_1+24.12n+113.77+15.68l+0.02n^2+0.03ln}{99.0\mu_0+2.0n\mu_1+297.0n+100.66+197.50l+0.33n^2+0.50ln}$

Table 3
Comparison between FTUC and HJ when both methods make the same number of function evaluations.

	FTUC ($m \geq 3$)	HJ ($m \geq 2$)
Number of steps	m	m
Convergence order	$3m-4$	$2m$
Function evaluations	$m-1$	$m-1$
Jacobian evaluations	2	2
LU decompositions	1	1
Matrix–vector multiplications	$m-1$	m
Vector–vector multiplications	$m+1$	$2m$
Number of solutions of systems of linear equations	$2m-2$	$2m-1$

4.3. Comparison of FTUC and HJ for the same number of function evaluations

The values of R when both methods make the same number of function evaluations (≥ 3) are given for several cases in Table 2. In the expressions for R, l is replaced by $l+1$. We can see that, for the same number of function evaluations, the convergence order of FTUC is higher than that of HJ for a number of steps larger than 4. Also, since $l > 1$, the values of $R-1$ reveal that FTUC has, for the same number of function evaluations, higher efficiency than HJ. Table 3 compares the characteristics of FTUC and HJ. For two function evaluations the convergence orders of FTUC and HJ are five and six, respectively. In that case, the performance of HJ is better than that of FTUC. However, for a number of function evaluations greater then two, the performance of FTUC is better than that of HJ.

4.4. Comparison of FTUC and MSF for the same number of function evaluations

The MSF method uses m_1 function evaluations and one second-order Fréchet derivative. For the purpose of comparison, we assume that the evaluation of a second-order Fréchet derivative has same computational cost as that of a function. The above assumption only makes sense if the second-order Fréchet derivative is a diagonal matrix. Under that assumption, MSF makes m_1+1 function evaluations to achieve a convergence order of $3m_1$ in m_1 steps and FTUC makes m_2-1 function evaluations to achieve a convergence order of $3m_2-4$ in m_2 steps. If we equate the number of function evaluations of FTUC and MSF we get

$$m_2 - 1 = m_1 + 1,$$
$$m_2 = m_1 + 2.$$

Suppose $m_1 = m$ and $m_2 = m+2$. Table 4 compares the characteristics of FTUC and MSF for those numbers of steps. MSF achieves a convergence order of $3m$ in m steps by making $m+1$ function evaluations while FTUC achieves a convergence order of $3m+2$ in $m+2$ steps by making $m+1$ function evaluations. MSF makes $m-3$, $m+3$ and $m-3$ more matrix–vector multiplications, vector–vector multiplications and solutions of upper and lower triangular systems of linear equations than FTUC. Obviously, for the same number of function evaluations, FTUC is computationally more efficient than MSF and achieves a better convergence order. The high computational cost of second-order Fréchet derivatives makes that method impractical for general systems of nonlinear equations. Since, even if the second-order Fréchet derivative is a diagonal matrix, MSF has worse performance than FTUC, we will not test numerically MSF against FTUC.

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

Table 4
Comparison of FTUC and MSF for, respectively, $m+2$ and m steps.

	FTUC	MSF
Number of steps	$m+2$	m
Convergence order	$3m+2$	$3m$
Function evaluations	$m+1$	$m+1$
Jacobian evaluations	2	2
LU decompositions	1	1
Matrix–vector multiplications	$m+1$	$2m-2$
Vector–vector multiplications	$m+3$	$m+2$
Number of solutions of lowers and upper triangular systems	$2m+2$	$3m-1$

Table 5
Results under FTUC and HJ for the first example.

