Full linear multistep methods as root-finders

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computer Science

CASA-Report 17-05
February 2017

Full linear multistep methods as root-finders

by

B.S. van Lith, J.H.M. ten Thije Boonkkamp, W.L. IJzerman

Centre for Analysis, Scientific computing and Applications
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven, The Netherlands
ISSN: 0926-4507

Full linear multistep methods as root-finders

Bart S. van Lith[a,1], Jan H.M. ten Thije Boonkkamp[a], Wilbert L. IJzerman[a,b]

[a] *Department of Mathematics and Computer Science, Eindhoven University of Technology - P. O. Box 513, NL-5600 MB Eindhoven, The Netherlands.*
[b] *Philips Lighting - High Tech Campus 44, 5656 AE, Eindhoven, The Netherlands.*

Abstract

Root-finders based on full linear multistep methods (LMMs) use previous function values, derivatives and root estimates to iteratively find a root of a nonlinear function. As ODE solvers, full LMMs are typically not zero-stable. However, used as root-finders, the interpolation points are convergent so that stability issues are circumvented. A general analysis is provided based on inverse polynomial interpolation, which is used to prove a fundamental barrier on the convergence rate of any LMM-based method. We show, using numerical examples, that full LMM-based methods perform excellently. Finally, we also provide a robust implementation based on Brent's method that is guaranteed to converge.

Keywords: Root-finder, nonlinear equation, linear multistep methods, iterative methods, convergence rate.

1. Introduction

Suppose we are given a sufficiently smooth nonlinear function $f : \mathbb{R} \to \mathbb{R}$ and we are asked to solve the equation

$$f(x) = 0. \tag{1}$$

This archetypical problem is ubiquitous in all fields of mathematics, science and engineering. For example, ray tracing techniques in optics and computer graphics need to accurately calculate intersection points between straight lines, rays, and objects of varying shapes and sizes [1, 2]. Implicit ODE solvers are often formulated like (1), after which a root-finder of some kind is applied [3].

Depending on the properties of the function f, there are several methods that present themselves. Sometimes the derivative is not available for various reasons, in which case the secant method will prove useful. If higher-order convergence is desired, inverse quadratic interpolation may be used [4]. If the derivative of f exists and is available, Newton's method is a solid choice, especially if f is also convex.

Email address: `b.s.v.lith@tue.nl` (Bart S. van Lith)
[1] Corresponding author

Recently, a new interpretation of root-finding methods in terms of ODEs has been introduce by Grau-Sánchez et al. [5, 6, 7]. Their idea is to consider the inverse function derivative rule as an ODE, so that any explicit ODE solver may be converted to a root-finding method. Indeed, Grau-Sánchez et al. have successfully introduced root-finders based on Adams-type multistep and Runge-Kutta integrators. It goes without saying that only explicit ODE solvers can be usefully converted to root-finding methods. However, predictor-corrector pairs are possible, as those methods are indeed explicit.

We argue that the ODE approach can be interpreted as inverse interpolation with (higher) derivatives. Indeed, any linear integration method is based on polynomial interpolation. Thus, the ODE approach can be seen as a generalisation of inverse interpolation methods such as the secant method or inverse quadratic interpolation. The analysis can thus be combined into a single approach based on inverse polynomial interpolation.

Our main theoretical result is a theorem on the convergence rate of root-finders based on explicit linear multistep methods. We furthermore prove a barrier on the convergence rate of LMM-based root-finders. It turns out that adding a few history points quickly boosts the convergence rate close to the theoretical bound. However, adding many history points ultimately proves an exercise in futility due to diminishing returns in the convergence rate. Two LMM-based methods are constructed explicitly, one using two history points with a convergence rate of $1 + \sqrt{3} \approx 2.73$ and another with three history points that converges with rate 2.91.

Using several numerical examples, we show that the LMM-based methods indeed achieve this higher convergence rate. Furthermore, pathological examples where Newton's method fails to converge are used to show increased stability. We also construct a robust LMM-based method combined with bisection to produce a method that can been seen as an extension of Brent's [8]. Similar to Brent's method, whenever an enclosing starting bracket is provided, an interval $[a, b]$ with $f(a)f(b) < 0$, our method is guaranteed to converge.

