Fixed Point Theory and Algorithms for Sciences and Engineering

RESEARCH Open Access

A generalized multivariable Newton method

Regina S. Burachik[1]*, Bethany I. Caldwell[1] and C. Yalçın Kaya[1]

*Correspondence:
regina.burachik@unisa.edu.au
[1]Mathematics, UniSA STEM, University of South Australia, Mawson Lakes, S.A. 5095, Australia

Abstract

It is well known that the Newton method may not converge when the initial guess does not belong to a specific quadratic convergence region. We propose a family of new variants of the Newton method with the potential advantage of having a larger convergence region as well as more desirable properties near a solution. We prove quadratic convergence of the new family, and provide specific bounds for the asymptotic error constant. We illustrate the advantages of the new methods by means of test problems, including two and six variable polynomial systems, as well as a challenging signal processing example. We present a numerical experimental methodology which uses a large number of randomized initial guesses for a number of methods from the new family, in turn providing advice as to which of the methods employed is preferable to use in a particular search domain.

MSC: 49M15; 65H04; 65H10

Keywords: Fixed-point theory; Fixed-point algorithms; Newton's method; Nonlinear systems of equations; Polynomial equations

1 Introduction

Newton's method and its variants are a fundamental tool for solving nonlinear equations. Namely, given a $\mathcal{C}^1$-function $f:\mathbb{R}^n \to \mathbb{R}^n$, Newton's method is designed to converge iteratively to a solution of the problem

$$f(x) = \mathbf{0}, \quad x \in \mathbb{R}^n. \tag{1}$$

Problem (1) arises in practically every pure and applied discipline, including mathematical programming, engineering, physics, health sciences and economics. As a result, studies of Newton's method form an extremely active area of research, with new variants being constantly developed and tested. Basic results on Newton's method and comprehensive lists of references can be found, e.g., in the books by Dennis and Schnabel [4], Ostrowski [13], Ortega and Rheinboldt [12], Deuflhard [5] and Corless and Fillion [3]. The interested reader will find an excellent survey of Newton's method in [14].

When started at an initial guess close to a solution, Newton's method is well defined and converges quadratically to a solution of (1), unless the Jacobian of f is singular or the second partial derivatives of f are not bounded. Hence, if the user has an idea of where a solution might be lying, Newton's method is well known to be the fastest and most effective method for solving (1).

To ensure *global convergence* (i.e., to ensure convergence to a solution from any initial point), suitable modifications of the Newton method are needed. An example of a globally convergent variant is the so-called *Levenberg–Marquardt method* [8, 9]. This involves a modification of Newton's search direction at each step of the method. Without this modification, however, quadratic convergence can only be ensured when the initial guess belongs to a *quadratic convergence region*, namely a region from which every starting point generates a quadratically convergent Newton sequence.

For a given solution x^* of (1), a convergence region is, in general, not known *a priori*. Kantorovich [7] and Smale [16] establish convergence to a solution of (1) under suitable assumptions on the initial guess, and they do so without modifying Newton's original search direction. Our aim, on the other hand, is to devise a suitable modification of the Newton iteration, so that the quadratic convergence region is (ideally) larger than the one resulting from the classical Newton iterations. Our approach can, in some sense, be seen as a *preconditioning* of the iterations, so that a larger convergence region is obtained. This preconditioning might be helpful when very little is known about the location of the solutions of (1).

In [2], the authors presented a generalized version of the classical univariate Newton iteration in which the original Problem (1) is replaced by a "modified" system (for $n = 1$)

$$f \circ s^{-1}(x) = 0, \tag{2}$$

where $s : \mathbb{R} \to \mathbb{R}$ is a $\mathcal{C}^1$-invertible function in a neighbourhood of the solution. The classical Newton method then corresponds to the choice $s(x) = x$. By judiciously choosing s in a way that relates to the nature of Problem (1), the reference [2] illustrates via numerical experiments, that the region of quadratic convergence can be enlarged, and hence a wider choice of initial guesses are likely to result in quadratic convergence.

In the present paper, we propose a multivariate version of the generalized Newton method proposed in [2]. We establish the quadratic convergence under suitable assumptions, and test this new method in our numerical experiments. For suitable choices of s, we illustrate via extensive numerical experiments that the region of convergence corresponding to the new method may be larger than the one observed for the classical Newton iteration.

Recall that, if a sequence (x^k) converges to x^* (with $x^k \neq x^*, \forall k$), it is said to converge quadratically to x^* whenever we have

$$\lambda := \lim_{k\to\infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^2} < \infty,$$

where λ denotes the so-called *asymptotic error constant* (see Definition 2.5). Moreover, the smaller λ is, the faster the convergence will be.

With different choices of s, the value of λ will also be different, in general. Our generalized Newton methods provide a tool for enlarging the region where $\lambda = \lambda(s)$ is finite. Moreover, a suitable choice of s might produce a smaller value $\lambda(s)$, thus resulting in an improvement of the convergence speed. We illustrate this phenomenon in Sect. 4.

The choice of a suitable function s is, however, not clear in general, and more studies are needed to develop a systematic way of designing such choices. An appropriate choice of

s may result in a more robust behaviour of the generalized Newton method over a larger search domain, as can be observed in the numerical experiments we carry out in Sect. 5.

The present paper is organized as follows. In Sect. 2, we state the basic definitions, useful remarks, and properties. In Sect. 3, we establish the quadratic convergence results for the generalized Newton's method. In Sect. 4, we establish bounds on the asymptotic error constant. In Sect. 5, we test and compare the classical and a number of generalized methods for example problems with two and six variables, one of the problems being a challenging signal processing example. In the last section, the conclusion and a discussion are presented.

2 Preliminaries

We present first the main definitions and assumptions.

Definition 2.1 Let $x \in \mathbb{R}^n$ and let $g : \mathbb{R}^n \to \mathbb{R}^n$ be twice continuously differentiable. We write

$$g(x) = \begin{bmatrix} g_1(x) \\ \vdots \\ g_n(x) \end{bmatrix} \in \mathbb{R}^n \quad \text{and, for each } i = 1, \ldots, n, \text{we have} \quad \nabla g_i(x) := \begin{bmatrix} \frac{\partial g_i}{\partial x_1}(x) \\ \vdots \\ \frac{\partial g_i}{\partial x_n}(x) \end{bmatrix} \in \mathbb{R}^n,$$

where the vector $\nabla g_i(x)$ is called the *gradient of* g_i *at* x for every $i = 1, \ldots, n$. The *Jacobian of* g *at* x, denoted by $J_g(x)$, is the $n \times n$ matrix which has for row i the (transpose of the) gradient of each g_i, for $i = 1, \ldots, n$. More precisely, $J_g(x)$ is defined as

$$J_g(x) := \begin{bmatrix} \frac{\partial g_1}{\partial x_1}(x) & \frac{\partial g_1}{\partial x_2}(x) & \cdots & \frac{\partial g_1}{\partial x_n}(x) \\ \vdots & \vdots & & \vdots \\ \frac{\partial g_n}{\partial x_1}(x) & \frac{\partial g_n}{\partial x_2}(x) & \cdots & \frac{\partial g_n}{\partial x_n}(x) \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

For each $i = 1, \ldots, n$, the *Hessian of* g_i *at* x is denoted by $\nabla^2 g_i(x)$ and defined as

$$\nabla^2 g_i(x) := \begin{bmatrix} \frac{\partial^2 g_i}{\partial x_1^2}(x) & \frac{\partial^2 g_i}{\partial x_1 \partial x_2}(x) & \cdots & \frac{\partial^2 g_i}{\partial x_1 \partial x_n}(x) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 g_i}{\partial x_n \partial x_1}(x) & \frac{\partial^2 g_i}{\partial x_n \partial x_2}(x) & \cdots & \frac{\partial^2 g_i}{\partial x_n^2}(x) \end{bmatrix} \in \mathbb{R}^{n \times n},$$

for $i = 1, \ldots, n$.

In what follows, we denote by $\|\cdot\|$ the Euclidean norm, i.e., the ℓ_2-norm in $\mathbb{R}^n$. Denote by $B(x, r) := \{z \in \mathbb{R}^n : \|z - x\| < r\}$ the open ball centered at x with radius r. Similarly, denote by $B[x, r] := \{z \in \mathbb{R}^n : \|z - x\| \leq r\}$ the closed ball centered at x with radius r. Given a matrix $A \in \mathbb{R}^{n \times n}$, recall that a *norm of* A, denoted as $\|A\|$, can be given by $\|A\| := \max\{\|Ax\| : \|x\| \leq 1\}$, often referred to as the ℓ_2-norm of A.

The following simple lemma will be used in the proof of Proposition 4.1.

Lemma 2.1 *Let $A_1, \dots, A_n \in \mathbb{R}^{n\times n}$ and fix $u \in \mathbb{R}^n$. Consider the vectors*

$$v := \begin{bmatrix} u^T A_1 u \\ \vdots \\ u^T A_n u \end{bmatrix} \in \mathbb{R}^n, \quad \text{and} \quad w := \begin{bmatrix} \|A_1\| \\ \vdots \\ \|A_n\| \end{bmatrix} \in \mathbb{R}^n,$$

where $\|A_i\|$ is the ℓ_2-norm of the matrix A_i for $i = 1, \dots, n$. Then $\|v\| \leq \|u\|^2 \|w\|$.

Proof By Cauchy–Schwartz and the definition of the norm, $u^T A_i u \leq \|u\|^2 \|A_i\|$. Hence,

$$\begin{aligned} \|v\|^2 &= \sum_{i=1}^{n} \left(u^T A_i u\right)^2 \leq \sum_{i=1}^{n} \left(\|u\|^2 \|A_i\|\right)^2 \\ &= \|u\|^4 \sum_{i=1}^{n} \|A_i\|^2 = \|u\|^4 \|w\|^2, \end{aligned}$$

which yields the statement. □

We recall next some standard definitions and notation we will use in our analysis.

Definition 2.2 Let $A \subset \mathbb{R}^n$ and $B \subset \mathbb{R}^m$. Let $h : A \to B$ and fix $D \subset A$.

(a) We say that $h \in \mathcal{C}^0(D)$ if h is continuous at every $x \in D$.

(b) We write $h \in \mathcal{C}^1(D)$ if h is continuously differentiable at every $x \in D$. Equivalently, $J_h(\cdot) : D \to \mathbb{R}^{m\times n}$ is a continuous function of x, for every $x \in D$.

(c) Assume that $m = n$ and $h : A \to B$ is bijective. So there exists $h^{-1} : B \to A$ with $B = h(A)$. If $h \in \mathcal{C}^0(A)$ and $h^{-1} \in \mathcal{C}^0(B)$, we say that h is a *homeomorphism from A to B*. Moreover, if $h \in \mathcal{C}^1(A)$ and $h^{-1} \in \mathcal{C}^1(B)$, we say that h is a *$\mathcal{C}^1$-homeomorphism from A to B*.

(d) Fix $\beta > 0$. We say that $h \in \mathrm{Lip}_\beta(D)$ if we have

$$\left\| h(x) - h(x') \right\| \leq \beta \left\| x - x' \right\|,$$

for every $x, x' \in D$.

Remark 2.1 Let $D \subset \mathbb{R}^n$ be an open set and let $s : \mathbb{R}^n \to \mathbb{R}^n$ be a $\mathcal{C}^1$-homeomorphism from D to $s(D)$. Then, for every $x \in D$, we see that $J_s(x) \in \mathbb{R}^{n\times n}$ is invertible and

$$\left[J_s(x)\right]^{-1} = J_{s^{-1}}\left(s(x)\right).$$

Indeed, given $x \in D$, there is a unique $y \in s(D)$ such that $y = s(x)$. Differentiate both sides of the equality $s \circ s^{-1}(y) = y$ to derive

$$I = J_s\left(s^{-1}(y)\right) J_{s^{-1}}(y) = J_s(x) J_{s^{-1}}\left(s(x)\right), \tag{3}$$

where we used the chain rule. Now (3) directly yields the claim.

Remark 2.2 Let $D \subset \mathbb{R}^n$ be an open set and consider two functions $s,f : \mathbb{R}^n \to \mathbb{R}^n$ such that s is a $\mathcal{C}^1$-homeomorphism from D to $s(D)$, and that $f \in \mathcal{C}^1(D)$. Define $F := f \circ s^{-1} : s(D) \to f(D)$. By the chain rule and Remark 2.1, we have, for every $x \in D$,

$$J_F\bigl(s(x)\bigr) = J_f(x)J_{s^{-1}}\bigl(s(x)\bigr) = J_f(x)\bigl[J_s(x)\bigr]^{-1}. \tag{4}$$

In particular, if $J_f(x)$ is invertible, we obtain

$$\bigl[J_F\bigl(s(x)\bigr)\bigr]^{-1} = J_s(x)\bigl[J_f(x)\bigr]^{-1},$$

for every $x \in D$.

Let $g : \mathbb{R}^n \to \mathbb{R}^n$, so $g(x) := (g_1(x), \dots, g_n(x))^T$. For each $i = 1, \dots, n$, the gradient and Hessian of g_i collect the first- and second-order information, respectively, of g_i. The Jacobian, on the other hand, collects in a single operator all the first-order information for all coordinates of g. Similarly, we will need an operator that encapsulates all second-order information for all coordinates of g. We formally introduce these operators next.