Iterative methods		FTUC	HJ
Number of iterations		3	1
Size of problem		4	4
Number of steps		6	7
Theoretical convergence order		14	14
Computational convergence order		14.1	14.1
Number of function evaluations per iteration		5	6
Solutions of system of linear equations per iteration		10	13
Number of matrix–vector multiplication per iteration		5	7
	Iteration		
$\|\mathbf{y}_q - \mathbf{y}^*\|_1$	1	8.21e−10	1.66e−10
	2	4.76e−136	2.51e−146
	3	5.26e−1918	1.84e−2062
Execution time (s)		3.37	3.82

5. Numerical results

We will use the definition for the computational convergence order (COC)

$$\rho_q = \frac{\log\left(\|\mathbf{y}_{q+2} - \mathbf{y}^*\|_1 / \|\mathbf{y}_{q+1} - \mathbf{y}^*\|_1\right)}{\log\left(\|\mathbf{y}_{q+1} - \mathbf{y}^*\|_1 / \|\mathbf{y}_q - \mathbf{y}^*\|_1\right)}, \tag{24}$$

where $\mathbf{y}^*$ is the zero of function $\mathbf{F}(\cdot)$ and $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \ldots$ is the sequence of approximations given by the multi-step iterative method. In all tests, we will use the Chebyshev pseudospectral collocation method to approximate derivatives. It should be noted that, these approximations have high accuracy with respect to other numerical methods for smooth solutions of initial value problems [19], boundary value problems [20], integral equations [21], partial differential equations [22] and stochastic partial differential equations [23].

5.1. General systems of nonlinear equations

As a first example, we will first solve a small system of nonlinear equations. The results will confirm the claimed convergence order and accuracy of our method. The system of nonlinear equations is

$$\begin{aligned} &x_2x_3 + x_4(x_2 + x_3) = 0\\ &x_1x_3 + x_4(x_1 + x_3) = 0\\ &x_1x_2 + x_4(x_1 + x_2) = 0\\ &x_1x_2 + x_1x_3 + x_2x_3 - 1 = 0. \end{aligned} \tag{25}$$

Its solution up to an accuracy of 32 digits is

$$\begin{aligned} x_1 &= 0.57735026918962576450914878050 2\\ x_2 &= 0.57735026918962576450914878050 2\\ x_3 &= 0.57735026918962576450914878050 2\\ x_4 &= -0.28867513459481288225457439025 1. \end{aligned} \tag{26}$$

Table 5 gives the results obtained under FTUC and HJ. The accuracy obtained by HJ method is superior to that of FTUC, but the execution time of HJ is longer than that of FTUC. In Table 6, we give results obtained by approximately equating the execution times of both methods. Now, the accuracy obtained by FTUC is better, showing the superiority of FTUC over HJ.

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

Table 6
Comparison of performance between FTUC and HJ when both have approximately same execution time.

Iterative methods		FTUC	HJ
Number of iterations		3	1
Number of steps		7	7
Theoretical convergence order		17	14
Computational convergence order		17.1	14.1
Number of function evaluations per iteration		6	6
Solutions of system of linear equations per iteration		12	13
Number of matrix–vector multiplication per iteration		6	7
	Iteration		
$\|\mathbf{y}_q - \mathbf{y}^*\|_1$	1	1.03e−11	1.66e−10
	2	6.58e−199	2.51e−146
	3	1.27e−3400	1.84e−2062
Execution time (s)		3.71	3.82

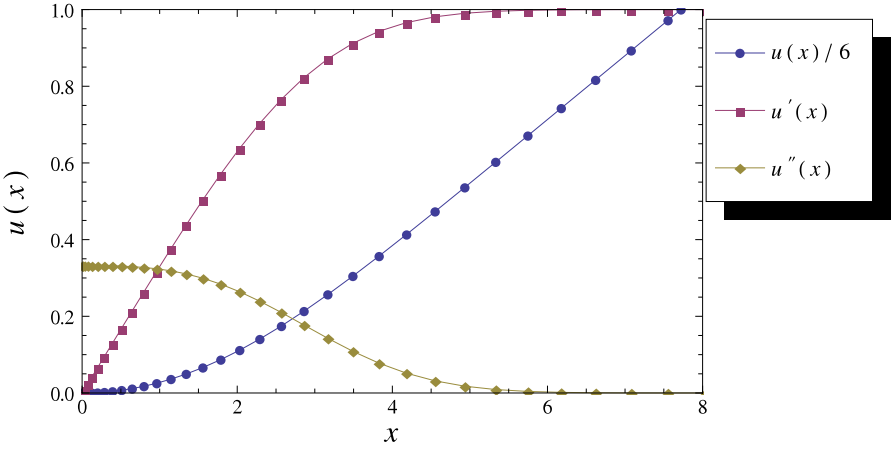


Fig. 1. Numerically calculated profile of $u(x)$, $u'(x)$ and $u''(x)$.