This article is organised in the following way. First, we find the convergence rate of a wide class of root-finders in Section 2 and prove a barrier on the convergence rates. Next, in Section 3 we derive new root-finders based on full linear multistep methods and show that such methods are stable when the initial guess is sufficiently close to the root. After this, some results are presented in Section 4 that verify our earlier theoretical treatment. Finally, we present our robust implementation in Section 5, after which we give our conclusions in Section 6.

2. Barriers on LMM root-finders

Root-finding methods based on the ODE approach of Grau-Sánchez et al. can be derived by assuming that the function f is sufficiently smooth and invertible in the vicinity of the root. Under these assumptions, the chain rule gives

$$\frac{\mathrm{d}x}{\mathrm{d}y} = [f^{-1}]'(y) = \frac{1}{f'(x)} = F(x), \tag{2}$$

which we may interpret as an autonomous ODE for the inverse. Integrating (2) from an initial guess $y_0 = f(x_0)$ to $y = 0$ yields

$$f^{-1}(0) = x_0 + \int_{y_0}^{0} F\big(x(y)\big)\,\mathrm{d}y. \tag{3}$$

Immediately, we see that applying the forward Euler method to (2) gives Newton's method. From (3), we see that the step size of the integrator should be taken as $0 - y_0 = -f(x_0)$. However, Newton's method may also be interpreted as an inverse linear Taylor method, i.e., a method where the inverse function is approximated by a first-order Taylor polynomial. Indeed, any linear numerical integration method applied to (2) can be interpreted as an inverse polynomial interpolation.

As such, explicit linear multistep methods applied to (2) will also produce a polynomial approximation to the inverse function. Such a method has the form

$$x_{n+s} + \sum_{k=0}^{s-1} a_k^{(n)} x_{n+k} = h_{n+s} \sum_{k=0}^{s-1} b_k^{(n)} F(x_{n+k}), \tag{4}$$

where indeed $b_s^{(n)} = 0$, otherwise we end up with an implicit root-finder, which would not be very useful. The coefficients of the method, $\{a_k^{(n)}\}_{k=0}^{s-1}$ and $\{b_k^{(n)}\}_{k=0}^{s-1}$, will depend on the previous step sizes and will therefore be different each step. The step sizes are given by $h_{n+k} = y_{n+k} - y_{n+k-1}$, the differences in y-values. Since we wish to find the root, we set $y_{n+s} = 0$, leading to $h_{n+s} = y_{n+s} - y_{n+s-1} = -y_{n+s-1}$. Furthermore, the y-values are of course given by the function values of the root estimates, i.e.,

$$h_{n+k} = f(x_{n+k}) - f(x_{n+k-1}) \quad \text{for } k = 1, \dots, s-1. \tag{5}$$

Like an ODE solver, we may use an implicit LMM in tandem with an explicit LMM to form a predictor-corrector pair, the whole forming an explicit method. Unlike an ODE solver, we may construct derivative-free root-finders based on the LMM approach by setting all $b_k^{(n)} = 0$ for $k = 0, \dots s-1$ and for all $n > 0$, e.g., the secant method. For an ODE solver this would obviously not make sense. Similar to ODE solvers, we may introduce higher derivatives by using

$$x_{n+s} + \sum_{k=0}^{s-1} a_k^{(n)} x_{n+k} = h_{n+s} \sum_{k=0}^{s-1} b_k^{(n)} F(x_{n+k}) + h_{n+s}^2 \sum_{k=0}^{s-1} c_k^{(n)} F'(x_{n+k}) + \dots \tag{6}$$

The following theorem provides the maximal convergence rate for any method of the form (6). Furthermore, it provides a fundamental barrier on the convergence rate of LMM-based root-finders.