Definition 2.3 Let $g : \mathbb{R}^n \to \mathbb{R}^n$ be twice continuously differentiable. With the notation of Definition 2.1, define the function $T_g : (\mathbb{R}^n)^n \to \mathbb{R}^{n\times(n\times n)}$ as

$$T_g\bigl(z^1, \dots, z^n\bigr) := \begin{bmatrix} \nabla^2 g_1(z^1) \\ \vdots \\ \nabla^2 g_n(z^n) \end{bmatrix},$$

where $z^j \in \mathbb{R}^n$ for $j = 1, \dots, n$. Given n vectors $z^1, \dots, z^n \in \mathbb{R}^n$, define the map $T_g(z^1, \dots, z^n) : (\mathbb{R}^n)^n \to \mathbb{R}^n$ as

$$T_g\bigl(z^1, \dots, z^n\bigr)_{(u^1,\dots,u^n)} := \begin{bmatrix} (u^1)^T \nabla^2 g_1(z^1) u^1 \\ \vdots \\ (u^n)^T \nabla^2 g_n(z^n) u^n \end{bmatrix} \in \mathbb{R}^n,$$

where $u^j \in \mathbb{R}^n$ for $j = 1, \dots, n$. Finally, define the following norm-like concept for the map $T_g(z^1, \dots, z^n)$:

$$\bigl\| T_g\bigl(z^1, \dots, z^n\bigr) \bigr\| := \sqrt{\sum_{i=1}^{n} \bigl\| \nabla^2 g_i(z^i) \bigr\|^2},$$

where the norms in the right-hand side are the ℓ_2-norms of the Hessians of the g_i. When $z^i = z^j = z$ for every $i,j \in \{1, \dots, n\}$, we use the short-hand notation

$$T_g(z, \dots, z) =: T_g(z).$$

Definition 2.4 Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$, denote its set of eigenvalues by $\Sigma(A)$. Recall that, when the matrix norm is induced by the ℓ_2-norm we have that $\|A\| = \max\{|\lambda| : \lambda \in \Sigma(A)\} =: \mathrm{SR}(A)$, the *spectral radius of* A. Denote by $\lambda_{\min}(A)$ the minimum eigenvalue of A, and by $\lambda_{\max}(A)$ the maximum eigenvalue of A.

Remark 2.3 Definition 2.4 and the fact that $\mathrm{SR}(\nabla^2 g_i(z^i)) = \|\nabla^2 g_i(z^i)\|$ for $i = 1,\dots,n$ directly yields

$$\|T_g(z^1,\dots,z^n)\| = \sqrt{\sum_{i=1}^{n}\left(\mathrm{SR}\left(\nabla^2 g_i(z^i)\right)\right)^2}.$$

For future use, we now give an elementary fact.

Fact 2.1 *Assume that $a \le b$ and $q \in [a,b]$. Denote by $c := \max\{|a|,|b|\}$. The following hold.*
(i) *If $a \ge 0$ then $a^2 \le q^2 \le b^2$.*
(ii) *If $b \le 0$ then $b^2 \le q^2 \le a^2$.*
(iii) *If $a < 0 < b$ then $0 \le q^2 \le c^2$.*

The concepts of rate of convergence and the asymptotic error constant will have an important role in our analysis, so we recall their definitions next.

Definition 2.5 Consider a method that generates a sequence $(x^k) \subseteq \mathbb{R}^n$ such that the sequence converges to x^*, where $x^k \neq x^*, \forall k$. If $\alpha > 0$ and $\lambda > 0$ with

$$\lim_{k\to\infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^\alpha} = \lambda,$$

then (x^k) is said to *converge to x^* with order α* and *asymptotic error constant λ*. When $\alpha = 2$, we say that the method converges *quadratically*.

2.1 Main assumptions

The following are our main assumptions for establishing quadratic convergence. We follow the analysis from [4].

(H_0) Problem (1) has a solution, denoted by x^*. There exists $r > 0$ such that $f \in \mathcal{C}^1(B(x^*,r))$. Denote $D_1 := B(x^*,r)$ throughout.

(H_1) $J_f(x)$ is nonsingular for all $x \in D_1$ and there exists $\gamma_1 > 0$ such that

$$\|J_f(x)^{-1}\| \le \gamma_1, \quad \text{for all } x \in D_1. \tag{5}$$

(H_2) The function s is a $\mathcal{C}^1$-homeomorphism from D_1 to $s(D_1)$ and there exists $\gamma_2 > 0$ such that

$$\|J_s(x)\| \le \gamma_2, \quad \text{for all } x \in D_1. \tag{6}$$

(H_3) Denote $D_2 := s(D_1)$. There exist $\beta_1, \beta_2 > 0$ such that $J_f \in \mathrm{Lip}_{\beta_1}(D_1)$ and $J_{s^{-1}} \in \mathrm{Lip}_{\beta_2}(D_2)$.

Given a set $A \subset \mathbb{R}^n$, we denote by $\mathrm{cl}(A)$ the closure of the set A.

Remark 2.4 Assumption (H_3) implies the existence of $M_1, M_2 > 0$ such that

$$\|J_f(x)\| \le M_1, \quad \text{and} \quad \|J_{s^{-1}}(s(x))\| = \left\|[J_s(x)]^{-1}\right\| \le M_2 \quad \text{for all } x \in D_1.$$

Indeed, this follows from the fact that the mappings are continuous over the compact sets $\mathrm{cl}(D_1)$ and $\mathrm{cl}(D_2)$, respectively.

Remark 2.5 Assumption (H_3) allows us to apply Lemma 4.1.16 in [4] to deduce that there exist $\epsilon > 0$, $L_0, L_1 > 0$ such that

$$L_1 \| z - z' \| \le \| s^{-1}(z) - s^{-1}(z') \| \le L_0 \| z - z' \|$$

for all $z, z' \in B(s(x^*), \epsilon) \cap s(D_1)$. Setting $z = s(x)$, $z' = s(x')$ this implies

$$L_1 \| s(x) - s(x') \| \le \| x - x' \| \le L_0 \| s(x) - s(x') \|.$$

The authors of [2] proposed a generalized Newton method for solving (1) for $n = 1$. This method can be described by the following iterative formula:

$$g(x^k) = x^{k+1} = s^{-1}\left(s(x^k) - s'(x^k) \frac{f(x^k)}{f'(x^k)} \right), \tag{7}$$

where $s : \mathbb{R} \to \mathbb{R}$ is a $\mathcal{C}^1$-invertible function in a neighbourhood of the solution. As mentioned in the Introduction, this modification can be seen as a preconditioning of Problem (1), in which the choice of a suitable function s can improve/enlarge the region of convergence of the method. Next we extend the above approach in a natural way to higher dimensions. Namely, for a $\mathcal{C}^1$-function $f : \mathbb{R}^n \to \mathbb{R}^n$, consider the problem of solving

$$f(x) = 0, \quad x \in \mathbb{R}^n. \tag{8}$$

In order to solve Problem (8), we replace the derivatives in (7) by the corresponding Jacobians. More precisely, consider the function $g : \mathbb{R}^n \to \mathbb{R}^n$ defined by

$$g(x) := s^{-1}\big(s(x) - J_s(x) [J_f(x)]^{-1} f(x) \big), \tag{9}$$

where $s : \mathbb{R}^n \to \mathbb{R}^n$ has an inverse s^{-1}, and $J_s(x)$, $J_f(x)$ are the (assumed nonsingular) Jacobians of s and f at x, respectively. It can be directly checked that a fixed point x^* of g is a solution of (8), as long as both Jacobians are invertible at x^*. The generalized Newton iteration is obtained by the rule $g(x^k) = x^{k+1}$, where g is the function defined in (9).

Definition 2.6 Assume that (H_0)–(H_2) hold. Given $x^k \in D_1$, define

$$x^{k+1} := s^{-1}\big(s(x^k) - J_s(x^k) [J_f(x^k)]^{-1} f(x^k) \big). \tag{10}$$

In the next section we extend the standard quadratic convergence results to the sequence defined by (10).

3 Convergence of the multivariate generalized Newton method

The following is Lemma 4.1 from [2] rewritten for the multivariate case. This lemma states that the iteration (10) coincides with the classical Newton iteration for the composite function $F := f \circ s^{-1}$.

Lemma 3.1 *With the notation and hypothesis of Definition 2.6, let $y^k = s(x^k)$ and $F := f \circ s^{-1}$. Iteration (10) can be written as*

$$y^{k+1} = y^k - \left[J_F\left(y^k\right)\right]^{-1} F\left(y^k\right). \tag{11}$$

Proof Using the definitions of y^k and y^{k+1}, and Remark 2.2, we can write the iteration in (11) as

$$s\left(x^{k+1}\right) = s\left(x^k\right) - J_s\left(x^k\right)\left[J_f\left(x^k\right)\right]^{-1} f\left(x^k\right).$$

Now applying s^{-1} to this equality yields Iteration (10). □

We will use [4, Theorem 5.2.1] which we quote next.

Theorem 3.1 *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be such that $F \in \mathcal{C}^1(D)$, for an open convex set $D \subseteq \mathbb{R}^n$. Assume that there exists $\tilde{x} \in \mathbb{R}^n$ such that $F(\tilde{x}) = 0$, and $\delta, \beta, \gamma > 0$ such that the following hold.*

(i) *$B(\tilde{x}, \delta) \subseteq D$, with $J_F(\tilde{x})$ invertible and $\|[J_F(\tilde{x})]^{-1}\| \le \beta$.*
(ii) *$J_F \in \operatorname{Lip}_\gamma(B(\tilde{x}, \delta))$ (i.e., for all $x, y \in B(\tilde{x}, \delta)$, $\|J_F(x) - J_F(y)\| \le \gamma \|x - y\|$).*

Then there exists $\epsilon > 0$ such that, for all $\tilde{x}_0 \in B(\tilde{x}, \epsilon)$, the sequence $(\tilde{x}^k)$ generated by the rule

$$\tilde{x}^{k+1} = \tilde{x}^k - \left[J_F\left(\tilde{x}^k\right)\right]^{-1} F\left(\tilde{x}^k\right), \tag{12}$$

for all $k \ge 0$, has the following properties:

(a) *$(\tilde{x}^k)$ is well defined (i.e. $[J_F(\tilde{x}^k)]^{-1}$ exists for all $k \ge 0$).*
(b) *The convergence to $\tilde{x}$ is quadratic, namely,*

$$\left\|\tilde{x}^{k+1} - \tilde{x}\right\| \le \beta\gamma \left\|\tilde{x}^k - \tilde{x}\right\|^2,$$

for all $k \ge 0$.

Next we show that we can apply this theorem to our setting for a suitable choices of F, $\tilde{x}$ and D.

Lemma 3.2 *Assumptions (H_0)–(H_3) imply that Conditions* (i)–(ii) *in Theorem 3.1 hold for $F := f \circ s^{-1}$, $\tilde{x} := s(x^*)$ and $D := D_1$. Consequently, there exists $\varepsilon > 0$ such that, for $y^0 \in B(s(x^*), \varepsilon)$, the sequence*

$$y^{k+1} := y^k - \left[J_F\left(y^k\right)\right]^{-1} F\left(y^k\right),$$

for all $k \ge 0$, has the following properties:

(a) *(y^k) is well defined (i.e. $[J_F(y^k)]^{-1}$ exists for all $k \ge 0$).*
(b) *The convergence to $s(x^*)$ is quadratic, namely,*

$$\left\|y^{k+1} - s\left(x^*\right)\right\| \le \eta \left\|y^k - s\left(x^*\right)\right\|^2, \quad \forall k \ge 0,$$

with $\eta := \gamma_1\gamma_2 L_0(\frac{M_1\beta_2}{L_1} + M_2\beta_1)$.

Proof Note that the statements (a) and (b) involving the sequence (y^k) will follow directly from Theorem 3.1 once we establish Conditions (i)–(ii) for suitable constants. Therefore, we proceed to prove (i) and (ii). By (H_0) and the definitions of F and $\tilde{x}$, we can write

$$F(\tilde{x}) = f \circ s^{-1}\big(s(x^*)\big) = f\big(x^*\big) = 0.$$

By (H_2) and (H_3) we have $D_2 = s(D_1)$ is an open neighbourhood of $\tilde{x} = s(x^*)$. Hence we can take $\delta > 0$ such that $B(\tilde{x}, \delta) \subset D_2 = s(B(x^*, r))$. Using Remark 2.2 we can write

$$J_F(\tilde{x}) = J_{f\circ s^{-1}}\big(s(x^*)\big) = J_f\big(x^*\big)\big[J_s\big(x^*\big)\big]^{-1}. \tag{13}$$

By (H_1)–(H_2) and Remark 2.1, we deduce that the matrix on the right-hand side of (13) is nonsingular and hence $J_F(\tilde{x})$ is nonsingular. Using Remark 2.1 again gives

$$\big\|\big[J_F(\tilde{x})\big]^{-1}\big\| = \big\|J_s\big(x^*\big)\big[J_f\big(x^*\big)\big]^{-1}\big\| \le \big\|J_s\big(x^*\big)\big\|\big\|\big[J_f\big(x^*\big)\big]^{-1}\big\| \le \gamma_1\gamma_2, \tag{14}$$

where we used (H_1)–(H_2) in the last inequality. The expressions (13)–(14) imply that condition (i) in Theorem 3.1 holds for $\tilde{x} \in B(\tilde{x}, \delta)$ with $\beta := \gamma_1\gamma_2$.

