LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-JRNL-652972

An Easy Method to Accelerate an Iterative Algebraic Solver, part II

L. L. LoDestro, J. Yao

April 10, 2014

Journal of Computational Physics

Disclaimer

An Easy Method to Accelerate an Iterative Algebraic Solver, part II

L. L. LoDestro and J. Yao
Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA, 94551-0808
(Dated: April 9, 2014)

This paper is a companion to [Yao, Journal of Computational Physics **267** (2014)]. We provide a compact alternative development of Yao's method to accelerate to order $2n-1$ a fixed-point iterative scheme with an original convergence-rate of order n. Using this approach, we show how to further improve the order of convergence by increasing s, the number of steps per iteration. This scheme extends to arbitrarily high order without extra derivative evaluations per iteration. We discuss the efficiency of the new methods, including $s = 2$ (Yao's method), and compare to that of the original method as a function of n; and we consider the efficiency for systems of equations of system-size M. For $n=2$ and large M, we find that the multi-step scheme peaks in efficiency at about $s \ln s \sim M$ function evaluations, where its computational speed is several times faster than Newton's method.

I. INTRODUCTION

In this paper we expand upon J. Yao's recent paper, 'An Easy Method to Accelerate an Iterative Algebraic Solver," [1]. We provide a compact development of the method, alternative to that in [1], for the acceleration to order $2n-1$ of fixed-point iterative schemes with an original convergence-rate of order n. Using this approach, we show how to further improve the order of convergence by increasing s, the number of steps per iteration. (By "step" is meant a new estimate of the root. $s = 1$ for, e.g., an n^{th}-order Taylor-series based method and $s = 2$ for the modified method in [1].) This scheme extends to arbitrarily high order without extra derivative evaluations per iteration. We discuss the efficiency of the new methods, including $s = 2$ (the method proposed in [1]), and compare to that of the original method as a function of n; and we consider the efficiency for systems of equations of system-size M. For $n=2$ and large M, we find that the multi-step scheme peaks in efficiency at about $s \ln s \sim M$ function evaluations, where its computational speed is several times faster than Newton's method. Finally, by iterating the method itself we generate a family of new methods with similiar performance to the multi-step schemes, but with a different sequence of intermediate points and final step-sizes, which may be of interest in the further development of new methods.

Note that for $M > 1$, all the methods discussed in this paper require one accurate Jacobian inverse at every iteration k.

II. DEPENDENCE OF THE EFFICIENCY OF FUNCTION EVALUATIONS UPON THE ORDER OF CONVERGENCE

It might seem at first that there is a large computational advantage in using high-order methods; but even for scalar f's with simple derivatives this is often not the case. The basic reason is that executing two iterations reduces the error by $\sim (f_k^n)^n \sim f_k^{(n^2)}$, not by f_k^{2n} (k is the iteration index, n the order of convergence); i.e., the exponent of the error increases non-linearly with n, so that fairly rapid reductions are achieved in k even at low n. It can be cheaper to take more iterations than to accomplish more per (more expensive) iteration with fewer iterations. The concept of error-reduction per function-evaluation ($\mathcal{E}_f$ in this paper) was introduced to study the question; it provides a measure of the efficiency of the function-evaluations called for by a given a method. Early mention of $\mathcal{E}_f$ can be found in A. M. Ostrowski [2]; see Refs. [3] and [4] and references therein for fuller discussions. We give a quick review of the concept here and use it in subsequent sections to assess the performance of our modified methods against that of an original n^{th}-order method.

Consider a fixed error-reduction exponent $\mathcal{E}$ after k iterations of an n^{th}-order method: $\mathcal{E} = n^k$. The total number of function-evaluations to reach $\mathcal{E}$ is $N_T = kN_k$, where N_k is the number of function-evaluations required per iteration. ("Function-evaluation" here refers to f as well as to any derivative or partial derivative. For simplicity we assume that the evaluations are all of comparable expense. Adjustments to account for finite-differenced or other numerically computed derivatives are easily made.) Then $\mathcal{E} = n^{N_T/N_k} = (n^{1/N_k})^{N_T}$, so that the reduction per function-evaluation is $\mathcal{E}_f = n^{1/N_k}$. Evaluating $\mathcal{E}_f$ for the original Taylor-series-based root-solver in [1] gives $\mathcal{E}_f = n^{1/n}$; and for the modified method, $\mathcal{E}_f = (2n-1)^{1/(n+1)}$. For large n, the efficiency of both these methods clearly declines with n. We explore $\mathcal{E}_f$ in more detail in Sec. IV, after further generalization of the fundamental acceleration method.

Finally we observe that $\mathcal{E}_f$ is a major determinant of the computational expense C_{CPU} to realize a given $\mathcal{E}$:

$$C_{\text{CPU}} \sim C_{\text{CPU}_f} N_T = C_{\text{CPU}_f} \frac{\ln \mathcal{E}}{\ln \mathcal{E}_f} \xrightarrow[(\mathcal{E}_f - 1) \ll 1]{} C_{\text{CPU}_f} \frac{\ln \mathcal{E}}{\mathcal{E}_f - 1}, \tag{1}$$

where C_{CPU_f} is the cost per function-call. Throughout this paper, we neglect the cost of inverting the Jacobian in estimating C_{CPU}.