Theorem 1. *For simple roots, the maximal convergence rate p for any method of the form* (6)*, where the coefficients are chosen so as to give the highest order of convergence, is given by the largest real root of*

$$p^s = \sum_{k=0}^{s-1} p^k (d + \sigma_k), \tag{7}$$

where d is the number of derivatives of f^{-1} used in the method. Thus, $d = 1$ for methods defined by (4)*. The coefficients σ_k indicate whether the coefficients $a_k^{(n)}$ are arbitrarily fixed from the outset or left free to maximise the order of convergence, i.e., $\sigma_k = 1$ if $a_k^{(n)}$ is free and $\sigma_k = 0$ otherwise. Moreover, the limiting convergence rate for any method using d derivatives is $d + 2$.*

Proof. 1. Any method of the form (6) implicitly uses inverse polynomial (Hermite) interpolation applied to the inverse function f^{-1}, let us call the resulting interpolation H. Let y_{n+k}, $k = 0, \ldots, s-1$ be the interpolation points. At each point y_{n+k}, there are $d + \sigma_k$ values are interpolated, the inverse function value x_k if $\sigma_k = 1$ and d derivative values. Thus, the polynomial interpolation error formula gives

$$f^{-1}(y) - H(y) = \frac{[f^{-1}]^{(N+1)}(\upsilon)}{(N+1)!} \prod_{k=0}^{s-1} (y - y_{n+k})^{d+\sigma_k},$$

where υ is in the interval spanned by the interpolation points and $N = sd + \sum_{k=0}^{s-1} \sigma_k$. The approximation to the root is then computed as $x_{n+s} = H(0)$. Let us denote the exact value of the root as α, then

$$|x_{n+s} - \alpha| = \frac{|[f^{-1}]^{(N+1)}(\upsilon)|}{(N+1)!} \prod_{k=0}^{s-1} |y_{n+k}|^{d+\sigma_k}.$$

Define $\varepsilon_{n+k} = x_{n+k} - \alpha$ and recognise that $f(x_{n+k}) = f(\alpha + \varepsilon_{n+k}) = f'(\alpha)\varepsilon_{n+k} + \mathcal{O}(\varepsilon_{n+k}^2)$, where $f'(\alpha) \neq 0$. Thus, we find

$$|\varepsilon_{n+s}| \approx A_0 |\varepsilon_{n+s-1}|^{d+\sigma_{s-1}} \cdots |\varepsilon_n|^{d+\sigma_0},$$

where $A_0 > 0$ is a constant depending on $[f^{-1}]^{(N+1)}(\upsilon)$, s and $f'(\alpha)$. The error behaviour is of the form

$$|\varepsilon_{l+1}| = C|\varepsilon_l|^p, \tag{$*$}$$

asymptotically as $p \to \infty$. Here, $C > 0$ is an arbitrary constant. Applying $(*)$ s times on the left and $s-1$ times on the right-hand side leads to

$$|\varepsilon_n|^{p^s} \approx A_1 |\varepsilon_n|^{\sum_{k=0}^{s-1} p^k (d+\sigma_k)},$$

where all the constants have been absorbed into A_1. Thus, (7) is established.

2. Finally, the highest convergence rate can be achieved by leaving all $a_k^{(n)}$ free. This way, we obtain

$$p^s = (d+1)\sum_{k=0}^{s-1} p^k = (d+1)\frac{p^s-1}{p-1}.$$

Simplifying, we obtain

$$p^{s+1} - (d+2)p^s + d + 1 = 0.$$

Note that $p = 1$ is always a solution of this equation. However, the maximal convergence rate is given by the largest real root, so that we look for solutions $p > 1$. Dividing by p^s yields

$$p - (d+2) + \frac{d+1}{p^s} = 0,$$

hence if $s \to \infty$, we obtain $p = d + 2$. □

From Theorem 1, we find several special cases, such as the derivative-free interpolation root-finders. Using $d = 0$, we find the following result.

Corollary 1. *Inverse polynomial interpolation root-finders, i.e., $d = 0$ resulting in all $b_k^{(n)} = 0$ in (4), can attain at most a convergence rate that is quadratic. Their convergence rates are given by the largest real root of*

$$p^{s+1} - 2p^s + 1 = 0. \tag{8}$$

Proof. The coefficients $\{a_k^{(n)}\}_{k=0}^{s-1}$ are chosen to maximise the order of convergence, so that $\sigma_k = 1$ for all $k = 0, \ldots, s-1$, while $d = 0$, leading to

$$p^s = \sum_{k=0}^{s-1} p^k = \frac{p^s-1}{p-1}.$$

Simplifying yields (8). Furthermore, the convergence rate is bounded by $d+2 = 2$, since $d = 0$. □

Inverse polynomial root-finders such as the secant method ($s = 2$) or inverse quadratic interpolation ($s = 3$) are derivative-free, so that their highest convergence rate is 2 according to Theorem 1. The first few convergence rates for derivative-free inverse polynomial interpolation methods are presented in Table 1. The well-known convergence rates for the secant method and the inverse quadratic interpolation method are indeed reproduced. As becomes clear from the table, the rates quickly approach 2 but never quite get there. The increase in convergence rate becomes smaller and smaller as we increase the number of interpolation points.