Next, we check Condition (ii) in Theorem 3.1 for $J_F = J_{f\circ s^{-1}}$. Namely, we show now that there exists $\gamma > 0$ such that $J_{f\circ s^{-1}} \in \mathrm{Lip}_\gamma(B(\tilde{x}, \delta))$. By (H_2), given $z, z' \in D_2 = s(D_1)$ there exist unique $x, x' \in D_1 = B(x^*, r)$ such that $z = s(x)$, $z' = s(x')$. Adding and subtracting a suitable term and using Remark 2.2 we obtain

$$\begin{aligned}
&\big\|J_{f\circ s^{-1}}(z) - J_{f\circ s^{-1}}\big(z'\big)\big\| \\
&\quad = \big\|J_{f\circ s^{-1}}(z) - J_f(x)\big[J_s\big(x'\big)\big]^{-1} + J_f(x)\big[J_s\big(x'\big)\big]^{-1} - J_{f\circ s^{-1}}\big(z'\big)\big\| \\
&\quad \le \big\|J_f(x)\big(\big[J_s(x)\big]^{-1} - \big[J_s\big(x'\big)\big]^{-1}\big)\big\| + \big\|\big(J_f(x) - J_f\big(x'\big)\big)\big[J_s\big(x'\big)\big]^{-1}\big\| \\
&\quad \le \big\|J_f(x)\big\|\big\|J_{s^{-1}}\big(s(x)\big) - J_{s^{-1}}\big(s\big(x'\big)\big)\big\| + \big\|\big[J_s\big(x'\big)\big]^{-1}\big\|\big\|J_f(x) - J_f\big(x'\big)\big\|.
\end{aligned} \tag{15}$$

By (H_3) we know that $J_{s^{-1}} \in \mathrm{Lip}_{\beta_2}(D_2)$ and hence for every $x, x' \in D_1$ we have

$$\big\|J_{s^{-1}}\big(s(x)\big) - J_{s^{-1}}\big(s\big(x'\big)\big)\big\| \le \beta_2\big\|s(x) - s\big(x'\big)\big\| \le \frac{\beta_2}{L_1}\big\|x - x'\big\|, \tag{16}$$

where we have used Remark 2.5 in the last inequality. By (H_3) we also have $J_f \in \mathrm{Lip}_{\beta_1}(D_1)$, so

$$\big\|J_f(x) - J_f\big(x'\big)\big\| \le \beta_1\big\|x - x'\big\|, \tag{17}$$

for every $x, x' \in D_1$. Using (16)–(17) in (15), together with Remark 2.4 gives

$$\begin{aligned}
\big\|J_{f\circ s^{-1}}(z) - J_{f\circ s^{-1}}\big(z'\big)\big\| &\le \left(\frac{M_1\beta_2}{L_1} + M_2\beta_1\right)\big\|x - x'\big\| = \bar{M}\big\|s^{-1}(z) - s^{-1}\big(z'\big)\big\| \\
&\le \bar{M}L_0\big\|z - z'\big\|;
\end{aligned}$$

so $J_F \in \mathrm{Lip}_{\bar{M}L_0}(D_2)$, where $\bar{M} = \frac{M_1\beta_2}{L_1} + M_2\beta_1$. Hence Condition (ii) holds for $\gamma := \bar{M}L_0$. This completes the proof of conditions (i) and (ii). Using now Theorem 3.1 and the definitions of β and γ, we obtain (a) and (b) for the sequence (y^k) with the stated value of η. □

Theorem 3.2 *With the notation of Definition 2.6, assume that (H_0)–(H_3) hold. The sequence (x^k) given by the rule (10) is well defined and converges quadratically to x^*.*

Proof By Lemma 3.2, the sequence (y^k) with $y^k := s(x^k)$ is well defined and converges quadratically to $s(x^*)$. By Lemma 3.1, the iteration on (y^k) can be equivalently written as

$$s(x^{k+1}) = s(x^k) - J_s(x^k)\big[J_f(x^k)\big]^{-1} f(x^k).$$

We will show now that (x^k) converges quadratically to x^*. Indeed, by part (b) of Lemma 3.2, we have

$$\|y^{k+1} - s(x^*)\| \le \eta \|y^k - s(x^*)\|^2, \tag{18}$$

for η as in Lemma 3.2(b). We can write

$$\begin{aligned}\|x^{k+1} - x^*\| &= \big\|s^{-1}\big(s(x^k) - J_s(x^k)\big[J_f(x^k)\big]^{-1} f(x^k)\big) - x^*\big\| \\ &= \big\|s^{-1}\big(s(x^k) - J_s(x^k)\big[J_f(x^k)\big]^{-1} f(x^k)\big) - s^{-1}\big(s(x^*)\big)\big\|.\end{aligned}$$

Applying Remark 2.5 to bound the right-most expression, we obtain

$$\begin{aligned}\|x^{k+1} - x^*\| &\le L_0 \big\|s(x^k) - J_s(x^k)\big[J_f(x^k)\big]^{-1} f(x^k) - s(x^*)\big\| \\ &= L_0 \|y^{k+1} - s(x^*)\|,\end{aligned}$$

where we have used the definition of y^{k+1} in the equality. We can now use (18) in the above expression to derive

$$\|x^{k+1} - x^*\| \le L_0 \eta \|y^k - s(x^*)\|^2 = L_0 \eta \|s(x^k) - s(x^*)\|^2.$$

Applying again Remark 2.5 to bound the right-most expression, we obtain

$$\|x^{k+1} - x^*\| \le \frac{L_0 \eta}{L_1} \|x^k - x^*\|^2.$$

So (x^k) converges quadratically to x^*, as desired. □

4 Bounds on the asymptotic error constant λ

While the results of the previous section hold for the specific iteration (10), the following results are true for any fixed-point iteration of the form $g(x^k) = x^{k+1}$. In this section we will always assume that g is twice continuously differentiable.

Proposition 4.1 *Assume that $g(x^*) = x^*$ and that $J_g(x^*) = 0$. Consider the sequence (x^k) defined by the fixed point iteration $x^{k+1} = g(x^k)$. Then there exist sequences $(\xi_1^k), \dots, (\xi_n^k)$ converging to x^* such that the asymptotic error constant λ given by Definition 2.5 verifies*

$$\lambda = \frac{1}{2} \lim_{k\to\infty} \frac{\|T_g(\xi_1^k, \dots, \xi_n^k)_{(x^k - x^*, \dots, x^k - x^*)}\|}{\|x^k - x^*\|^2},$$

where T_g is as in Definition 2.3.

Proof Since $J_g(x^*) = 0$, it is well known that (x^k) converges quadratically to x^*. We begin by writing $g(x)$ into a Taylor polynomial around x^*, coordinate by coordinate,

$$g(x) = g(x^*) + J_g(x^*)(x - x^*) + \frac{1}{2}\begin{bmatrix}(x-x^*)^T\nabla^2 g_1(\xi_1)(x-x^*)\\ \vdots \\ (x-x^*)^T\nabla^2 g_n(\xi_n)(x-x^*)\end{bmatrix},$$

where $\xi_j, j = 1,\dots,n$, are between x and x^*. Using now Definition 2.3 as well as the equalities $g(x^*) = x^*$ and $J_g(x^*) = 0$, we obtain

$$g(x) = x^* + \frac{1}{2}T_g(\xi_1,\dots,\xi_n)_{(x-x^*,\dots,x-x^*)}.$$

By taking $x := x^k$ and using the definition of the fixed-point iteration, we obtain

$$g(x^k) = x^{k+1} = x^* + \frac{1}{2}T_g(\xi_1^k,\dots,\xi_n^k)_{(x^k-x^*,\dots,x^k-x^*)},$$

where $\xi_j^k, j = 1,\dots,n$, are between x^k and x^*. Re-arranging, taking norms, and then dividing by $\|x^k - x^*\|^2$ yield

$$\frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|^2} = \frac{\|T_g(\xi_1^k,\dots,\xi_n^k)_{(x^k-x^*,\dots,x^k-x^*)}\|}{2\|x^k-x^*\|^2}.$$

We know that the sequences (x^k) and (ξ_j^k) converge to x^* as $k \to \infty$ for $j = 1,\dots,n$. Using these facts and taking limits yield

$$\lambda = \lim_{k\to\infty}\frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|^2} = \frac{1}{2}\lim_{k\to\infty}\frac{\|T_g(\xi_1^k,\dots,\xi_n^k)_{(x^k-x^*,\dots,x^k-x^*)}\|}{\|x^k-x^*\|^2}, \tag{19}$$

where we have also invoked Definition 2.5. This proves the proposition. □

Our aim in this section is to use Proposition 4.1 to establish upper and lower bounds for the asymptotic error constant. For this we will need the following definition. For $j \in \{1,\dots,n\}$, we denote by $[v]_j$ the jth coordinate of the vector $v \in \mathbb{R}^n$.

Definition 4.1 Given n vectors $x^1,\dots,x^n$, and the matrices $\nabla^2 g_1(x^1),\dots,\nabla^2 g_n(x^n)$, define the vector $\mu(x^1,\dots,x^n) \in \mathbb{R}^n_+$ as

$$[\mu(x^1,\dots,x^n)]_j := \begin{cases} 0, & \text{if } \lambda_{\min}(\nabla^2 g_j(x^j)) < 0 < \lambda_{\max}(\nabla^2 g_j(x^j)),\\ \lambda_{\min}(\nabla^2 g_j(x^j)), & \text{if } \lambda_{\min}(\nabla^2 g_j(x^j)) \ge 0,\\ |\lambda_{\max}(\nabla^2 g_j(x^j))|, & \text{if } \lambda_{\max}(\nabla^2 g_j(x^j)) \le 0,\end{cases} \tag{20}$$

for $j = 1,\dots,n$. Define also the vector $\rho(x^1,\dots,x^n) \in \mathbb{R}^n_+$ as

$$[\rho(x^1,\dots,x^n)]_j := \mathrm{SR}(\nabla^2 g_j(x^j)) = \|\nabla^2 g_j(x^j)\|,$$

for $j = 1,\dots,n$. For simplicity in appearance, we will write $\mu(x)$ and $\rho(x)$ when $x^j = x$ for all $j = 1,\dots,n$. Namely,

$$\mu(x) := \mu(x,\dots,x) \quad \text{and} \quad \rho(x) := \rho(x,\dots,x).$$

Remark 4.1 With the notation of Definition 2.4, it is well known that the spectral radius $\mathrm{SR}(A)$ is a continuous function of the matrix A. Hence, if g is twice continuously differentiable in a neighbourhood around x, the function ρ will be a continuous function of x. Therefore, whenever $(\xi_1^k),\dots,(\xi_{n-1}^k)$ and (ξ_n^k) are sequences converging to the same point z, we will have $\rho(\xi_1^k,\dots,\xi_n^k)$ converging to $\rho(z,\dots,z) = \rho(z)$. A similar fact can be established for the function μ.

Remark 4.2 Clearly, the function $[\mu(\cdot)]_j$ can be equivalently defined as

$$\begin{aligned}&\bigl[\mu\bigl(x^1,\dots,x^n\bigr)\bigr]_j\\ &\quad= \begin{cases} 0, & \text{if } \lambda_{\min}(\nabla^2 g_j(x^j)) < 0 < \lambda_{\max}(\nabla^2 g_j(x^j)),\\ \min\{|\lambda_{\min}(\nabla^2 g_j(x^j))|, |\lambda_{\max}(\nabla^2 g_j(x^j))|\}, & \text{if } \begin{cases}\lambda_{\min}(\nabla^2 g_j(x^j)) \ge 0 \quad \text{or}\\ \lambda_{\max}(\nabla^2 g_j(x^j)) \le 0,\end{cases}\end{cases}\end{aligned}$$

for $j = 1,\dots,n$. So the definition of $[\mu(\cdot)]_j$ is given over two complementary sets, one of them open and the other closed. Since in each of these sets $[\mu(\cdot)]_j$ is given by a continuous function, and the values of these two functions coincide at the boundary of the two complementary sets, we deduce that μ is a continuous function.

The next technical result will be used in Theorem 4.1. Recall from Remark 2.3 and Definition 2.3 that

$$\bigl\| T_g\bigl(x^1,\dots,x^n\bigr)\bigr\| = \bigl\|\rho\bigl(x^1,\dots,x^n\bigr)\bigr\|. \tag{21}$$

Proposition 4.2 *With the notation of Definition 4.1, we have*

$$\bigl(\bigl[\mu\bigl(x^1,\dots,x^n\bigr)\bigr]_j\bigr)^2 \le \frac{(u^T\nabla^2 g_j(x^j)u)^2}{\|u\|^4} \le \bigl(\bigl[\rho\bigl(x^1,\dots,x^n\bigr)\bigr]_j\bigr)^2,$$

for $j = 1,\dots,n$ and all nonzero $u \in \mathbb{R}^n$.