III. GENERALIZATION OF THE ACCELERATION SCHEME TO FIXED-POINT ITERATIVE METHODS

Acceleration of $n > 2$ Taylor-series-based root-solvers, while achieving convergence of order $2n-1$ with $n+1$ function-evaluations, has the drawback in common with the original solver that roots of polynomials of degree $n-1$ must be solved for the intermediate and final step-sizes; and the appropriate roots from these solves must be identified. Here we circumvent these complications by developing the acceleration method for fixed-point iteration: One solves for $f(x) = 0$ by iterating upon x_k according to

$$x_{k+1} \;=\; x_k + \delta_k \tag{2}$$

with $\delta_k = \delta(f_k, f_k', f_k'', \dots)$, where $f_k \equiv f(x_k),\ f_k' \equiv f'(x_k),\ f_k'' \equiv f''(x_k)$, etc. Without loss of generality, δ can be written in the form

$$\delta \;=\; -\frac{f}{f'}(1+g)\,, \tag{3}$$

with $g = g(f, f', f'', \dots)$ and $g_k \equiv g(f_k, f_k', f_k'', \dots)$. The iteration scheme is said to be n^{th}-order convergent if

$$f_{k+1} \;\sim\; \mathcal{O}(\delta_k^n)\,. \tag{4}$$

The Taylor expansion of f_{k+1} then becomes (employing the alternate notation $f^{(i)}$ for the i^{th} derivative of f where convenient):

$$f_{k+1} = \sum_{i=0}^{\infty} f_k^{(i)} \frac{\delta_k^i}{i!} = -f_k g_k + \sum_{i=2}^{\infty} \frac{f_k^{(i)}}{i!} \left(\frac{-f_k}{f_k'}\right)^i (1+g_k)^i\,. \tag{5}$$

In the second equality, note that $f_k g_k$, the remainder of the two lowest-order terms, can be at most of second order (since $f_{k+1} \sim \mathcal{O}(\delta_k^n)$), which then implies $f_k \sim \delta_k$, which in turn restricts g_k to at most first order. The series is in the form of a function, h, that is analyzed in the Appendix. It is shown there what conditions g must satisfy in order that $f_{k+1} \sim \mathcal{O}(\delta_k^n)$.

We now apply our acceleration approach to this fixed-point iteration. The functions f, g, and δ will remain the same. To distinguish the modified iterates, we will use use tilde's, i.e., $\tilde\delta_k \equiv \delta(\tilde f_k, \tilde f_k{}', \tilde f_k{}'' \dots),\ \tilde f_k \equiv f(\tilde x_k),\ \tilde f_k{}' \equiv f'(\tilde x_k),\ \tilde g_k \equiv g(\tilde f_k, \tilde f_k{}', \tilde f_k{}'' \dots)$, etc. We set

$$\tilde x_{k+1} = \tilde x_k + \Delta_k$$

with

$$\Delta_k = -\frac{\tilde f_k + f^*}{\tilde f_k{}'}(1+g^*)$$
$$g^* \equiv g(\tilde f_k + f^*, \tilde f_k{}', \tilde f_k{}'' \dots)$$

and

$$f^* \equiv f(\tilde x_k + \tilde\delta_k) = \sum_{i=0}^{\infty} \tilde f_k{}^{(i)} \frac{\tilde\delta_k^i}{i!} = -\tilde f_k \tilde g_k + \sum_{i=2}^{\infty} \frac{\tilde f_k{}^{(i)}}{i!} \left(\frac{-\tilde f_k}{\tilde f_k{}'}\right)^i (1+\tilde g_k)^i\,.$$

We then have

$$\tilde f_{k+1} = \sum_{i=0}^{\infty} \tilde f_k{}^{(i)} \frac{\Delta_k^i}{i!} = -f^* - (\tilde f_k + f^*) g^* + \sum_{i=2}^{\infty} \frac{\tilde f_k{}^{(i)}}{i!} \left(\frac{-(\tilde f_k + f^*)}{\tilde f_k{}'}\right)^i (1+g^*)^i\,. \tag{6}$$

In the second equality, observe that both f^* and the remaining terms are each in the form of the function h introduced as the right-hand-side of Eq. (5). Thus the results of the Appendix can be applied to them. This gives their scalings as order $\tilde f_k{}^n$ and $(\tilde f_k + f^*)^n \sim \tilde f_k{}^n$ respectively, and Eq. (A7) is then used to obtain

$$\tilde f_{k+1} = \mathcal{O}(f^* \times \max(f^*, \tilde f_k)^{n-1}) = \mathcal{O}(\tilde f_k{}^{2n-1}).$$

As in Ref. [1], with a cost of a single additional function-evaluation, the convergence of the original n^{th}-order method has been raised to $2n-1$.