5.2. Classical Blasius flat-plate problem

The Classical Blasius flat-plate flow problem [24] is the boundary value problem

$$\begin{aligned} &u'''(x) + \frac{1}{2}u''(x)\,u(x) = 0 \\ &u(0) = u'(0) = 0, \qquad \left(u',\ u''\right) \longrightarrow (1,\ 0) \quad \text{as } x \longrightarrow \infty \\ &\mathbf{F}(\boldsymbol{u}) = D_x^3\boldsymbol{u} + \frac{1}{2}\boldsymbol{u}D_x^2\boldsymbol{u} \\ &\mathbf{F}'(\boldsymbol{u}) = D_x^3 + \frac{1}{2}\left(diag\left(D_x^2\boldsymbol{u}\right) + diag\,(\boldsymbol{u})\,D_x^2\right). \end{aligned} \tag{27}$$

We considered that problem for a domain for x [0, 200] with a mesh giving a problem size of 250. From a practical point of view, many researchers have been interested in computing $u''(0)$, and Howarth [24] reported $u''(0) = 0.332057$. We have computed $u''(0)$ with an accuracy of twenty five digits with the result

0.33205733628286351714556307. The FTUC iterative method produced numerical values of u' and u'' at $x = 200$ with accuracies of 3.87e−26 and 5.65e−25, respectively. The initial guess was

$$u_0(x) = \begin{cases} x^2 & \text{if } x \le 1 \\ x & \text{otherwise.} \end{cases} \tag{28}$$

Fig. 1 shows $u(x)$, $u'(x)$ and $u''(x)$. Fig. 2 plots $\log\left(\|\mathbf{F}(\mathbf{u})\|_1\right)$ versus the number of steps for the first three iterations. FTUC showed clear dominance over HJ. Table 7 gives the results obtained under FTUC and HJ with 30 steps.

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

Table 7
Performance of FTUC and HJ for the classical Blasius flat-plate problem for 30 steps.

Iterative methods		FTUC	HJ
Number of iterations		3	3
Number of steps		30	30
Theoretical convergence order		86	60
Number of function evaluations per iteration		29	29
Solutions of linear systems per iteration		58	59
Number of matrix–vector multiplication per iteration		29	30
	Iteration		
$\log\left(\|\mathbf{F}(\mathbf{u}_q)\|_1\right)$	1	4.63e−3	2.47e−3
	2	4.78e−8	8.605e−4
	3	4.98e−26	1.83e−7
Execution time (s)		125.42	158.82

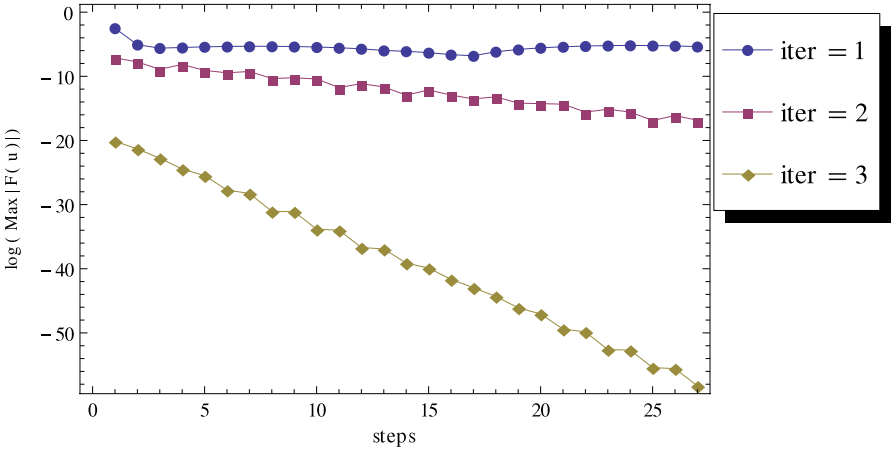


Fig. 2. Plot of $\log\left(\|\mathbf{F}(\mathbf{u})\|_1\right)$ versus the number steps, for the first three iterations.