Proof Recall that by Rayleigh quotient properties,

$$\lambda_{\min}\bigl(\nabla^2 g_j\bigl(x^j\bigr)\bigr) \le \frac{u^T\nabla^2 g_j(x^j)u}{\|u\|^2} \le \lambda_{\max}\bigl(\nabla^2 g_j\bigl(x^j\bigr)\bigr),$$

for all $j = 1,\dots,n$. So for each fixed j we can apply Fact 2.1 with

$$\begin{aligned} a &:= \lambda_{\min}\bigl(\nabla^2 g_j\bigl(x^j\bigr)\bigr), & b &:= \lambda_{\max}\bigl(\nabla^2 g_j\bigl(x^j\bigr)\bigr),\\ q &= \frac{u^T\nabla^2 g_j(x^j)u}{\|u\|^2}, & c &:= \max\bigl\{\bigl|\lambda_{\min}\bigl(\nabla^2 g_j\bigl(x^j\bigr)\bigr)\bigr|, \bigl|\lambda_{\max}\bigl(\nabla^2 g_j\bigl(x^j\bigr)\bigr)\bigr|\bigr\}.\end{aligned}$$

Assume that $a = \lambda_{\min}(\nabla^2 g_j(x^j)) \geq 0$. In this situation, by definition we have $[\mu(x^1,\dots,x^n)]_j = a = \lambda_{\min}(\nabla^2 g_j(x^j))$ and $[\rho(x^1,\dots,x^n)]_j = b = \lambda_{\max}(\nabla^2 g_j(x^j))$. Now part (i) of Fact 2.1 directly yields

$$\begin{aligned}\left(\left[\mu\left(x^1,\dots,x^n\right)\right]_j\right)^2 &= a^2 = \left(\lambda_{\min}\left(\nabla^2 g_j\left(x^j\right)\right)\right)^2 \leq q^2\\ &= \frac{(u^T\nabla^2 g_j(x^j)u)^2}{\|u\|^4} \leq b^2 = \left(\lambda_{\max}\left(\nabla^2 g_j\left(x^j\right)\right)\right)^2.\end{aligned}$$

If $b = \lambda_{\max}(\nabla^2 g_j(x^j)) \leq 0$ then $[\rho(x^1,\dots,x^n)]_j = |\lambda_{\min}(\nabla^2 g_j(x^j))|$ and by part (ii) of Fact 2.1 we have

$$\begin{aligned}\left(\left[\mu\left(x^1,\dots,x^n\right)\right]_j\right)^2 &= b^2 = \left(\left|\lambda_{\max}\left(\nabla^2 g_j\left(x^j\right)\right)\right|\right)^2\\ &\leq \frac{(u^T\nabla^2 g_j(x^j)u)^2}{\|u\|^4} = q^2 \leq \left(\left|\lambda_{\min}\left(\nabla^2 g_j\left(x^j\right)\right)\right|\right)^2\\ &= a^2 = \left(\left[\rho\left(x^1,\dots,x^n\right)\right]_j\right)^2.\end{aligned}$$

Finally, if $\lambda_{\min}(\nabla^2 g_j(x^j)) < 0 < \lambda_{\max}(\nabla^2 g_j(x^j))$ then by definition $[\mu(x^1,\dots,x^n)]_j = 0$ and $[\rho(x^1,\dots,x^n)]_j = \max\{|\lambda_{\min}(\nabla^2 g_j(x^j))|, |\lambda_{\max}(\nabla^2 g_j(x^j))|\} = c$. Using part (iii) of Fact 2.1 yields

$$\begin{aligned}0 = \left(\left[\mu\left(x^1,\dots,x^n\right)\right]_j\right)^2 &\leq \frac{(u^T\nabla^2 g_j(x^j)u)^2}{\|u\|^4} = q^2 \leq c^2\\ = \left(\max\left\{\left|\lambda_{\min}\left(\nabla^2 g_j\left(x^j\right)\right)\right|, \left|\lambda_{\max}\left(\nabla^2 g_j\left(x^j\right)\right)\right|\right\}\right)^2\\ = \left(\left[\rho\left(x^1,\dots,x^n\right)\right]_j\right)^2.\end{aligned}$$

This completes the proof. □

Theorem 4.1 *Suppose* $g(x^*) = x^*$ *and* $J_g(x^*) = 0$, *with g twice continuously differentiable in a neighbourhood of* x^*. *Consider the asymptotic error constant* $\lambda \geq 0$ *for the multivariate iterative method* $x^{k+1} = g(x^k)$. *Then*

$$\frac{1}{2}\left\|\mu\left(x^*\right)\right\| \leq \lambda \leq \frac{1}{2}\left\|T_g\left(x^*\right)\right\|,$$

where $\mu(x^*)$ *is as in Definition 4.1, and* T_g *as in Definition 2.3.*

Proof Recall that, by (21), $\|T_g(x^*)\| = \|\rho(x^*)\|$. Hence, it is enough to establish the upper bound with $\|\rho(x^*)\|$ instead of $\|T_g(x^*)\|$. By Proposition 4.1 we have

$$\lambda = \lim_{k\to\infty} \frac{\|T_g(\xi_1^k,\dots,\xi_n^k)_{(x^k-x^*,\dots,x^k-x^*)}\|}{2\|x^k - x^*\|^2},$$

where $(\xi_1^k),\dots,(\xi_n^k)$ are n sequences converging to x^*. Using now Proposition 4.2 for $x^j = \xi_j^k$ and $u = x^k - x^* \neq 0$ we deduce that

$$\left(\left[\mu\left(\xi_1^k,\dots,\xi_n^k\right)\right]_j\right)^2 \leq \frac{((x^k - x^*)^T\nabla^2 g_i(\xi_j^k)(x^k - x^*))^2}{\|(x^k - x^*)\|^4} \leq \left(\left[\rho\left(\xi_1^k,\dots,\xi_n^k\right)\right]_j\right)^2. \tag{22}$$

By definition of T_g and Lemma 2.1 we have

$$\left\| T_g\left(\xi_1^k,\dots,\xi_n^k\right)_{(x^k-x^*,\dots,x^k-x^*)} \right\| \le \left\| x^k - x^* \right\|^2 \left\| T_g\left(\xi_1^k,\dots,\xi_n^k\right) \right\|.$$

Combine this fact with (22) and Definition 2.3 to derive

$$\begin{aligned} \frac{1}{2}\lim_{k\to\infty} \left\| \mu\left(\xi_1^k,\dots,\xi_n^k\right) \right\| &\le \lim_{k\to\infty} \frac{\| T_g(\xi_1^k,\dots,\xi_n^k)_{(x^k-x^*,\dots,x^k-x^*)} \|}{2\|x^k-x^*\|^2} \\ &\le \frac{1}{2}\lim_{k\to\infty} \left\| T_g\left(\xi_1^k,\dots,\xi_n^k\right) \right\|. \end{aligned} \tag{23}$$

By Remarks 4.1, 4.2 and the fact that g is twice continuously differentiable, the functions T_g and μ are continuous, so

$$\lim_{k\to\infty} \mu\left(\xi_1^k,\dots,\xi_n^k\right) = \mu\left(x^*\right) \quad \text{and} \quad \lim_{k\to\infty} \left\| T_g\left(\xi_1^k,\dots,\xi_n^k\right) \right\| = \left\| T_g\left(x^*\right) \right\| = \left\| \rho\left(x^*\right) \right\|,$$

where we also used (21) in the right-most equality. These facts, (23) and the definitions of λ, ρ and μ yield

$$\frac{1}{2}\left\| \mu\left(x^*\right) \right\| \le \lambda \le \frac{1}{2}\left\| \rho\left(x^*\right) \right\| = \frac{1}{2}\left\| T_g\left(x^*\right) \right\|,$$

as required. □

5 Numerical experiments

In this section we compare the classical and generalized Newton methods on five example systems of equations of the form $f(x) = \mathbf{0}$ as in (1), with two variables (where visualization is possible) as well as six variables. The equations in these test problems involve cubic, quartic and exponential functions. Using various choices of the generalizing function s, we look at both local and global behaviour of the classical and generalized methods, and we do this in the following sense.

- *Local behaviour*: For each test problem we check that, with either method, convergence to a solution is quadratic, verifying Theorem 3.2. We do this by obtaining numerical estimates of the asymptotic error constant λ. Comparisons of the estimates of λ for each method give us an idea as to which method is faster locally. Recall that for any method that we consider the convergence rate is quadratic. So by comparing the λ, we compare the local "speeds" of quadratically convergent methods. In other words, the ratio of the λ of two methods will tell us how many times a method is "faster" or "slower" than the other, locally. We also provide theoretical bounds on λ, i.e., intervals in which λ lies, for each example by using Theorem 4.1.
- *Global behaviour*: It is well known that the main drawback of the classical Newton method is its dependence on the quality of the initial guess (or the starting point) used in the Newton iterations. For the systems with two equations in two unknowns, we visualize graphically the colour-coded number of iterations that either method is required to converge, if at all, over domains of various sizes, with the set tolerance of 10^{-8}. These graphs serve to demonstrate the value of the generalized method: The domain in which the generalized method converges in a reasonable number of

iterations can be made larger, by choosing the generalizing function s carefully. We also present statistical information about the global convergence properties by means of a large number of randomly generated starting points for each method, in order to verify the information conveyed by the graphs. This information ultimately leads to a decision as to which of the methods considered is best to use, on the average, in a given search domain.

For computations, we use MATLAB Release 2019b, update 5. In getting the estimates of λ, variable precision arithmetic (`vpa` of MATLAB) making use of a large number of digits is utilized to be able to obtain these estimates with relatively reliable number of significant figures.

The CPU times are reported by running MATLAB on a 13-inch 2018 model MacBook Pro, with the operating system macOS Mojave (version 10.14.6), the processor 2.7 GHz Intel Core i7 and the memory 16 GB 2133 MHz LPDDR3.

The MATLAB code we have written to generate the colour-coded portraits of number of iterations is based on Cleve Moler's code for viewing fractals generated by univariate Newton iterations [10] in complex plane.

The methodology used in obtaining the statistical information, such as the average number of iterations, the rate of success of a method, and the CPU times for each iteration and each successful run, on the average, are explained in detail only once, in the first example in Sect. 5.1. For brevity, we avoid repetitions of this information in the subsequent four examples.

5.1 Quartic equations

To compare the classical and generalized methods we will first consider the following example system involving simple quartic functions:

$$f(x) = \begin{bmatrix} x_2 x_1^3 - 1 \\ x_1 x_2^3 - 1 \end{bmatrix} = \mathbf{0}, \tag{24}$$

where $f : \mathbb{R}^2 \to \mathbb{R}^2$ and $\mathbf{0} \in \mathbb{R}^2$. Clearly, System (24) has two real solutions; namely $x^* = (1,1)$ and $x^* = (-1,-1)$. The expression for the fixed-point map g associated with this system can be derived by using (9) with a chosen s. In Table 1, we do this first with $s(x) = (x_1, x_2)$ for the system in (24) and get g for the classical Newton method. The appearance of x_1^3 and x_2^3 in the first and second equations, respectively, prompts us to choose $s(x) = (x_1^3, x_2^3)$ for the generalized Newton method. Then we use this s to get the g for the generalized method as displayed in Table 1. We refer to the method obtained in this way the *cube-generalized Newton method*.

By Theorem 3.2, the fixed-point methods (classical and generalized Newton) using the choices of g listed in Table 1 are quadratically convergent and this can numerically be

Table 1 System (24): Fixed-point map g with different choices of s

$f(x)$	$s(x)$	$g(x)$
$\begin{bmatrix} x_2 x_1^3 - 1 \\ x_1 x_2^3 - 1 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$	$\begin{bmatrix} x_1 - (2x_1^3x_2^3 - 3x_2^2 + x_1^2)/(8x_1^2x_2^3) \\ x_2 - (2x_1^3x_2^3 - 3x_1^2 + x_2^2)/(8x_1^3x_2^2) \end{bmatrix}$
	$\begin{bmatrix} x_1^3 \\ x_2^3 \end{bmatrix}$	$\begin{bmatrix} \sqrt[3]{x_1^3 - 3(2x_1^3x_2^3 - 3x_2^2 + x_1^2)/(8x_2^3)} \\ \sqrt[3]{x_2^3 - 3(2x_1^3x_2^3 - 3x_2^2 + x_1^2)/(8x_1^3)} \end{bmatrix}$

Table 2 System (24): Asymptotic error constants of the classical ($s_i(x) = x_i$, $i = 1, 2$) and generalized ($s_i(x) = x_i^3$, $i = 1, 2$) Newton methods

Soln	$s_i(x)$	$[\|\mu(x^*)\|/2, \|\rho(x^*)\|/2]$	λ	λ_N/λ_{GN}
1 & 2	x_i	$[0, 1.7]$	1.06	
	x_i^3	$[0, 0.8]$	0.35	3.0

verified. What can one say about the asymptotic error constant? Table 2 encapsulates the ensuing answer. The values listed for λ are the numerical estimates of the asymptotic error constant $\lim_{k\to\infty} \|x^{k+1} - x^*\|/\|x^k - x^*\|^2$ from Definition 2.5. These estimates are the same for both of the solutions $x^* = (1, 1)$ and $x^* = (-1, -1)$ (referred to as Solutions 1 & 2) because of the symmetry of the equations in System (24). We note that the estimates of λ consistently fall into the intervals defined by the theoretical bounds established for λ in Theorem 4.1, which are also shown in the table. The (estimated) ratio λ_N/λ_{GN} of the asymptotic error constants of the classical and generalized Newton methods, respectively, implies that the generalized method is about three times faster near a solution, for this example.