To complete contact with Ref. [1], we also calculate

$$(\Delta_k - \tilde{\delta}_k)\tilde{f}_k{}' = (\tilde{g}_k - g^*)\tilde{f}_k - (1+\tilde{g}_k)f^* \sim -\tilde{f}_k\tilde{g}^{(1)}f^* - f^* \sim f^* \sim \mathcal{O}(\tilde{\delta}_k^n).$$

IV. INCREASING THE CONVERGENCE-RATE FURTHER WITHOUT ADDITIONAL DERIVATIVE EVALUATIONS

In this section we consider additional function evaluations (but no additional derivative evaluations) in going from f_k to f_{k+1} with a fixed-point iterative method. Our primary interest is to accelerate schemes for large systems (which typically call for $n = 2$ to avoid the expense of derivatives beyond the first), but we begin with general n and with a scalar f for simplicity (the generalization to $M > 1$ is straightforward).

We begin with some new notation. The subscript k will be dropped; all quantities should be understood to be at iteration k unless explicitly noted otherwise. Similarly the tilde ($\tilde{}$), denoting a modified method's iterates (as opposed to the original's, i.e., with Taylor series (5)) will be dropped. We identify step quantities with a bar ($\bar{}$) and the step index with a capitalized Roman-numeral superscript; and we define s to be the number of steps. We map our new variables onto the accelerated fixed-point method of Sec. III:

$$\begin{aligned}
\bar{x}^{\mathrm{S_0}} &\leftrightarrow \tilde{x}_k \\
\bar{f}^{\mathrm{S_0}} &\leftrightarrow \tilde{f}_k \\
\bar{A}^{\mathrm{S_0}} &\leftrightarrow \tilde{f}_k \\
\bar{g}^{\mathrm{S_0}} &\leftrightarrow \tilde{g}_k = g(\tilde{f}_k, \tilde{f}_k{}', \tilde{f}_k{}'' \dots) \\
\bar{\delta}^{\mathrm{S_0}} &\leftrightarrow \tilde{\delta}_k = -(\tilde{f}_k/\tilde{f}_k{}') \times (1+\tilde{g}_k)
\end{aligned}$$

$$\begin{aligned}
\bar{x}^{\mathrm{S_I}} &\leftrightarrow x^* = \tilde{x}_k + \tilde{\delta}_k \\
\bar{f}^{\mathrm{S_I}} &\leftrightarrow f^* = f(x^*) \\
\bar{A}^{\mathrm{S_I}} &\leftrightarrow \tilde{f}_k + f^* \\
\bar{g}^{\mathrm{S_I}} &\leftrightarrow g^* = g(\tilde{f}_k + f^*, \tilde{f}_k{}', \tilde{f}_k{}'' \dots) \\
\bar{\delta}^{\mathrm{S_I}} &\leftrightarrow \Delta_k = -((\tilde{f}_k + f^*)/\tilde{f}_k{}') \times (1+g^*)
\end{aligned}$$

$$\begin{aligned}
\bar{x}^{\mathrm{S_{II}}} &\leftrightarrow \tilde{x}_{k+1} = \tilde{x}_k + \Delta_k \\
\bar{f}^{\mathrm{S_{II}}} &\leftrightarrow \tilde{f}_{k+1} = f(\tilde{x}_{k+1}),
\end{aligned}$$

where we have introduced the new variable $\bar{A}^{\mathrm{S_i}} \equiv \sum_{j=0}^{i} \bar{f}^{\mathrm{S_j}}$.

We write the new scheme (omitting updates of quantities which simply follow their definitions, such as $\bar{f}^{\mathrm{S_{i+1}}} = f(\bar{x}^{\mathrm{S_{i+1}}})$):

$$\begin{aligned}
\bar{x}^{\mathrm{S_0}} &= \bar{x} \\
\bar{g}^{\mathrm{S_0}} &= g(\bar{A}^{\mathrm{S_0}}, f', f'', \dots) \\
\bar{\delta}^{\mathrm{S_0}} &= -(\bar{A}^{\mathrm{S_0}}/f') \times (1+\bar{g}^{\mathrm{S_0}})
\end{aligned}$$

$$\begin{aligned}
\bar{x}^{\mathrm{S_i}} &= \bar{x} + \bar{\delta}^{\mathrm{S_{i-1}}} \\
\bar{g}^{\mathrm{S_i}} &= g(\bar{A}^{\mathrm{S_i}}, f', f'', \dots) \\
\bar{\delta}^{\mathrm{S_i}} &= -(\bar{A}^{\mathrm{S_i}}/f') \times (1+\bar{g}^{\mathrm{S_i}})
\end{aligned}$$

$$\begin{aligned}
\bar{x}^{\mathrm{S}_s} &= \bar{x} + \bar{\delta}^{\mathrm{S}_{s-1}} \\
x_{k+1} &= \bar{x}^{\mathrm{S}_s},
\end{aligned} \tag{7}$$

giving for the Taylor expansion of f_{k+1}:

$$f_{k+1} = \sum_{i=0}^{\infty} f^{(i)} \frac{(\bar{\delta}^{\mathrm{S}_{s\text{-}1}})^i}{i!} = f - (1+\bar{g}^{\mathrm{S}_{s\text{-}1}})\bar{A}^{\mathrm{S}_{s\text{-}1}} + \sum_{i=2}^{\infty} \frac{f^{(i)}}{i!} \left(\frac{-\bar{A}^{\mathrm{S}_{s\text{-}1}}}{f'}\right)^i \left(1+\bar{g}^{\mathrm{S}_{s\text{-}1}}\right)^i = f - \bar{A}^{\mathrm{S}_{s\text{-}1}} + \bar{h}^{\mathrm{S}_{s\text{-}1}}. \tag{8}$$