5.3. Klein–Gordon equation

The Klein–Gordon equation [25] is the relativistic case of the Schrödinger equation

$$\begin{aligned} &u_{tt} - c^2 u_{xx} + f(u) = p \quad -\infty < x < \infty, \quad t > 0 \\ &\mathbf{F}(\boldsymbol{u}) = \left(D_t^2 - c^2 D_x^2\right)\boldsymbol{u} + f(\boldsymbol{u}) - \boldsymbol{p} \\ &\mathbf{F}' = D_t^2 - c^2 D_x^2 + diag\left(f'(\boldsymbol{u})\right), \end{aligned} \tag{29}$$

where $f(u)$ is an odd function of u and initial conditions are given by

$$\begin{aligned} &u(x, 0) = g_1(x) \\ &u_t(x, 0) = g_2(x). \end{aligned} \tag{30}$$

We chose $f(u) = ku - \gamma u^3$ and a domain $[-10, 10] \times [0, 1]$. We used Chebyshev pseudospectral collocation method for the discretization of space and time with 120 grid points for space and 30 grid points for time. The resulting problem size was 3600. Table 8 shows the results. The FTUC method clearly outperforms the HJ method. The exact solution can be written as

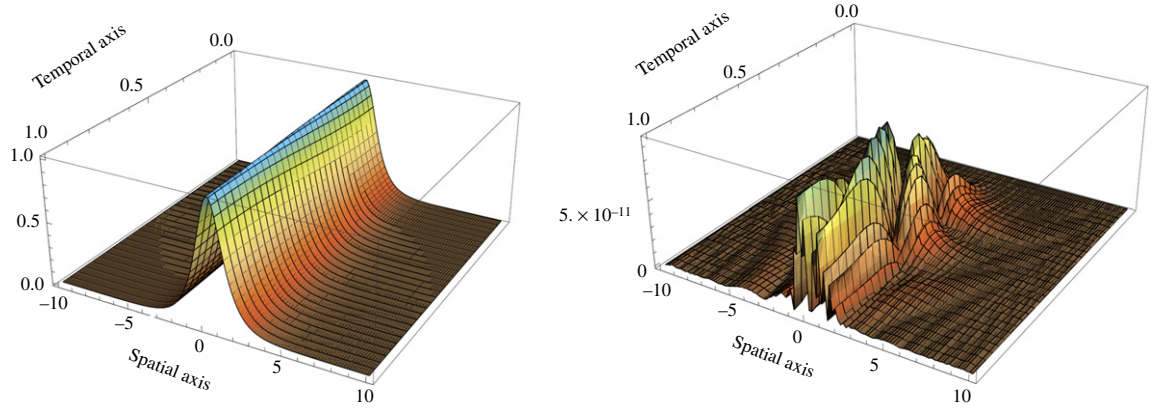
$$\begin{aligned} &\delta = \sqrt{\frac{2k}{\gamma}} \\ &\kappa = \sqrt{\frac{k}{c^2 - v^2}} \\ &u(x, t) = \delta \ \mathrm{sech}\left(\kappa\left(x - vt\right)\right), \end{aligned} \tag{31}$$

where $c = 1, \gamma = 1, v = 0.5$ and $k = 0.5$ (see Fig. 3).

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

Table 8
Performance of comparison between FTUC and HJ for Klein–Gordon problem.

Iterative methods		FTUC	HJ
Number of iterations		1	1
Number of steps		9	10
Theoretical convergence order		23	20
Number of function evaluations per iteration		8	9
Solutions of linear systems per iteration		16	19
Number of matrix–vector multiplication per iteration		8	10
	Iteration		
$\|\boldsymbol{u}_q - \boldsymbol{u}^*\|_1$	1	4.08e−1	5.99e−1
	2	5.67e−1	4.15e−2
	3	6.45e−2	2.34e−3
	4	3.62e−3	6.72e−5
	5	1.06e−4	1.15e−6
	6	1.84e−6	1.31e−8
	9	9.63e−11	1.24e−10
	10		9.12e−11
Execution time (s)		4.91	5.34



(a) Numerically calculated solution.