We pointed out in the Introduction that, for a given solution, a quadratic convergence region is in general not known *a priori*. Next we graphically illustrate in Fig. 1 that the (quadratic) convergence regions about the solutions for the generalized Newton method we have devised for this example are larger than those resulting from the classical Newton method. We also look at the regions of convergence over larger domains than just the neighbourhoods of the solutions.

In the graphs in Fig. 1, the number of iterations needed to converge with tolerance 10^{-8} to a solution from a given point (i.e., an initial guess) is colour-coded as indicated by the colour bar next to each graph: while 2–4 iteration runs are represented by dark blue, 14 or more iteration runs, which are regarded as "*unsuccessful*," are represented by yellow. The initial guesses are generated over a 1000×1000 grid in the *search domains* $[-3, 3]^2$, $[-10, 10]^2$ and $[-100, 100]^2$. The following immediate observations point to some desirable properties of the cube-generalized method for this example:

- Overall, the graphs associated with the cube-generalized method have far smaller yellow regions.
- We note by looking at the $[-3, 3]^2$-domain that the regions in which convergence is achieved in 2–6 iterations are much larger than that for the classical method.
- In particular, if the search domain is chosen to be much larger, for example $[-100, 100]^2$, then the classical method is unlikely to converge, while the cube-generalized method has a much better chance to converge.

Next we carry out further numerical experiments to support some of our visual observations in Fig. 1. In each of the domains $[-3, 3]^2$, $[-10, 10]^2$ and $[-100, 100]^2$, we randomly generate one million starting points and record the number of iterations needed to converge from each point. We re-iterate that, if the number of iterations is 14 or greater, then we deem that particular run *unsuccessful*. In Table 3, we list, for several typical choices of s, the average number of iterations over each of the search domains for the successful runs. We also list the percentage of the runs that were successful, namely the success rate.

The CPU time taken by a single iteration of the successful runs on the average cannot be found reliably by simply measuring and recording each successful run time and then averaging them, since the very short CPU time of a single run (to the order of 10^{-6}) cannot be measured reliably. Therefore, with 100 random starting points, we repeat each successful

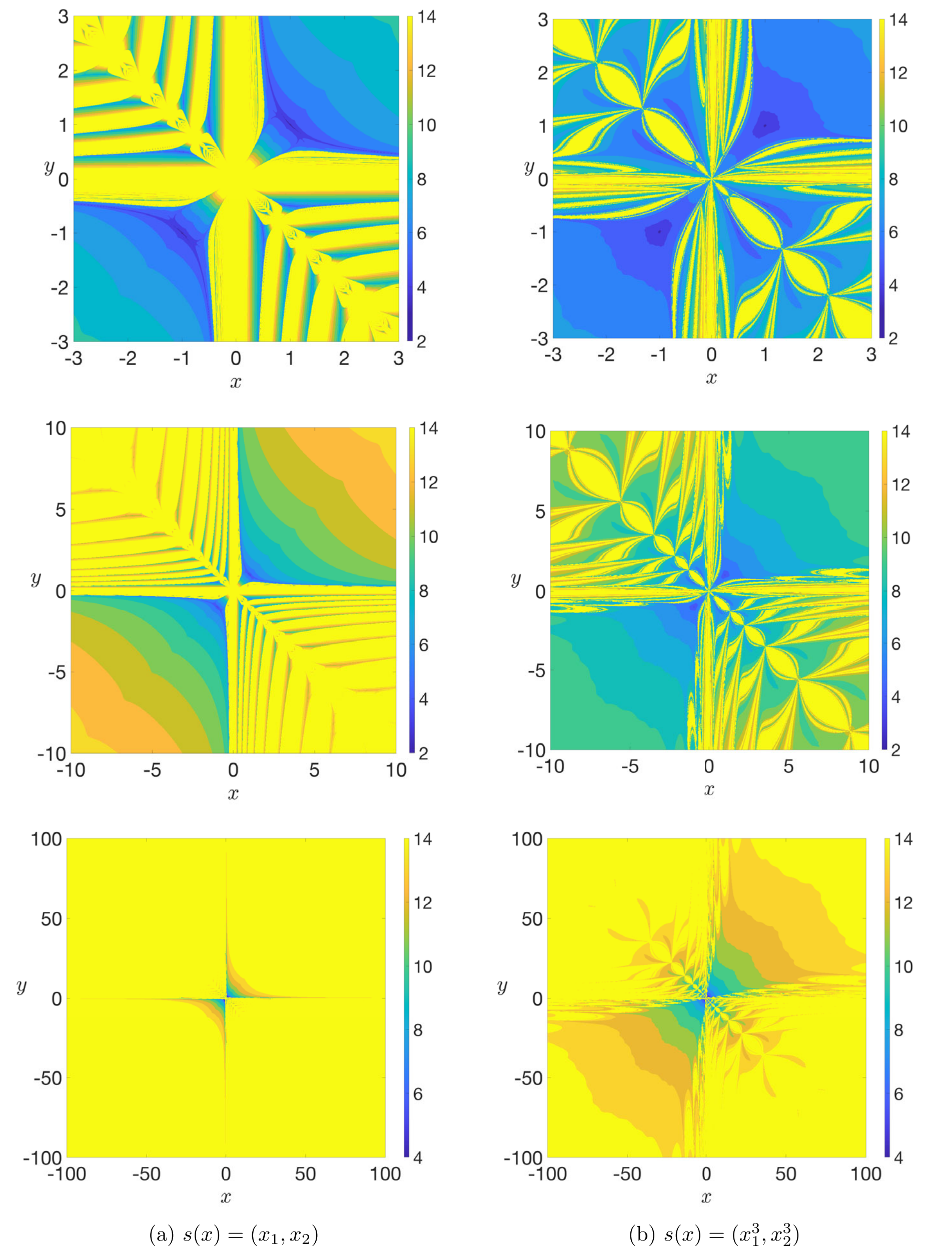

Figure 1 System (24): Portraits of colour-coded number of iterations required for convergence

Table 3 System (24): Performance of the classical and generalized Newton methods with one million randomly generated starting points in domains of various sizes

$s_i(x)$	$[-3,3]^2$		$[-10,10]^2$		$[-100,100]^2$		CPU time/ successful iter [sec]
	Ave iter	Success rate [%]	Ave iter	Success rate [%]	Ave iter	Success rate [%]	
x_i	8.0	56.4	10.5	56.9	11.8	2.0	2.7×10^{-6}
x_i^3	7.1	77.0	8.9	78.6	12.3	36.2	4.7×10^{-6}
$\sinh(x_i)$	7.9	67.7	9.0	25.7	9.0	0.3	2.9×10^{-6}
e^{x_i}	9.0	76.0	10.7	27.6	10.6	0.3	5.1×10^{-6}
$\tan x_i$	5.9	10.9	6.5	14.8	7.1	0.3	3.0×10^{-6}

Table 4 System (24): CPU time needed on average by the classical and generalized Newton methods to obtain a solution in less than 14 iterations, based on the data in Table 3

$s_i(x)$	Time needed to get a single soln [sec]		
	$[-3,3]^2$	$[-10,10]^2$	$[-100,100]^2$
x_i	3.8×10^{-5}	$\boxed{5.0 \times 10^{-5}}$	1.6×10^{-3}
x_i^3	4.3×10^{-5}	5.3×10^{-5}	$\boxed{1.6 \times 10^{-4}}$
$\sinh(x_i)$	$\boxed{3.4 \times 10^{-5}}$	1.0×10^{-4}	8.7×10^{-3}
e^{x_i}	6.0×10^{-5}	2.0×10^{-4}	1.8×10^{-2}
$\tan x_i$	1.6×10^{-4}	1.3×10^{-4}	7.1×10^{-3}

run 10^5 times and take the average. This provides an accurate averaged measure of the CPU time per iteration, which is listed for each method in the last column of Table 3.

Table 3 tells us that over the search domain $[-3,3]^2$ the cube-generalized method is successful 77% of the time it is run, while the success rate of the classical method is 56%. When we generate initial points randomly over a much larger domain, i.e., over $[-100,100]^2$ (this might as well be the situation when we have no knowledge of the location of a solution), the difference in the success rates of the two methods is striking: while the cube-generalized method is successful 36% of the time, the classical method is successful a mere 2% of the time it is run. Although the latter case tells clearly what method to use in the domain $[-100,100]^2$, in the other cases, the success rates alone are not sufficient to tell which method will be (globally) "better" to use.

To be able to have a clear idea about which method is more desirable than the others, we need to find the time a method needs before it obtains a solution. Suppose that, for a given method, the CPU time for a successful run is 3.1×10^{-5} sec and the success rate is 50%. Then, statistically speaking, on average one will need to run that method twice to get a single solution and the time required for this effort will be 6.2×10^{-5} sec. So we can find the time required to obtain a solution by a given method as: the CPU time per successful iteration, times the average number of iterations, divided by the success rate written as a decimal. The CPU times obtained in this way for each method are tabulated in Table 4.

From the global convergence point of view, the method with the smallest CPU time over a domain in Table 4 should be selected, which are framed for each of the three domains of concern. For the domain $[-3,3]^2$, the time required by the classical Newton method is about 12% worse than the generalized method with $s_i(x) = \sinh(x_i)$, $i = 1,2$, which we refer to as the *sinh-generalized Newton method*. For the domain $[-10,10]^2$, the classical method seems to be the best to use, although its closest contender, the cube-generalized method, takes only 6% longer time to find a solution. Over the domain $[-100,100]^2$, the cube-generalized method is clearly the best method to use, as the classical method needs about 10 times more time in obtaining a solution. To rephrase the latter statement: the cube-generalized method is expected to obtain 10 solutions by the time the classical method finds one.

5.2 Equations involving exponentials

The following system is a special instance of the Jennrich and Sampson test problem presented in [6, 11]:

$$f(x) = \begin{bmatrix} e^{x_1} + e^{x_2} - 3 \\ e^{2x_1} + e^{2x_2} - 6 \end{bmatrix} = \mathbf{0}. \tag{25}$$

Table 5 System (25): Asymptotic error constants of the classical and generalized Newton methods

Soln	$s_i(x)$	$[\|\mu(x^*)\|/2, \|\rho(x^*)\|/2]$	λ	λ_N/λ_{GN}
1 & 2	x_i	[0.05, 2.81]	0.9	
	e^{x_i}	[0.19, 2.64]	0.35	2.6

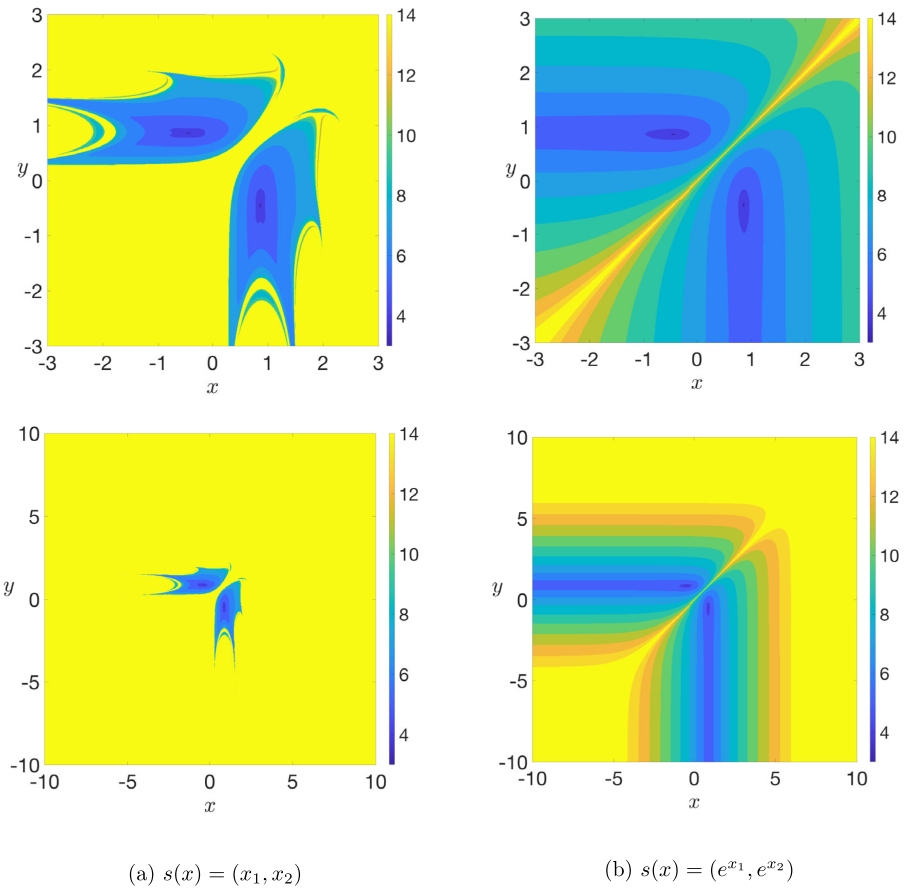

Figure 2 System (25): Portraits of colour-coded number of iterations required for convergence

System (25) has two solutions, namely $x^* = (a, b)$ and $x^* = (b, a)$, referred to here as *Solutions* 1 and 2, respectively, where $a = \ln((3 + \sqrt{3})/2) \approx 0.861211502516490$ and $b = \ln((3 - \sqrt{3})/2) \approx -0.455746394408326$, with the approximations correct to 15 dp. The appearance of the exponential functions in the equations prompts us to choose $s(x) = (e^{x_1}, e^{x_2})$ for the generalized Newton method's fixed-point map in (9). We refer to this method as *exp-generalized Newton method*. As before, $s(x) = (x_1, x_2)$ is used for the classical Newton method.