To proceed, we develop an iterative relation for the steps from the intermediate Taylor series, beginning with:

$$\bar{f}^{\mathrm{S_{i+1}}} = f - \bar{A}^{\mathrm{S_i}} + \bar{h}^{\mathrm{S_i}}.$$

Subtracting the series for $\bar{f}^{\mathrm{S_i}}$ gives:

$$\bar{f}^{\mathrm{S_{i+1}}} - \bar{f}^{\mathrm{S_i}} = -\bar{A}^{\mathrm{S_i}} + \bar{A}^{\mathrm{S_{i\text{-}1}}} + \bar{h}^{\mathrm{S_i}} - \bar{h}^{\mathrm{S_{i\text{-}1}}},$$

so that

$$\bar{f}^{\mathrm{S_{i+1}}} = \bar{h}^{\mathrm{S_i}} - \bar{h}^{\mathrm{S_{i\text{-}1}}} \sim (\bar{A}^{\mathrm{S_i}} - \bar{A}^{\mathrm{S_{i\text{-}1}}})\,\mathcal{O}(f^{n-1}) \sim \bar{f}^{\mathrm{S_i}}\,\mathcal{O}(f^{n-1}) \sim \mathcal{O}(f^{(i+1)(n-1)+1}),$$

where we have used Eq. (A7). Using this result in Eq. (8), we obtain the convergence rate:

$$f_{k+1} \sim \mathcal{O}(f^{s(n-1)+1}), \tag{9}$$

having evaluated one derivative and s f-functions.

The error-reduction per function-evaluation of this scheme, assuming f' costs about the same as f to evaluate, is $\mathcal{E}_f = (s(n-1)+1)^{1/(n+s-1)}$. Evaluations of $\mathcal{E}_f$ at small s and n reveal a single maximum, $\mathcal{E}_f = 1.495$ at $s=2$, $n=3$, i.e., at the first case analyzed in [1]—Halley's scheme accelerated to 5^{th}-order. For comparison, the original Halley's scheme, $s=1$, $n=3$, has $\mathcal{E}_f = 1.442$.

We note that for $n=2$ and a scalar f, this multi-step method can be found in [4]. The order of convergence is proved there in a different way—by making use of the mean-value theorem—which may be of interest.

A. Accelerating Newton's method for systems of equations

Our interest is primarily in system-size $M > 1$, so we now specialize to $n=2$—Newton's method—in order to limit the expense of derivative evaluations to M^2 scaling. At large M the trends in $\mathcal{E}_f$ are dramatically altered from the observations following Eq. (9). The efficiency measure becomes

$$\mathcal{E}_f = (s+1)^{1/(sM+M^2)} \xrightarrow[1 \ll s \ll M]{} s^{1/M^2},$$

from which it is readily seen that there is substantial improvement with increasing s and an eventual decline at $s > M$; and that as $M \to \infty$, $\mathcal{E}_f \to 1$, i.e., the per-call improvement becomes negligible. Setting $d\mathcal{E}_f/ds = 0$ to optimize the efficiency yields the approximate solution $s \ln s \sim M$.

In Fig. 1 we illustrate the efficiency of an s-step method's function-evaluations as a function of s and system-size M, compared to Newton's method: We plot constant contours of $(\mathcal{E}_f(s,M)-1)/(\mathcal{E}_f(1,M)-1)$. The figure shows that for large M, the introduction of additional steps s at each iteration k offers a significant advantage: Referring to Eq. (1), we see that for $M \sim 10$–50, C_{CPU} is reduced by a factor of two to three.

B. Two examples

Next we present two examples of the scheme (7) with $n = 2$, $s = 3$, and $M = 32$. In order to illustrate the performance of the scheme as it may often occur in practice, we use 8-byte arithmetic and, to further control precision, invert the Jacobian matrices ourselves.