(b) Absolute error plot.

Fig. 3. Klein–Gordon equation, domain $= [-10,\ 10] \times [0,\ 1]$, grid points in spatial dimension $= 120$, grid points in temporal dimension $= 30$.

5.4. Two-dimensional sinh-Poisson equation

The stationary two-dimensional Euler flow free of body forces satisfies [26]

$$\nabla^2\phi + \sigma \sinh(\phi) = 0, \quad \sigma > 0. \tag{32}$$

The analytical solution of (32) for $\sigma = 1$ is given in [27].

$$\phi(x, y) = 4\tanh^{-1}\left[\frac{\beta\cos\left(\sqrt{1+\beta^2}x\right)}{\sqrt{1+\beta^2}\cosh(\beta y)}\right]. \tag{33}$$

That solution is called the Mallier–Maslowe vortices for $\sigma = 1$. We choose a domain $[-1.3, 1.3] \times [-1.3, 1.3]$ with 30 grid points for each dimension. That resulted in a problem size 500. The computed solution corresponds to $\beta = 0.5$. Table 9 shows the obtained results. The FTUC method achieves almost the same accuracy as the HJ method with a smaller number of iterations and, consequently, a smaller execution time. Fig. 4 depicts the numerically calculated solution and the absolute error over the spatial grid.

5.5. Three-dimensional nonlinear Poisson equation

The numerical solution of nonlinear 3-D nonlinear Poisson equation is treated in [28]. The governing equation is

$$\nabla \cdot (K(u)\nabla u) - g = 0, \tag{34}$$

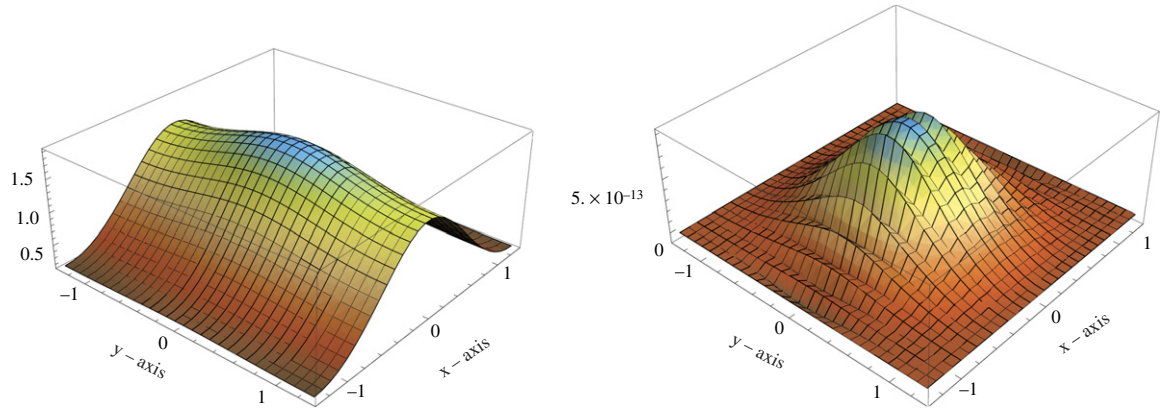
where $K(u)$ can be any function of u and g is a force term. We will adopt the expression for $K(u)$ from [28]

$$K(u) = \frac{100 + 27u}{300 + 27u}. \tag{35}$$

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

Table 9
Performance of FTUC and HJ for the sinh-Poisson equation.

Iterative methods	FTUC	HJ
Number of iterations	1	1
Number of steps	111	183
Theoretical convergence order	329	366
Number of function evaluations per iteration	110	182
Solutions of linear systems per iteration	220	365
Number of matrix–vector multiplication per iteration	110	183
$\|\boldsymbol{\phi}_q - \boldsymbol{\phi}^*\|_1$	9.22e−13	8.81e−13
Execution time (s)	42.87	68.225



(a) Solution. (b) Absolute error.

Fig. 4. Solution of sinh-Poisson equation and absolute error in the computed solution.