Table 5 lists the numerical estimates and the theoretical intervals for the asymptotic error constant λ, giving some idea about the local behaviour around a solution. However, we note that the ratio λ_N/λ_{GN} is not so accurate in this case as the values obtained in the latter iterations for λ_N seem to fluctuate between 0.4 and 1.4, which we have averaged as 0.9. The approximate value listed for λ_N/λ_{GN} implies that, close enough to a solution, the exp-generalized method is more than twice faster.

As in the example in Sect. 5.1, we depict, in Fig. 2 the colour-coded number of iterations needed to converge to any one of the two solutions. The success of the exp-generalized

Table 6 System (25): Performance of the classical and generalized Newton methods with one million randomly generated starting points in domains of various sizes

$s_i(x)$	$[-3,3]^2$		$[-10,10]^2$		CPU time/ successful iter [sec]
	Ave iter	Success rate [%]	Ave iter	Success rate [%]	
x_i	6.6	25.0	6.7	2.4	2.7×10^{-6}
x_i^3	7.3	12.3	7.3	1.1	4.6×10^{-6}
$\sinh(x_i)$	6.2	17.4	6.2	1.6	2.9×10^{-6}
e^{x_i}	7.8	98.3	9.6	53.3	6.7×10^{-6}
$\tan x_i$	6.1	9.4	6.4	10.0	2.9×10^{-6}

Table 7 System (25): CPU time needed on the average by the classical and generalized Newton methods to obtain a solution in less than 14 iterations, based on the data in Table 6

$s_i(x)$	Time needed to get a single soln [sec]	
	$[-3,3]^2$	$[-10,10]^2$
x_i	7.1×10^{-5}	7.5×10^{-4}
x_i^3	2.7×10^{-4}	3.1×10^{-3}
$\sinh(x_i)$	1.0×10^{-4}	1.1×10^{-3}
e^{x_i}	$\boxed{5.2 \times 10^{-5}}$	$\boxed{1.1 \times 10^{-4}}$
$\tan x_i$	1.9×10^{-4}	1.9×10^{-4}

method is even more striking in this case: (i) the graphs for the exp-generalized method have far smaller yellow regions, (ii) local convergence regions (that are achieved in 4–6 iterations, shown in darker shades of blue) for the exp-generalized method are much larger and (iii) over the larger domain $[-10,10]^2$, the exp-generalized method has a far better chance of converging in less than 14 iterations.

Table 6 provides some statistical data as in the case of Table 3 for System (24) in the previous subsection. It should be noted that the percentage success rates in the table are in agreement with the percentage of the regions which are not yellow in Fig. 2, for the cases of $s(x) = (x_1, x_2)$ and $s(x) = (e^{x_1}, e^{x_2})$. Table 6 also includes other choices of s for a wider comparison.

When successful the CPU time one iteration of the classical Newton method spends on the average (over the domain $[-3,3]^2$) is 2.7×10^{-6} sec. The same CPU time for the exp-generalized Newton method is 6.7×10^{-6} sec, which is about 2.5 times longer. On the other hand, over the domain $[-3,3]^2$, the chance of finding a solution for the exp-generalized method in less than 14 iterations is nearly 4 times higher than using the classical method. Moreover, over the domain $[-10,10]^2$, the exp-generalized method is 23 times more likely to find a solution in the same manner. These likelihoods of success which are greatly in favour of the exp-generalized method seems to offset the higher computational times per iteration.

To make sure of the conclusion we have just drawn above as to which method is preferred, we can again prepare a table listing the average CPU time needed for a successful run by each method, as it was previously done in Table 4 for System (24). Table 7 lists these times, which immediately reconfirms that the exp-generalized method should indeed be the preferred method over $[-3,3]^2$, as it would take the classical method 37% more time to find a solution. Over the larger domain $[-10,10]^2$, by the time the classical method finds a solution the exp-generalized method will have already found about seven solutions—see the framed CPU times.

This is yet another example which clearly illustrates how the structure of the problem can be exploited to solve a system of equations by means of a generalized Newton method.

5.3 Cubic equations in two variables

The example we deal with in this section emanates from an unconstrained global optimization problem solved in [1], which asks one to minimize the function $\varphi : \mathbb{R}^2 \to \mathbb{R}$ given as

$$\varphi(x) = \left(x_1^2 - 1\right)^2 + \left(x_2^2 - 2\right)^2 - 0.7x_1x_2 + 0.2x_1 + 0.3x_2. \tag{26}$$

Although the numerical optimization method proposed in [1] can find the global minimizer of φ, common numerical optimization approaches often only find a stationary point of the function φ, by finding a zero of the gradient of φ, namely, effectively, they find a solution to the system of equations

$$f(x) := \nabla\varphi(x) = \begin{bmatrix} 4x_1^3 - 4x_1 - 0.7x_2 + 0.2 \\ 4x_2^3 - 8x_2 - 0.7x_1 + 0.3 \end{bmatrix} = \mathbf{0}. \tag{27}$$

In [1], five extremal solutions of the function in (26) are listed as in Table 8. Solutions 1–4 are all local minima while Solution 5 is a local maximum.

The appearance of x_1^3 and x_2^3 in (27) prompts us to consider $s(x) = (x_1^3, x_2^3)$ as the first generalized method in Table 9. Via experiments we observe that the choice $s(x) = (\sinh x_1, \sinh x_2)$ yields another worthwhile generalization of the Newton method. Table 9

Table 8 Extremal solutions of $\varphi(x)$ in (26)

Soln	x_1	x_2
1	−1.128494496205920	−1.477960288994776
2	1.088972069871674	1.442265902284124
3	0.79262879889394	−1.398008585571904
4	−0.888779137505495	1.352613115553849
5	0.044197271093630	0.033651793151170

Table 9 System (27): Asymptotic error constants of the classical and generalized Newton methods

Soln	$s_i(x)$	$[\|\mu(x^*)\|/2, \|\rho(x^*)\|/2]$	λ	λ_N/λ_{GN}
1	x_i	[0.08, 1.5]	1.2	
	x_i^3	[0.08, 0.44]	0.3	4.0
	$\sinh(x_i)$	[0.08, 0.96]	0.8	1.5
2	x_i	[0.09, 1.6]	1.0	
	x_i^3	[0.09, 0.5]	0.3	3.3
	$\sinh(x_i)$	[0.09, 1.1]	0.6	1.7
3	x_i	[0, 2.9]	2.7	
	x_i^3	[0, 1.5]	1.4	1.9
	$\sinh(x_i)$	[0, 2.5]	2.4	1.1
4	x_i	[0, 2.3]	1.2	
	x_i^3	[0, 0.94]	0.4	3.0
	$\sinh(x_i)$	[0, 1.8]	0.7	1.7
5	x_i	[0, 0.14]	0.05	
	x_i^3	[0, 37]	29.8	0.002
	$\sinh(x_i)$	[0, 0.17]	0.16	0.3

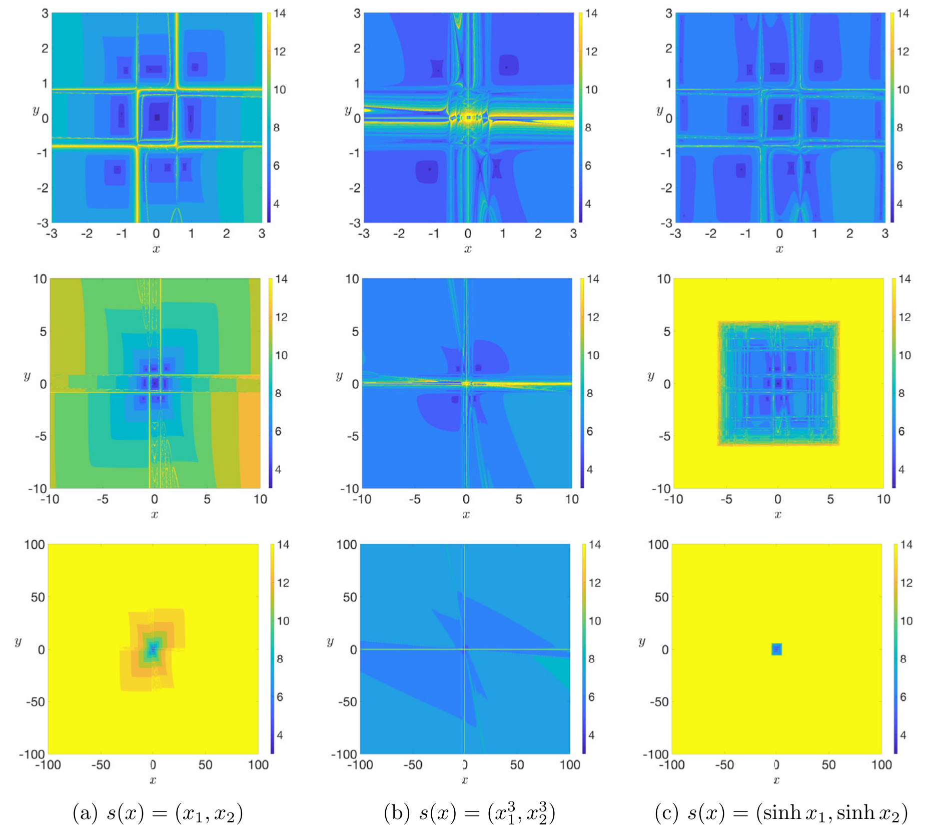

Figure 3 System (27): Portraits of colour-coded number of iterations required for convergence

Table 10 System (27): Performance of the classical and generalized Newton methods with one million randomly generated starting points in domains of various sizes

$s_i(x)$	$[-3,3]^2$		$[-10,10]^2$		$[-100,100]^2$		CPU time/ successful iter [sec]
	Ave iter	Success rate [%]	Ave iter	Success rate [%]	Ave iter	Success rate [%]	
x_i	7.0	98.6	9.7	99.3	12.2	9.8	3.7×10^{-6}
x_i^3	6.1	98.6	6.3	99.7	6.8	100.0	6.1×10^{-6}
$\sinh(x_i)$	5.9	99.8	7.9	34.8	7.8	0.3	4.1×10^{-6}
e^{x_i}	7.1	98.7	10.4	42.4	10.4	0.4	4.7×10^{-6}
$\tan x_i$	6.7	70.7	7.3	57.5	7.8	3.3	4.0×10^{-6}

reveals that the estimates of λ for both of the generalized methods are (by two to four times) smaller at Solutions 1–4, and larger only at Solution 5.

To illustrate the overall behaviour, Fig. 3 provides a visualization of the success of three methods in terms of the number of iterations. In addition to the classical method, we consider the cube- and sinh-generalized methods.

Glancing at the 3×3 matrix of graphs of Fig. 3, while the graphs in the entries $(2,2)$ and $(3,2)$ have more of the shades of blue than those in the same rows, the graph in $(1,3)$ appears to have more of the shades of blue and almost no yellow. The success rates by judging from the non-yellow regions in these graphs are corroborated by the success rates presented in Table 10. What seem to be the best-performing methods by looking at these

Table 11 System (27): CPU time needed on the average by the classical and generalized Newton methods to obtain a solution in less than 14 iterations, based on the data in Table 10

$s_i(x)$	Time needed to get a single soln [sec]		
	$[-3,3]^2$	$[-10,10]^2$	$[-100,100]^2$
x_i	2.6×10^{-5}	$\boxed{3.6\times10^{-5}}$	4.6×10^{-4}
x_i^3	3.8×10^{-5}	3.9×10^{-5}	$\boxed{4.1\times10^{-5}}$
$\sinh(x_i)$	$\boxed{2.4\times10^{-5}}$	9.3×10^{-5}	1.1×10^{-2}
e^{x_i}	3.4×10^{-5}	1.2×10^{-4}	1.2×10^{-2}
$\tan x_i$	3.8×10^{-5}	5.1×10^{-5}	9.5×10^{-4}

graphs are also in agreement with the ones corresponding to the framed entries in Table 11.

By looking at Table 11, we deduce easily that the sinh-generalized method is the best, although the classical method is only slightly worse, in terms of the time they take for a successful run in the domain $[-3,3]^2$. The classical Newton is the best for $[-10,10]^2$, with this time the cubic-generalized method being slightly worse, taking 8% longer time in finding a solution. In the largest domain $[-100,100]^2$, the cube-generalized method is by far the best, as its nearest contender, the classical Newton method, takes more than 11 times longer to obtain a single solution.

Going back to Fig. 3, we deduce from the first row of graphs that the regions of convergence in 4–6 iterations of the classical method are considerably enlarged by both of the generalized methods. This is in agreement with the estimated values of $\lambda_N/\lambda_{\mathrm{GN}}$ in Table 9.