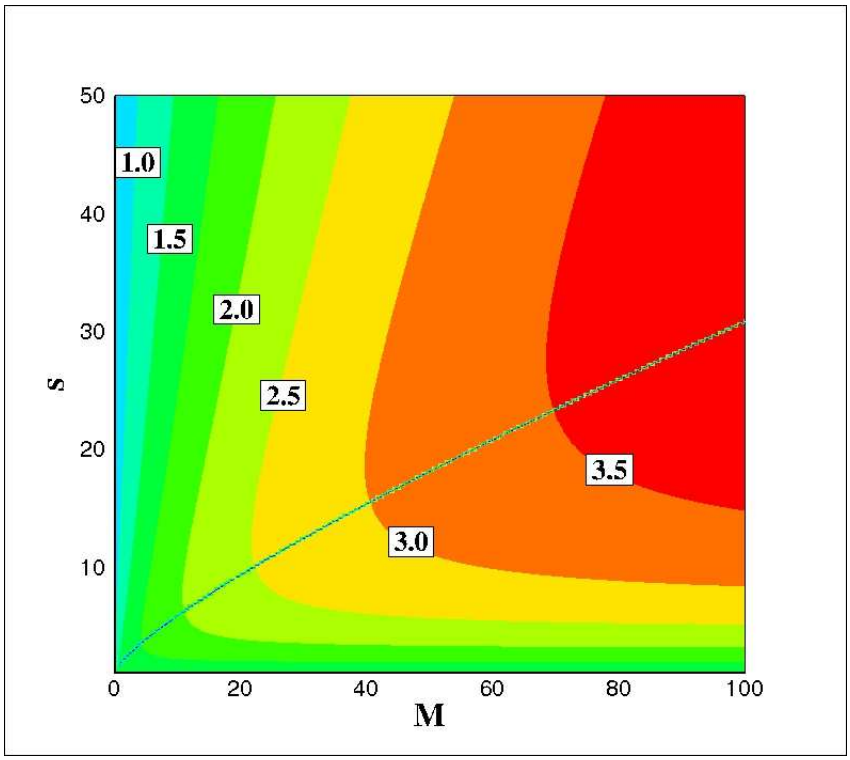

FIG. 1: The color-map of the relative efficiency metric (see text for its definition) for the accelerated Newton scheme, Eqs. (7), with $n=2$. The dotted blue curve plots the approximate s to maximize $\mathcal{E}_f$, $M = s \ln s$.

A TRI-DIAGONAL SYSTEM

We choose the following set of equations:

$$\begin{aligned} x_1 + \frac{1}{2}\sin(x_2) &= 1.0; \\ \frac{1}{2}\sin(x_1) + x_2 + \frac{1}{2}\sin(x_3) &= 0; \\ \cdots \quad \cdots \quad \cdots & \\ \frac{1}{2}\sin(x_{i-1}) + x_i + \frac{1}{2}\sin(x_{i+1}) &= 0; \\ \cdots \quad \cdots \quad \cdots & \\ \frac{1}{2}\sin(x_{M-1}) + x_M &= 0. \end{aligned} \tag{10}$$

This results in a tri-diagonal Jacobian matrix, which we invert by an explicit back-substitution. The L_1 residues obtained with Newton's method and the accelerated scheme are plotted in Fig. 2. The initial guess is set to $x_i = 1/2$ for each variable. Until the final iterations, which have reached machine accuracy, the points follow their predicted convergence rates.

A POSITIVE DEFINITE SYMMETRIC SYSTEM

This equation system is described by the following equation-set for $i = 1, 2, ...M$

$$M x_i + \sum_{j=1}^{M} \frac{\sin(x_i + x_j)}{i + j - 1} = M + \sum_{j=1}^{M} \frac{\sin(\frac{1}{i} + \frac{1}{j})}{i + j - 1}. \tag{11}$$

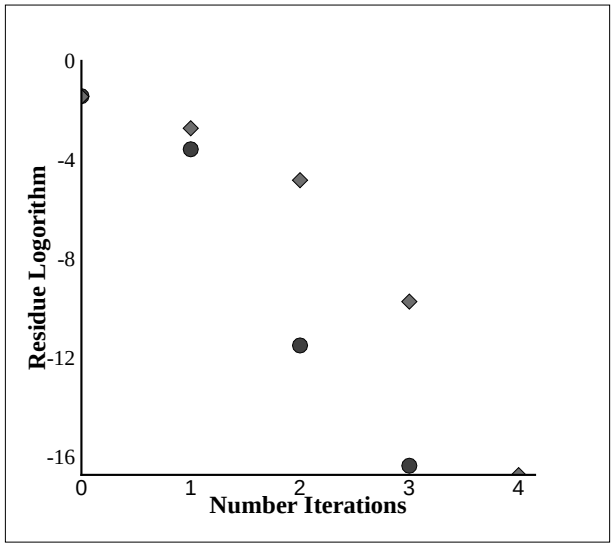

FIG. 2: The convergence map (residue logarithm vs. number of iterations) of Newton's method (diamonds, following a parabola) and the accelerated method (circles, following a cubic curve) for the equation system (10).

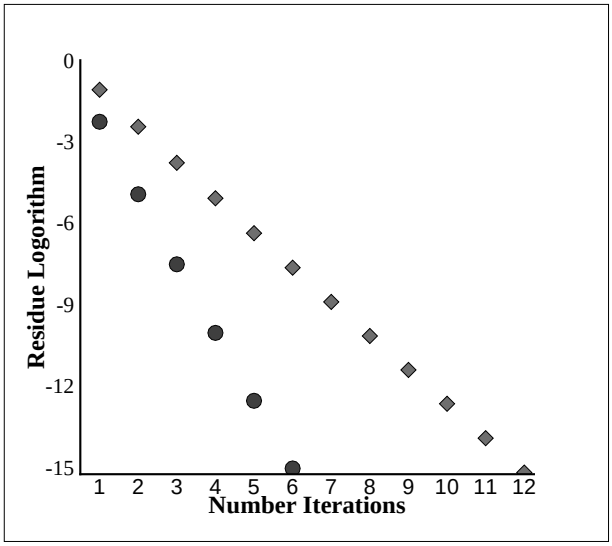

FIG. 3: The convergence map of Newton's method and the accelerated method for the equation system (11). Note the linear convergence rates introduced by a finite accuracy of the matrix decomposition.