Next, we cover the Adams-Bashforth methods also discussed in [5]. As ODE solvers, Adams-Bashforth methods are explicit integration methods that have

Table 1: The first few convergence rates for s points using only function values.

s	p
2	1.62
3	1.84
4	1.92
5	1.97

order of accuracy s [9]. However, as Theorem 1 suggests, as root-finders they will have a convergence rate that is smaller than cubic, since $d = 1$. In fact, the convergence rate of Adams-Bashforth root-finders is bounded by $\frac{3+\sqrt{5}}{2} = 2.62$ as was proven by Grau-Sánchez et al. [5]. The following corollary is a generalisation of their result.

Corollary 2. *The Adams-Bashforth root-finder methods with $s \geq 2$ have maximal convergence rates given by the largest real root of*

$$p^{s+1} - 3p^s + p^{s-1} + 1 = 0. \tag{9}$$

Moreover, the convergence rate for any root-finder based on Adams-Bashforth methods is bounded by $\frac{3+\sqrt{5}}{2} \approx 2.62$.

Proof. Adams-Bashforth methods have $a_k^{(n)} = 0$ for $k = 0, \ldots, s-2$, resulting in $\sigma_k = 0$ for $k = 0, \ldots, s-2$ and $\sigma_{s-1} = 1$. Furthermore, the methods use a single derivative of f^{-1} so that $d = 1$. The $s = 1$ method is equal to Newton's method, which has a quadratic convergence rate. For $s \geq 2$, we find from (7) that

$$p^s = p^{s-1} + \sum_{k=0}^{s-1} p^k = p^{s-1} + \frac{p^s - 1}{p - 1}.$$

Simplifying yields (9). Again, we assume that $p > 1$ and we divide by p^{s-1}, so that

$$p^2 - 3p + 1 + \frac{1}{p^{s-1}} = 0,$$

where we again let $s \to \infty$, yielding

$$p^2 - 3p + 1 = 0,$$

which has as the largest real root $\frac{3+\sqrt{5}}{2} \approx 2.62$. $\square$

The first few convergence rates for the Adams-Bashforth root-finder methods are given in Table 2 and agree with the rates found by Grau-Sánchez et al. As becomes clear from the table, the convergence rates quickly draw near the bound of 2.62. Yet again we are met with steeply diminishing returns as we increase the number of history points s.

The Adams-Bashforth root-finder methods cannot attain a convergence rate higher than 2.62, which is still some way off the cubic bound given by Theorem 1.

Table 2: The first few convergence rates for Adams-Bashforth root-finder method using s points.

s	p
1	2
2	2.41
3	2.55
4	2.59
5	2.61

Using another linear multistep method may therefore result in convergence rates closer to cubic. For ODE solvers, trying to obtain a higher convergence rate by increasing the number of points often leads to instabilities. In fact, polynomial interpolation on equispaced points can even lead to diverging results, e.g., Runge's phenomenon [10]. However, root-finders generate a convergent set of interpolation points, therefore a higher convergence rate can be achieved by adding more history points.

Let us inspect the convergence rates of different LMM-based root-finders using Theorem 1, see Table 3. These convergence rates are computed under the assumption that all derivatives and point values are used, i.e., $\sigma_k = 1$ for $k = 0, \ldots, s-1$ in Theorem 1. The convergence rate of a d-derivative method can be boosted by at most 1, and the table shows that this mark is attained very quickly indeed. Adding a few history points raises the convergence rate significantly, but finding schemes with $s > 3$ is likely to be a waste of time.

Table 3: The first few convergence rates for s points (vertical) using function values and the first d derivatives (horizontal).