5.4 Cubic equations in six variables

We consider another system of cubic equations, but this time the number of equations and unknowns is six. The system originates from the problem of (globally) minimizing the function $\varphi:\mathbb{R}^6\to\mathbb{R}$, which was studied in [1, 15], given by

$$\varphi(x)=\sum_{i=1}^{6}a_ix_i^4+x^TBx+d^Tx, \tag{28}$$

where

$$a=\begin{bmatrix}9\\2\\6\\4\\8\\7\end{bmatrix},\qquad B=\begin{bmatrix}4&4&9&3&4&1\\4&3&7&9&9&2\\9&7&4&7&6&6\\3&9&7&4&2&6\\4&9&6&2&8&3\\1&2&6&6&3&5\end{bmatrix},\qquad d=\begin{bmatrix}2\\6\\5\\0\\0\\2\end{bmatrix}.$$

We consider the problem of finding the zeroes of the gradient $\nabla\varphi(x)$ of $\varphi(x)$, in other words, the zeroes of

$$f(x):=\nabla\varphi(x)=4\begin{bmatrix}a_1x_1^3\\a_2x_2^3\\\vdots\\a_6x_6^3\end{bmatrix}+2Bx+d=\mathbf{0}. \tag{29}$$

Table 12 Some of the stationary points of φ in (28)

	Soln 1	Soln 2	Soln 3
x_1	0.545218813388361	−0.599208065573669	0.590580847289543
x_2	−1.464410189791729	−1.571013884485518	1.338889774602320
x_3	−0.720606654276266	0.678323332400517	−0.853265510869097
x_4	1.178144265591973	1.076080413893220	−0.955745102979906
x_5	0.794065108243717	0.745744375791400	−0.646924271685709
x_6	−0.465794119447879	−0.762615830412707	0.708688334528434

Table 13 System (29): Asymptotic error constants of the classical and generalized Newton methods

Soln	$s_i(x)$	$[\|\mu(x^*)\|/2, \|\rho(x^*)\|/2]$	λ	λ_N/λ_{GN}
1	x_i	$[0, 3.4]$	0.7	
	x_i^3	$[0, 1.2]$	0.2	3.5
	$\sinh x_i$	$[0, 2.7]$	1.3	0.5
2	x_i	$[0, 3.1]$	0.8	
	x_i^3	$[0, 1.0]$	0.4	2
	$\sinh x_i$	$[0, 2.4]$	0.5	1.6
3	x_i	$[0, 3.1]$	0.9	
	x_i^3	$[0, 0.9]$	0.4	2.3
	$\sinh x_i$	$[0, 2.3]$	0.8	1.1

Table 14 System (29): Performance of the classical and generalized Newton methods with one million randomly generated starting points in domains of various sizes

$s_i(x)$	$[-3, 3]^2$		$[-10, 10]^2$		$[-100, 100]^2$		CPU time/ successful iter [sec]
	Ave iter	Success rate [%]	Ave iter	Success rate [%]	Ave iter	Success rate [%]	
x_i	10.5	58.8	11.9	41.2	–	0.0	6.3×10^{-6}
x_i^3	8.0	76.7	8.5	48.9	8.8	17.7	9.5×10^{-6}
$\sinh(x_i)$	8.9	74.9	11.1	17.4	–	0.0	6.9×10^{-6}
e^{x_i}	10.8	62.4	12.3	2.2	–	0.0	1.0×10^{-5}
$\tan x_i$	9.2	3.2	9.8	0.6	–	0.0	7.3×10^{-6}

Solutions of (29) are stationary points of φ, in other words, they are candidates for (locally) optimal solutions of φ, three of which are listed in Table 12. The first solution listed in Table 12 is a global minimizer of φ, as reported in [1]. Our aim here is to look at the behaviour of the classical and generalized methods in finding a zero of f, which is only a stationary point of φ.

First we look at the (local) behaviour near the solutions listed in Table 12. Table 13 tabulates the theoretical intervals where λ lies, found using Theorem 4.1, as well as the λ estimated numerically, for each method. We observe that the numerical estimates fall into the theoretical intervals. We also observe that the cube-generalized method has λ consistently 2 to 3.5 times smaller than that of the classical method, and therefore locally faster by the same factors. The sinh-generalized method, on the other hand, is observed to be not so fast. The reason we have included the sinh-generalized method here is that as we will see in Tables 14 and 15 it can have a desirable performance on a larger scale, in search domains of moderate size.

Since System (29) has six variables, we cannot have the kind of visualization of performance as we had in the previous (two-variable) examples. However, we can still carry out runs with randomized (one million) initial points and make some statistical observations

Table 15 System (29): CPU time needed on the average by the classical and generalized Newton methods to obtain a solution in less than 14 iterations, based on the data in Table 14

$s_i(x)$	Time needed to get a single soln [sec]		
	$[-3,3]^2$	$[-10,10]^2$	$[-100,100]^2$
x_i	1.1×10^{-4}	1.8×10^{-4}	–
x_i^3	9.9×10^{-5}	$\boxed{1.7 \times 10^{-4}}$	$\boxed{4.7 \times 10^{-4}}$
$\sinh(x_i)$	$\boxed{8.2 \times 10^{-5}}$	4.4×10^{-4}	–
e^{x_i}	1.7×10^{-4}	5.6×10^{-3}	–
$\tan x_i$	2.1×10^{-3}	1.2×10^{-2}	–

as we did for the previous example systems. Table 14, and subsequently Table 15, provide advice as to which method can be chosen for efficiency.

The framed average CPU times required to get a single solution in Table 15 indicate that the sinh-generalized method should be chosen in the domain $[-3,3]^2$, while the cube-generalized method should be preferred in the larger domains. In $[-3,3]^2$, compared to the sinh-generalized method, the cube-generalized method takes about 21% more time to find a solution, while the classical method requires 34% more time. In the search domain $[-100,100]^2$, the cube-generalized method is unrivalled as none of the other methods is viable to use. We note that in the largest search domain since the other methods has a success rate less than 0.04%, their success rates have been entered as 0.0% into Table 14, with no average number of iterations reported.

5.5 A signal processing problem

In optimum broad-band antenna processing the minimization of the mean output power subject to linear constraints is a common problem [17, 18]. In [18], a 70-tuple example from [17] about this signal processing problem has been transformed into the global minimization of the quartic polynomial function $\varphi : \mathbb{R}^2 \to \mathbb{R}$ given in equation (30) below. The details of this transformation can be found in [18, Appendix C].

$$\begin{aligned}\varphi(x) = {} & a_1 - a_2x_1^2 + a_3x_1^4 - a_4x_1x_2 + a_5x_1^3x_2 - a_6x_2^2 + a_7x_1^2x_2^2 \\ & + a_8x_1x_2^3 + a_9x_2^4,\end{aligned} \tag{30}$$

where

$$\begin{aligned} a_1 &= 0.337280011659804177, & a_2 &= 0.122071359035091510, \\ a_3 &= 0.077257128600040819, & a_4 &= 0.217646697603541049, \\ a_5 &= 0.233083387816363887, & a_6 &= 0.129244611969892874, \\ a_7 &= 0.286227131697582205, & a_8 &= 0.175571952500361967\,3, \\ a_9 &= 0.056769191379277343\,3. \end{aligned}$$

Here we are interested in the problem of finding a stationary point of φ, namely a solution of the system

$$f(x) := \nabla\varphi(x) = \begin{bmatrix} -2a_2x_1 + 4a_3x_1^3 - a_4x_2 + 3a_5x_1^2x_2 + 2a_7x_1x_2^2 + a_8x_2^3 \\ a_4x_1 + a_5x_1^3 - 2a_6x_2 + 2a_7x_1^2x_2 + 3a_8x_1x_2^2 + 4a_9x_2^3 \end{bmatrix} = \mathbf{0}. \tag{31}$$

Table 16 Extremal points for the function (30)

Soln	x_1	x_2	$\varphi(x)$
1	−1.037925846421872	1.188144940421522	0.314501964946967
2	1.037925846421872	−1.188144940421522	0.314501964946967
3	−0.150370553810688	−0.948134491036906	0.262292001977528
4	0.150370553810688	0.948134491036906	0.262292001977528
5	0	0	0.337280011659804

Table 17 System (31): Asymptotic error constants of the classical and generalized Newton methods

Soln	$s_i(x)$	$[\Vert\mu(x^*)\Vert/2, \Vert\rho(x^*)\Vert/2]$	λ	$\lambda_N/\lambda_{\mathrm{GN}}$
1	x_i	$[0.29, 2.6]$	1.6	
	x_i^3	$[0.14, 2.2]$	1.7	0.9
	$\sinh x_i$	$[0.24, 2.4]$	0.6	2.7
	e^{x_i}	$[0.23, 2.3]$	0.4	4
2	x_i	$[0.29, 2.6]$	1.0	
	x_i^3	$[0.14, 2.2]$	1.5	0.7
	$\sinh x_i$	$[0.24, 2.4]$	0.6	1.7
	e^{x_i}	$[0.34, 3.0]$	1.5	0.7
3	x_i	$[0, 3.0]$	1.3	
	x_i^3	$[0, 4.9]$	4.5	0.3
	$\sinh x_i$	$[0, 2.8]$	1.1	1.2
	e^{x_i}	$[0, 3.7]$	2.0	0.7
4	x_i	$[0, 3.0]$	1.3	
	x_i^3	$[0, 4.9]$	0.4	3.3
	$\sinh x_i$	$[0, 2.8]$	1.1	1.2
	e^{x_i}	$[0, 2.5]$	1.0	1.3
5	x_i	–	0.0	
	x_i^3	–	–	–
	$\sinh x_i$	–	0.0	–
	e^{x_i}	$[0, 0.71]$	0.5	0.0

The extremal points of φ, which are zeroes f, and the corresponding functional values are given in Table 16 (also see [18]). Solutions 3 and 4 are the global minimizers of φ.

Table 17 reconfirms that, for Solutions 1–4, convergence is quadratic and the numerically estimated values of λ lie in the theoretical intervals found by using Theorem 4.1. We observe that the sinh-generalized method has λ consistently 1.2 to 2.7 times smaller than that of the classical method, and therefore locally faster by the same factors.

The missing entries for Solution 5 in Table 17 warrants an explanation. Numerical experiments imply that the classical and sinh-generalized Newton methods' rate of convergence at $x = (0, 0)$ is higher than quadratic, since the asymptotic error constants λ of each method for quadratic convergence is estimated to be zero. In fact, interestingly, the rate of convergence for either method (numerically) turns out to be cubic, with the associated asymptotic error constants estimated as 1.0 and 1.6, respectively. The exp-generalized method is the only method in the list which is verified to be quadratically convergent at Solution 5. On the other hand, the cube-generalized method has a singularity at $x = (0, 0)$, and it fails to converge to the solution, no matter how close to the solution the initial guess is chosen.

A visualization of the global performances of the methods listed in Table 17 is provided in Fig. 4. By looking at the graphs, the sinh-generalized method appears to be the best to use in the search domain $[-3, 3]^2$ since it has smaller yellow regions and the domain is dominated more by shades of blue, a sign of quicker convergence. In the slightly bigger search domain of $[-10, 10]^2$, however, while the cube-generalized method seems to have

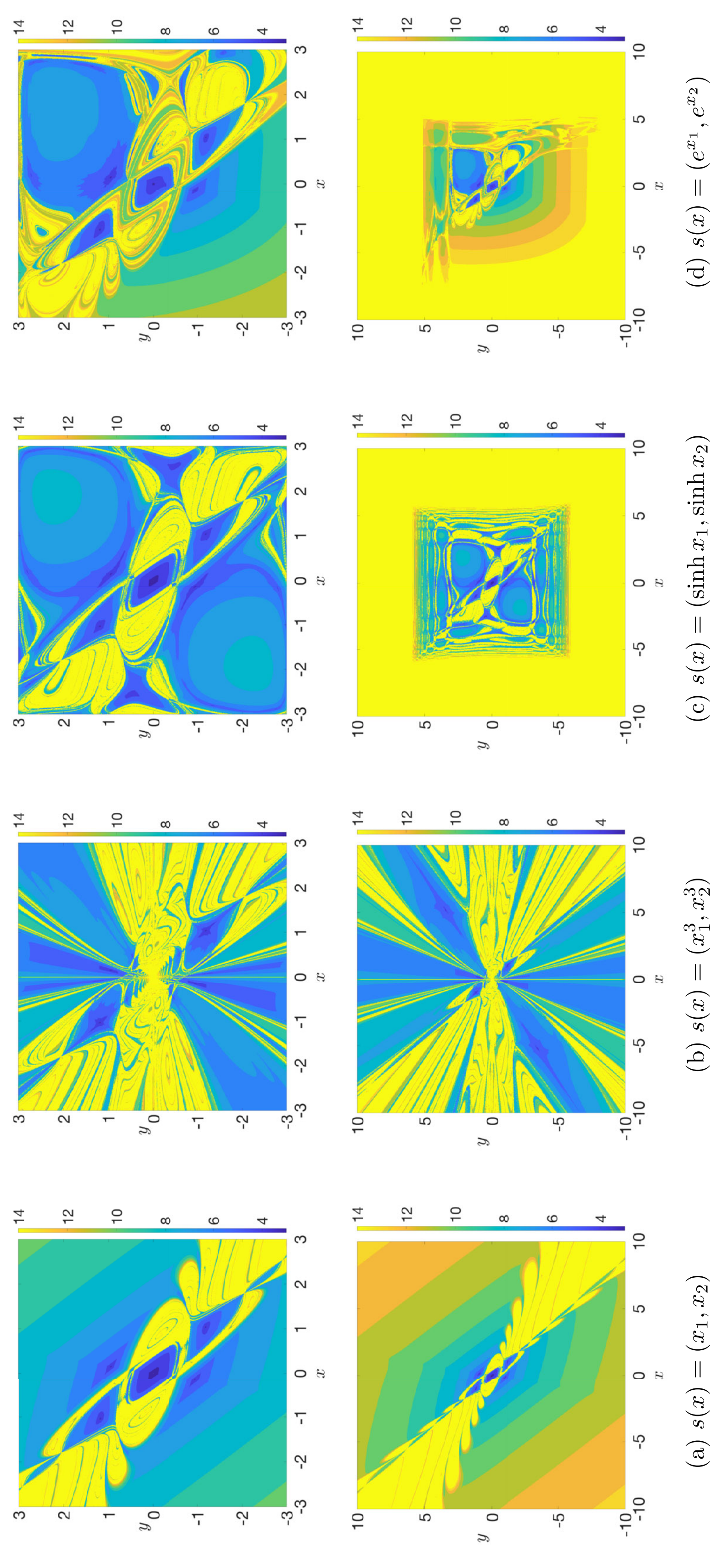

Figure 4 System (31): Portraits of colour-coded number of iterations required for convergence