This system has the solution $x_i = 1/i$ for all i. We employ the Cholesky algorithm to decompose the Jacobian. The initial guess is the solution plus a random perturbation of up to 50% of the magnitude of each root. The L_1 residues obtained with Newton's method and the accelerated scheme are plotted in Fig. 3. Here we see that successive iterates are in a fixed ratio, i.e., we have obtained only linear convergence for each of the schemes. This is because the condition number of the Jacobian matrix is large and, given the precision of our calculation, our inverse inaccurate. But we nevertheless include this example, because it shows that our accelerated scheme still outperforms the standard scheme by a factor of two in convergence rate while only slightly increasing the cost $\big(M(2{+}M)$ compared to $M(1{+}M)$ function evaluations per iteration$\big)$.

V. MULTIPLE APPLICATIONS OF THE ACCELERATION METHOD

Observing that the accelerated scheme given in Sec. III can itself be cast as an "original" method with convergence rate $n_1 = 2n_0 - 1$, we consider accelerating it to achieve a convergence rate $n_2 = 2n_1 - 1 = 2(2n_0 - 1) - 1$, to be followed by successive such iterations. Here we have defined $n_0 \equiv n$, the convergence rate of the fixed-point method given by Eqs. (2)–(4). After ν accelerations, the order of convergence would be:

$$n_\nu = 2^\nu n_0 - 2^{\nu-1} \cdots - 2^1 - 1 = 2^\nu n_0 - \sum_{i=0}^{\nu-1} 2^i = 2^\nu (n_0 - 1) + 1.$$

The procedure appears to have promise: It results in a rapid increase in the order of convergence while calling for one additional function evaluation per method iteration (cf. Ref. [3], Sec. 5). Furthermore, it is easily generalizable to $M > 1$, whereas the method in [3] entails rational functions of f. We will find, however, that the number of function evaluations is more than it first appears, and the procedure results in methods with $\mathcal{E}_f$ comparable to the multi-step methods of Sec. IV.

Each application of the acceleration method analyzed in Sec. III generates a new method, which takes the form:

$$\begin{aligned}
\delta_\nu &= -\frac{f}{f'}(1+g_\nu)\,,\\
f^*_\nu &\equiv f(x+\delta_\nu) = h_\nu(f) = \mathcal{O}(f^{n_{\nu-1}})\\
g^*_\nu &\equiv g_\nu(f+f^*_\nu, f', f'', \dots)\\
\Delta_\nu &= -\frac{f+f^*_\nu}{f'}(1+g^*_\nu) = \delta^*_\nu\\
f_{k+1} &= f(x+\Delta_\nu).
\end{aligned}$$

All quantities here are understood to be evaluated at step k of the particular method unless explicitly noted otherwise. The function g_ν (and therefore h_ν), which determines the order of the "original" method, changes with each application of the method, of course. We obtain g_ν by setting the functional form of δ_ν to that of $\Delta_{\nu-1}$, which is already known and which gives a method whose order of convergence is $n_{\nu-1}$. Then

$$g_\nu = [(f+f^*_{\nu-1})g^*_{\nu-1} + f^*_{\nu-1}]/f.$$

$\nu=1$ is the accelerated method given in Sec. III, with $g_1 = g$ specifying the given n_0^{th}-order (unaccelerated) method.

The Taylor expansion of f_{k+1} is in a form identical to that of Eq. (6), so we can immediately write:

$$f_{k+1} = -h_\nu(f, f', f'', \dots) + h_\nu(f+f^*_\nu, f', f'', \dots) = f^*_\nu \times \mathcal{O}(f^{n_{\nu-1}-1}) = \mathcal{O}(f^{2n_{\nu-1}-1}).$$

In developing the successive g^*_ν, quantities such as $f^*_{\nu-1}\Big|^{f\to f+f^*_\nu}$ will arise. For this we may use the following identity:

$$f(x+\gamma)\big|^{f\to f+\phi} = \left[\sum_{i=0}^{\infty} f^{(i)}\frac{\gamma^i}{i!}\right]\Bigg|^{f\to f+\phi} = \left[f + \sum_{i=1}^{\infty} f^{(i)}\frac{\gamma^i}{i!}\right]\Bigg|^{f\to f+\phi} = \phi + f(x+\gamma|^{f\to f+\phi})\,.$$

To procede with general ν, we introduce the shorthand notation $|^{+\nu}$ for $|^{f\to f+f^*_\nu}$, and obtain a recurrence relation for g^*_ν:

$$\begin{aligned}
(f+f^*_\nu)g^*_\nu &= [\{(f+f^*_{\nu-2})g^*_{\nu-2} + f^*_{\nu-2}\}|^{+(\nu-1)} + f^*_{\nu-1}]|^{+\nu}\\
&= \{(f+f^*_1)g^*_1 + f^*_1\}|^{+2}|^{+3}|^{+\cdots}|^{+\nu} + f^*_2|^{+3}|^{+\cdots}|^{+\nu} + \cdots + f^*_{\nu-2}|^{+(\nu-1)}|^{+\nu} + f^*_{\nu-1}|^{+\nu},
\end{aligned}$$

after which

$$-f'\Delta_\nu = \{(f+f^*_1)g^*_1 + f^*_1\}|^{+2}|^{+3}|^{+\cdots}|^{+\nu} + f^*_2|^{+3}|^{+\cdots}|^{+\nu} + \cdots + f^*_{\nu-2}|^{+(\nu-1)}|^{+\nu} + f^*_{\nu-1}|^{+\nu} + f^*_\nu + f.$$