Table 10
Performance of FTUC and HJ for nonlinear Poisson equation.

Iterative methods	FTUC	HJ
Number of iterations	1	1
Number of steps	13	16
Theoretical convergence order	35	32
Number of function evaluations per iteration	12	15
Solutions of linear systems per iteration	24	31
Number of matrix–vector multiplication per iteration	12	16
$\|\boldsymbol{u}_q - \boldsymbol{u}^*\|_1$	5.77e−15	5.32e−15
Execution time (s)	10.608	11.646

We took a domain $[-1, 1]^3$ and constructed g in a way that makes $u = x^2 + y^2 + z^2$ the solution of (34). The grid points were taken so that the problem size is 1331. Table 10 gives the obtained results. FTUC achieves a similar accuracy as HJ with smaller number of steps and a smaller execution time. Fig. 5 plots the maximum absolute error in the solution vector $\mathbf{u}(\mathbf{x})$ against the number of steps.

6. Conclusions

Multi-step iterative methods with a large number of steps are computationally efficient because the computational cost required to perform the additional steps is small. The increase of convergence order in the multi-step part depends on the base method. So, the design of the base method is crucial. All numerically conducted tests conclude that the proposed FTUC multi-step iterative method requires relatively less execution time to achieve same numerical accuracy than HJ.

Acknowledgments

The authors thank the referees for their valuable comments and helpful suggestions which led to the improved version of the paper. The first author is funded by the Spanish MICINN grant AYA2010-15685.

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012

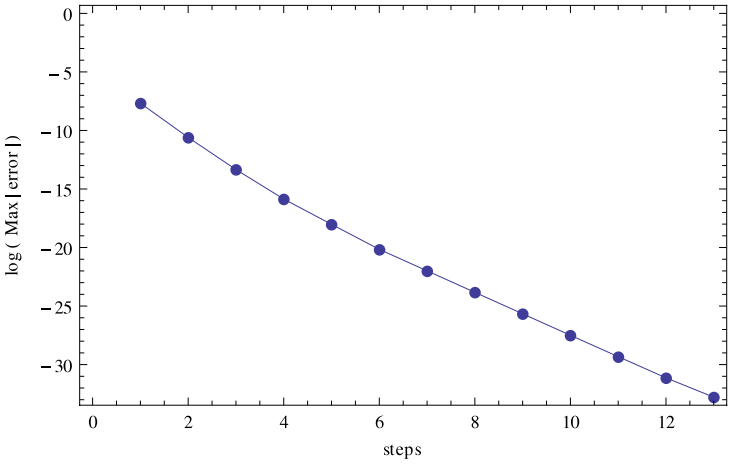


Fig. 5. $\log\left(\|\mathbf{u}(\mathbf{x}) - \mathbf{u}^*\|_1\right)$ as a function of the number of steps.