Table 18 System (31): Performance of the classical and generalized Newton methods with one million randomly generated starting points in domains of various sizes

$s_i(x)$	$[-3,3]^2$		$[-10,10]^2$		$[-100,100]^2$		CPU time/ successful iter [sec]
	Ave iter	Success rate [%]	Ave iter	Success rate [%]	Ave iter	Success rate [%]	
x_i	7.8	80.1	10.5	81.1	12.2	4.2	3.7×10^{-6}
x_i^3	7.8	68.6	8.1	69.7	8.7	67.3	6.2×10^{-6}
$\sinh(x_i)$	6.9	78.5	8.4	25.0	8.3	0.2	3.9×10^{-6}
e^{x_i}	8.6	81.4	10.9	27.6	10.9	0.3	5.1×10^{-6}
$\tan x_i$	6.7	34.9	7.3	24.4	7.9	0.4	3.9×10^{-6}

Table 19 System (31): CPU time needed on the average by the classical and generalized Newton methods to obtain a solution in less than 14 iterations, based on the data in Table 18

$s_i(x)$	Time needed to get a single soln [sec]		
	$[-3,3]^2$	$[-10,10]^2$	$[-100,100]^2$
x_i	3.6×10^{-5}	$\boxed{4.8 \times 10^{-5}}$	1.1×10^{-3}
x_i^3	7.0×10^{-5}	7.2×10^{-5}	$\boxed{8.0 \times 10^{-5}}$
$\sinh(x_i)$	$\boxed{3.4 \times 10^{-5}}$	1.3×10^{-4}	1.6×10^{-2}
e^{x_i}	5.4×10^{-5}	2.0×10^{-4}	1.9×10^{-2}
$\tan x_i$	7.5×10^{-5}	1.2×10^{-4}	7.7×10^{-3}

the largest regions of shades of blue, the classical method looks to have the smallest regions of yellow. In view of the difficulty of judging from Fig. 4 as to which method is preferable, we will resort to the statistical data presented in Tables 18 and 19 for a more conclusive decision.

As in the previous examples, we have run the classical and various generalized methods, with randomized initial points, to obtain the statistics in Tables 18. The only clear case for the choice of a method is when the search domain is $[-100,100]^2$, for which the cube-generalized method should certainly be the method of preference, given the relatively very high success rate, smaller number of iterations and not so much longer CPU time per iteration, all on the average. To determine, ultimately, which of the methods will be preferable in the other search regions, we need to refer to Table 19, as in the previous examples.

The boxed CPU times in Table 19 dictate that while the sinh-generalized method should be the method of choice in $[-3,3]^2$, the classical Newton method should better be used in $[-10,10]^2$. We observe that the cube-generalized method is more than 700 times more efficient than its nearest contender, the classical method, in the large search domain $[-100,100]^2$. This equivalently means that by the time the classical method finds a single solution, the cube-generalized method will have obtained more than 13 solutions, on the average.

6 Conclusion and discussion

We have proposed a family of generalized Newton methods facilitated by an auxiliary, or generalizing, function s, for solving systems of nonlinear equations. The method reduces to the classical Newton method if the generalizing function is the identity map, i.e., $s(x) = x$. Under mild assumptions, we have proved that the new family of methods are quadratically convergent just like the classical one. We derived expressions for the bounds on the asymptotic error constants of the family. These bounds, which can be computed for practical problems easily as illustrated in the numerical experiments, can provide an idea about the relative local speeds of the classical and generalized methods, although they are not tight.

For numerical experimentation, we have considered three types of problems, namely systems of equations involving quartic (in two variables), cubic (in two and six variables), and exponential (in two variables), functions. We carried out extensive numerical experiments using $s(x) = x$ (the classical method), and $s(x) = x^3$, $\sinh x$, e^x, and $\tan x$ (the cube-, sinh-, exp- and tan-generalized methods, respectively), for each of the example problems.

For the problems in two variables, we constructed graphs depicting, for search domains of various sizes, a portrait of the colour-coded number of iterations a method would need to converge to a solution. These graphs, or portraits, were observed to provide a broad idea as to which method is likely to be more preferable. Using one million randomly generated initial points in each chosen search domain, we presented tables reporting the success rate and average number of iterations of a method as well as the average CPU time one iteration of that method takes. By using this data, we were able to identify the method with the smallest CPU time required for finding a single solution, as the preferred method in a particular search domain.

For the cubic and quartic problems we have considered, the sinh-generalized method seems to be particularly successful in relatively smaller search domains. In slightly larger domains where the sinh-generalized method is not so successful anymore, the classical Newton method looks like the method of preference for the two-variable problems. For very large domains, the cube-generalized method certainly looks to be the method of choice, if not the only successful method for some problems. We found that, for the exponential problem we studied, the exp-generalized method seems to be the only method one should use in domains of any size.

The kind of numerical exploration we performed could be particularly useful in the case when a system of equations involves parameters and these parameters change only slightly so that the portraits and the statistical data are not altered much. In other words, given a system of equations

$$f(x,p) = \mathbf{0}, \tag{32}$$

with a vector of n unknowns, $x \in \mathbb{R}^n$, and a fixed vector of m parameters, $p \in \mathbb{R}^m$, the task would be to find a solution of (32) as efficiently as possible. Suppose that we have identified the preferred generalized method through the numerical exploration we have devised in this paper. When p has changed slightly, the preferred method might then be employed to find a new solution of (32).

We have demonstrated that a suitable choice of s is possible for some specific forms of systems of equations. Making informed choices of s for more general problems still stands as a challenging research problem. Figuratively speaking, Eq. (2) implies that the best choice of s would be f, i.e. $s = f$, in which case Eq. (1) is readily solved. On the other hand, with the choice of $s = f$, s^{-1} cannot in general be written in terms of elementary functions. Therefore the generalized Newton iterate we are proposing, which involves s^{-1} on the right-hand side, cannot be written explicitly and thus cannot be employed as a numerical procedure. Bearing this metaphor in mind, we propose to choose an s for which s^{-1} can be written explicitly and is as close (in some sense) to f as possible. One way of achieving the latter is to choose s as one of the elementary functions appearing in f which seems to be "dominating" over the other elementary functions appearing in f. This is one of the insights that we use in choosing s in the numerical experiments, and numerically show that certain choices of s for certain given equations do provide an edge over the classical Newton method.

To re-iterate, in the current paper, we have numerically investigated (over domains of various sizes) as to what function s needs to be chosen, by considering the candidates $s(x) = x$ (the classical Newton), x^3, $\sinh(x)$, e^x and $\tan x$, for f involving cubic, quartic and exponential functions. Perhaps this list of functions for s can be expanded, for example by including the choices of the s functions, $\log x$, $1/x$, x^5, $1/x^3$ as in [2], as well as other possible choices of elementary functions, depending on a given f. This is something we leave as an interesting future direction of work.

Like most available modifications on the Newton method, our generalized version may switch to the classical one (i.e., with $s(x) = x$) or to another generalized method when the current choice of s is either inconvenient or provides no detectable advantage. One may refer to this version as a *hybrid* generalized Newton method. For example, when solving the cubic problems, based on the results in the tables with CPU times to get a single solution, it might be interesting to devise a hybrid method which switches from the classical or the cube-generalized method to the sinh-generalized method as iterations fall into the search domain $[-3,3]^2$.

In the future, it would be valuable to study convergence regions as it was done in [7, 16] by Kantorovich and Smale, for the new generalized methods. Initial leads for such a study can for example be found in the classical text book by Dennis and Schnabel [4, Sect. 5.3]. The latter result, stated for the system $F(y) = 0$, where $F = f \circ s^{-1}$ asserts that: if $J_F(y_0)$ invertible, J_F is Lipschitz continuous in a region containing y_0, and the first step of the Newton method is "sufficiently small relative to the nonlinearity of F," then there must be a root in this region, and furthermore this root must be unique. Subsequently, in [4, Theorem 5.3.1], convergence of the Newton method from y_0 is ensured. Although the rate of this convergence is only guaranteed to be r-quadratic, it would still be interesting to explore particular choices of s (for a given f), which can make this *Kantorovich-type* region larger. Such an analysis might indeed provide a more practical rule, or guide, for the choice of s.

It would also be interesting to consider extending the work presented in this paper to situations where global convergence to a solution is guaranteed, such as the Levenberg–Marquardt approach [8, 9].

Acknowledgements
The authors would like to offer their warm thanks to the two anonymous referees whose comments and suggestions improved the paper.

Availability of data and materials
All data on which the conclusions of this manuscript relies is deposited in an Arxiv preprint with the same title and the same authors.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
RSB, BIC and CYK contributed to the theoretical setting, the convergence analysis, and the bounds on the asymptotic error constant. The numerical experiments were also carried out by all the named authors in coordination with one another. All authors read and approved the final manuscript.

Publisher's Note

Received: 29 March 2021 Accepted: 25 August 2021 Published online: 20 September 2021

References

1. Burachik, R.S., Kaya, C.Y.: Steklov convexification and a trajectory method for global optimization of multivariate quartic polynomials. Math. Program. (2020). https://doi.org/10.1007/s10107-020-01536-8
2. Burachik, R.S., Kaya, C.Y., Sabach, S.: A generalized univariate Newton method motivated by proximal regularization. J. Optim. Theory Appl. **155**, 923–940 (2012)
3. Corless, R.M., Fillion, N.: A Graduate Introduction to Numerical Methods: From the Viewpoint of Backward Error Analysis. Springer, New York (2013)
4. Dennis, J.E. Jr, Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. SIAM Classics in Applied Mathematics. SIAM, Philadelphia (1996)
5. Deuflhard, P.: Newton Methods for Nonlinear Problems: Affine Invariant and Adaptive Algorithms. Springer, Berlin (2011)
6. Jennrich, R.I., Sampson, P.F.: Application of stepwise regression to nonlinear estimation. Technometrics **10**, 63–72 (1968)
7. Kantorovich, L.V.: On Newton's method for functional equations. Dokl. Akad. Nauk **59**, 1237–1240 (1948)
8. Levenberg, K.: A method for the solution of certain nonlinear problems in least squares. Q. Appl. Math. **2**, 164–168 (1944)
9. Marquardt, D.: An algorithm for least squares estimation of nonlinear parameters. SIAM J. Appl. Math. **11**, 431–441 (1963)
10. Moler, C.: Fractal global behavior of Newton's method (2016). https://blogs.mathworks.com/cleve/2016/01/18/fractal-global-behavior-of-newtons-method. Accessed 24 March 2021
11. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**, 17–41 (1981)
12. Ortega, J.M., Rheinboldt, W.C.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York (1970)
13. Ostrowski, A.M.: Solution of Equations and Systems of Equations. Academic Press, Basel (1960)
14. Polyak, B.T.: Newton's method and its use in optimization. Eur. J. Oper. Res. **181**, 1086–1096 (2007)
15. Qi, L., Wan, Z., Yang, Y.-F.: Global minimization of normal quartic polynomials based on global descent directions. SIAM J. Optim. **15**, 275–302 (2004)
16. Smale, S.: Newton's method estimates from data at one point. In: Ewing, R., Gross, K., Martin, C. (eds.) The Merging of Disciplines: New Directions in Pure, Applied and Computational Mathematics, pp. 185–196. Springer, Berlin (1986)
17. Thng, I., Cantoni, A., Leung, Y.H.: Derivative constrained optimum broad-band antenna arrays. IEEE Trans. Signal Process. **41**, 2376–2388 (1993)
18. Thng, I., Cantoni, A., Leung, Y.H.: Analytical solutions to the optimization of a quadratic cost function subject to linear and quadratic equality constraints. Appl. Math. Optim. **34**, 161–182 (1996)