We illustrate with $\nu=2$—i.e., with one additional acceleration step after the scheme of Sec. III—and $n_0=2$. Then $g_1 =$ Sec. III's $g = 0$, $g_2 = f^*_1/f$, and

$$-f'\Delta_2 = f^*_1|^{+2} + f^*_2 + f = f(x+\delta_1|^{+2}) + 2f^*_2 + f = f(x - \tfrac{f+f^*_2}{f'}) + 2f^*_2 + f,$$

with $-f'\delta_2 = f + f^*_1$. Thus we find we have needed two additional f-evaluations compared to the $\nu=1$ method, for a total of four, to raise the order of convergence from $2^1+1=3$ to $2^2+1=5$. The performance of this method is then the same as that of the $s=4$ multi-step method, although both the distribution of the intermediate x points and the final x_{k+1} differ between the two methods. Similarly, but with considerably more algebra, we find $\nu=3$ requires a total of eight f-evaluations, providing the same order of convergence as with $s=8$.

VI. SUMMARY AND DISCUSSION

In this paper we have extended the method to accelerate non-linear root-solvers proposed in Ref. [1] to an arbitrary number of steps s per iteration; and, with a simplifying assumption for derivatives, we have estimated the error reduction per function evaluation, $\mathcal{E}_f$, and assessed the computational efficiency C_{CPU} of the methods.

For a scalar problem, we find $\mathcal{E}_f = \big(s(n-1)+1\big)^{1/(n+s-1)}$. for the multi-step scheme. For an original n^{th}-order method ($s=1$) this gives $\mathcal{E}_f = n^{1/n}$; and for Yao's proposed accelerated method ($s=2$), $\mathcal{E}_f = (2n-1)^{1/(n+1)}$. The per-function-evaluation improvement of the proposed method for scalar problems is then rather small (see details just

above Sec. IV A). For general system-size M, however, there is substantial improvement to be gained. Restricting attention to $n = 2$, we have $\mathcal{E}_f = (s+1)^{1/(sM+M^2)}$, so that for $sM \ll M^2$, $\mathcal{E}_f$ increases with s. Optimizing in s gives $s \ln s \sim M$. For M in the range 10–50, we find that the CPU cost is reduced by a factor of two to three.

We have illustrated the latter result with two examples, beginning with Newton's method and using $M = 32$ in both cases. Our second example produced an ill-conditioned Jacobian matrix, which compromised the accuracy of our direct inversion; while it did not achieve the theoretical convergence-order, it converged twice as fast as the original Newton method. It is often the case that the high precision required for higher-order methods is unobtainable; this example suggests the possibility of a cheap way to nevertheless obtain more rapid convergence by going to higher order even when an algorithm's performance is already (in the lower order scheme) damaged by noise. It remains to analyze the effects of the noise on the scheme (or the general-s multi-step scheme); but we conjecture that these schemes, although delivering only linear convergence, may be of use for $M > 1$ systems where straightforward functional iteration is difficult to stabilize.

Finally, by iterating the ($s = 2$) acceleration method itself, we have shown how to generate a new family of methods which very rapidly increase the order of convergence with succesive iterations while remaining easy to generalize to $M > 1$. Although the $\mathcal{E}_f$ for the first few iterations equal those for the multi-step scheme of the same order, the sequences of intermediate points and final step-sizes differ for the two families, and this appraoch may be of interest in the further development of new methods.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

APPENDIX A: THE FUNCTION h

In this appendix we consider the function

$$h(z_0, z_1, \ldots, z_m, \ldots) \equiv -z_0\, g + \sum_{i=2}^{\infty} \frac{z_i}{i!} \left(\frac{-z_0}{z_1}\right)^i (1+g)^i \,, \tag{A1}$$

where $g = g(z_0, z_1, \ldots, z_m)$ is the same function of its arguments that appears in Sec. III, so that h can be expanded in a Taylor series in z_0. It is assumed again that $z_1 \neq 0$. (It is not assumed here, however, that the z_i are interrelated.)

We first expand g and $(1+g)^i$ in Taylor series in z_0:

$$\begin{aligned} g &= \sum_{j=0}^{\infty} \frac{z_0^j}{j!} g^{(j)} \\ (1+g)^i &= \sum_{l=0}^{\infty} z_0^l\, C_{l,i} \,, \end{aligned} \tag{A2}$$

where we have defined $g^{(i)}(z_1, \ldots, z_m) \equiv \frac{\partial^i g}{\partial z_0^i}\Big|^{z_0=0}$ and introduced the functions $C(z_1, \ldots, z_m)_{l,i}$. Explicit expressions for the $C_{l,i}$ in terms of the $g^{(j)}$ will not be needed here. The important point is that neither the $g^{(j)}$ nor the $C_{l,i}$ functions depend on z_0. Equation (A1) becomes

$$h(z_0, z_1, \ldots) = -z_0 \sum_{i=0}^{\infty} \frac{z_0^i}{i!} g^{(i)} + \sum_{i=2}^{\infty} \frac{z_i}{i!} \left(\frac{-1}{z_1}\right)^i \sum_{l=0}^{\infty} z_0^{l+i}\, C_{l,i} \,.$$