References

[1] M.Z. Ullah, F. Soleymani, A.S. Al-Fhaid, Numerical solution of nonlinear systems by a general class of iterative methods with application to nonlinear PDEs, Numer. Algorithms 67 (2014) 223–242.
[2] F. Soleymani, T. Lotfi, P. Bakhtiari, A multi-step class of iterative methods for nonlinear systems, Optim. Lett. 8 (2014) 1001–1015.
[3] H. Montazeri, F. Soleymani, S. Shateyi, S.S. Motsa, On a new method for computing the numerical solution of systems of nonlinear equations, J. Appl. Math. (2012) 15. Article ID 751975.
[4] J.R. Sharma, H. Arora, Efficient Jarratt-like methods for solving systems of nonlinear equations, Calcolo 51 (2014) 193–210.
[5] A. Cordero, J.L. Hueso, E. Martínez, J.R. Torregrosa, A modified Newton–Jarratt's composition, Numer. Algorithms 55 (2010) 87–99.
[6] M.Z. Ullah, S. Serra-Capizzano, F. Ahmad, An efficient multi-step iterative method for computing the numerical solution of systems of nonlinear equations associated with ODEs, Appl. Math. Comput. 250 (2015) 249–259.
[7] J.F. Traub, Iterative Methods for the Solution of Equations, Prentice-Hall, Englewood Cliffs, 1964.
[8] J.M. Gutiérrez, M.A. Hernández, A family of Chebyshev–Halley type methods in Banach spaces, Bull. Aust. Math. Soc. 55 (1997) 113–130.
[9] M. Frontini, E. Sormani, Some variant of Newton's method with third-order convergence, Appl. Math. Comput. 140 (2003) 419–426.
[10] H.H. Homeier, A modified Newton method with cubic convergence: the multivariable case, J. Comput. Appl. Math. 169 (2004) 161–169.
[11] A. Cordero, J.R. Torregrosa, Variants of Newton's method using fifth-order quadrature formulas, Appl. Math. Comput. 190 (2007) 686–698.
[12] M. Grau-Sánchez, À. Grau, M. Noguera, Ostrowski type methods for solving systems of nonlinear equations, Appl. Math. Comput. 218 (2011) 2377–2385.
[13] J.R. Sharma, R.K. Guha, R. Sharma, An efficient fourth order weighted-Newton method for systems of nonlinear equations, Numer. Algorithms 62 (2013) 307–323.
[14] A.M. Ostrowski, Solution of Equations and Systems of Equations, Academic Press, New York, 1966.
[15] P. Jarratt, Some fourth order multipoint iterative methods for solving equations, Math. Comp. 20 (1966) 434–437.
[16] C.T. Kelley, Solving Nonlinear Equations with Newton's Method, SIAM, Philadelphia, 2003.
[17] E.S. Alaidarous, M.Z. Ullah, F. Ahmad, A.S. Al-Fhaid, An efficient higher-order quasilinearization method for solving nonlinear BVPs, J. Appl. Math. (2013) 11. Article ID 259371.
[18] M.S. Petkovic, On a general class of multipoint root-finding methods of high computational efficiency, SIAM J. Numer. Anal. 49 (2011) 1317–1319.
[19] E. Tohidi, Kh. Erfani, M. Gachpazan, S. Shateyi, A new tau method for solving nonlinear Lane–Emden type equations via Bernoulli operational matrix of differentiation, J. Appl. Math. (2013) http://dx.doi.org/10.1155/2013/850170. Article ID 850170.
[20] E. Tohidi, A. Kilicman, A collocation method based on the Bernoulli operational matrix for solving nonlinear BVPs which arise from the problems in calculus of variation, Math. Probl. Eng. (2013) http://dx.doi.org/10.1155/2013/757206. Article ID 757206.
[21] T. Tang, X. Xu, J. Cheng, On spectral methods for Volterra integral equations and the convergence analysis, J. Comput. Math. 26 (2008) 825–837.
[22] J. Hesthaven, S. Gottlieb, D. Gottlieb, Spectral Methods for Time-Dependent Problems, Cambridge University Press, Cambridge, 2007.
[23] M. Dehghan, M. Shirzadi, Meshless simulation of stochastic advection–diffusion equations based on radial basis functions, Eng. Anal. Bound. Elem., http://dx.doi.org/10.1016/j.enganabound.2014.11.011.
[24] L. Howarth, On the solution of the laminar boundary layer equations, Proc. R. Soc. Lond. Ser. A 164 (1938) 547–579.
[25] T.S. Jang, An integral equation formalism for solving the nonlinear Klein–Gordon equation, Appl. Math. Comput. 243 (2014) 322–338.
[26] D. Gurarie, K.W. Chow, Vortex arrays for sinh-Poisson equation of two-dimensional fluids: Equilibria and stability, Phys. Fluids 16 (2004) 165–178.
[27] K.W. Chow, S.C. Tsang, C.C. Mak, Another exact solution for two-dimensional, inviscid sinh Poisson vortex arrays, Phys. Fluids 15 (2003) 264–281.
[28] M.B. Averick, J.M. Ortega, Fast solution of nonlinear Poisson-type equations, Argonne National Laboratory, August 1991.

Please cite this article in press as: F. Ahmad, et al., Higher order multi-step Jarratt-like method for solving systems of nonlinear equations: Application to PDEs and ODEs, Computers and Mathematics with Applications (2015), http://dx.doi.org/10.1016/j.camwa.2015.05.012