We change variables from l to $s = l + i$:

$$h = -z_0 \sum_{i=0}^{\infty} \frac{z_0^i}{i!} g^{(i)} + \sum_{i=2}^{\infty} \frac{z_i}{i!} \left(\frac{-1}{z_1}\right)^i \sum_{s=i}^{\infty} z_0^s\, C_{s-i,i} \,.$$

Next we invert the order of the double sum, obtaining

$$h = -z_0 \sum_{i=0}^{\infty} \frac{z_0^i}{i!} g^{(i)} + \sum_{s=2}^{\infty} z_0^s \sum_{i=2}^{s} \frac{z_i}{i!} \left(\frac{-1}{z_1}\right)^i C_{s-i,i} \,. \tag{A3}$$

For the remainder of this appendix we restrict attention to the case when it is known that $h = 0$ through order $n-1$, with z_0 the ordering parameter. In this case the coefficients of the powers of z_0 through $n-1$ in Eq. (A3) must vanish separately. From this we find requirements for g's dependence on its first argument in terms of its other arguments:

$$0 = g^{(0)} \tag{A4}$$

$$0 = -\frac{g^{(j-1)}}{(j-1)!} + \sum_{i=2}^{j} \frac{z_i}{i!} \left(\frac{-1}{z_1}\right)^i C_{j-i,i}\,, \qquad 2 \le j \le n-1. \tag{A5}$$

Observe that Eq. (A4) reproduces Sec. III's result $g_k \sim f_k$, argued from considerations of the first few terms of the Taylor series (5). It follows from $g^{(0)} = 0$ and Eq. (A2) that $C_{0,i} = 1$ for all i.

Equations (A5) are easily solved for the $g^{(i)}$ in terms of $z_1, \ldots, z_m$: Referring to Eq. (A2), we see that for any i, $C_{l,i}$ is a function of only those $g^{(j)}$'s with j in the range $1, \ldots, l$. The $C_{j-i,i}$ in Eq. (A5), which range from $C_{1,-}$ to $C_{j-2,-}$ (omitting $C_{0,-}$ since it equals unity), therefore bring in $g^{(j)}$'s only with j in the range $1, \ldots, j-2$. So the conditions that $g^{(j)}$ must satisfy separate; they can be derived one at a time, starting from $j-1 = 1$. For $j = 2$, for example, the requirement is $g^{(1)} = z_2/(2z_1^2)$, as is also quickly seen from the lowest-order expansion of Eq. (A1). We are left with

$$h = \sum_{j=n}^{\infty} z_0^j \left[-\frac{g^{(j-1)}}{(j-1)!} + \sum_{i=2}^{j} \frac{z_i}{i!} \left(\frac{-1}{z_1}\right)^i C_{j-i,i} \right] \sim \mathcal{O}(z_0^n)\,. \tag{A6}$$

Finally we note that

$$\begin{aligned} h(y, z_1, \ldots) - h(z_0, z_1, \ldots) &\sim (y - z_0)\, \mathcal{O}(y^{n-1}, z_0^{n-1}) \left[-\frac{g^{(n-1)}}{(n-1)!} + \sum_{i=2}^{n} \frac{z_i}{i!} \left(\frac{-1}{z_1}\right)^i C_{n-i,i} \right] \\ &= \mathcal{O}((y - z_0) \times \max(y, z_0)^{n-1}). \end{aligned} \tag{A7}$$

Comments:

The conditions (A5) are independent of z_0. If it is known, as it is for the Taylor series of the original n^{th}-order method, Eq. (5) in Sec. III, that h vanishes through $\mathcal{O}(n-1)$ for a particular argument set $\{z_0, z_1, \ldots, z_m, \ldots\} = \{f_k, f_k^{(1)}, f_k^{(2)}, \ldots, f_k^{(m)}, \ldots\}$, then Eqs. (A5) will be satisfied for *any* f in $h(f, f_k^{(1)}, f_k^{(2)}, \ldots, f_k^{(m)}, \ldots)$.

Similarly, though no longer constrained, the square-bracketed term in Eq. (A6) depends only on k and j and is of order unity.

For series including h where the order of interest is $n-1$ and in which $f^{(n-1)}$ does not vanish, $f^{(n-1)}$ must appear in h; then $m \ge n-1$ is required.

[1] J. Yao, Journal of Computational Physics **267**, 139 (2014), URL `http://dx.doi.org/10.1016/j.jcp.2014.02.027`.

[2] A. M. Ostrowski, *Solution of Equations and Systems of Equations* (Academic Press, New York, 1966), 2nd ed.

[3] H. T. Kung and J. F. Traub, Journal of the Association for Computing Machinery **21**, 643 (1974).

[4] R. Wait, *The Numerical Solution of Algebraic Equations* (John Wiley & Sons, Chichester, 1979).