$s \backslash d$	1	2	3	4
1	2	3	4	5
2	2.73	3.79	4.82	5.85
3	2.91	3.95	4.97	5.98
4	2.97	3.99	4.99	5.996

Thus, provided that the root-finders are stable, a higher convergence rate can be achieved by adding history points, as well as their derivative information. However, we note that the stability issues in ODE solvers arises from the fact that polynomial interpolation is applied on an equispaced grid. Root-finders are designed to home in on a root, and when convergent, the step sizes will decrease rapidly. Just as Runge's phenomenon can be countered by switching to, e.g., Gauß nodes, polynomial interpolation is stable on the set of points generated by the root-finder itself, provided the starting guess is sufficiently close.

3. Full LMM-based root-finders

Let us investigate full LMM-based root-finders that use a single derivative, thus methods of the form (4). The current step size is then given by $h_{n+s} =$

$-f(x_{n+s-1})$. Let us define $q_k^{(n)}$ as

$$q_k^{(n)} = \frac{f(x_{n+k})}{f(x_{n+s-1})}, \quad k = 0, \ldots, s-2, \tag{10}$$

so that $h_{n+s} q_k^{(n)} = -f(x_{n+k})$ is the total step between y_{n+k} and $y_{n+s} = 0$. The Taylor expansions of $x(y_{n+k})$ and $x'(y_{n+k})$ about y_{n+s} are then given by

$$x(y_{n+k}) = x(y_{n+s}) + \sum_{m=1}^{\infty} \frac{1}{m!}(-h_{n+s}q_k)^m x^{(m)}(y_{n+s}), \tag{11a}$$

$$x'(y_{n+k}) = x'(y_{n+s}) + \sum_{m=1}^{\infty} \frac{1}{m!}(-h_{n+s}q_k)^m x^{(m+1)}(y_{n+s}), \tag{11b}$$

where we have dropped the superscript (n) for brevity. Substituting these into (4), we obtain

$$\begin{aligned} &x(y_{n+s})\left[1 + \sum_{k=0}^{s-1} a_k\right] - h_{n+s}x'(y_{n+s})\left[\sum_{k=0}^{s-1} a_k q_k + b_k\right] \\ &+ \sum_{m=2}^{\infty} \frac{1}{(m-1)!}(-h_{n+s})^m x^{(m)}(y_{n+s}) \sum_{k=0}^{s-1} \left[\tfrac{1}{m} q_k^m a_k + q_k^{m-1} b_k\right] = 0. \end{aligned} \tag{12}$$

The consistency conditions then are given by

$$\sum_{k=0}^{s-1} a_k = -1, \tag{13a}$$

$$\sum_{k=0}^{s-1} a_k q_k + b_k = 0. \tag{13b}$$

This gives us two equations for $2s$ coefficients, so that we can eliminate another $2s-2$ leading order terms, resulting in the conditions

$$\sum_{k=0}^{s-1} \frac{q_k^m}{m} a_k + q_k^{m-1} b_k = 0, \tag{14}$$

where $m = 2, \ldots, 2s-1$.

3.1. The $s = 2$ method

The $s = 2$ LMM-based method is given by

$$x_{n+2} + a_1 x_{n+1} + a_0 x_n = h_{n+2}\Big(b_1 F(x_{n+1}) + b_0 F(x_n)\Big), \tag{15}$$

where we have again suppressed the superscript (n) on the coefficients. Here, $h_{n+2} = -f(x_{n+1})$ so that we may write $q = q_0$, i.e.

$$q = \frac{f(x_n)}{f(x_{n+1})}. \tag{16}$$

Applying (13) and (14), we find a set of linear equations, i.e.,

$$a_1 + a_0 = -1, \tag{17a}$$
$$a_1 + qa_0 + b_1 + b_0 = 0, \tag{17b}$$
$$\tfrac{1}{2}a_1 + \tfrac{1}{2}q^2 a_0 + b_1 + qb_0 = 0, \tag{17c}$$
$$\tfrac{1}{3}a_1 + \tfrac{1}{3}q^3 a_0 + b_1 + q^2 b_0 = 0. \tag{17d}$$

These equations may be solved, provided $q \neq 1$, to yield

$$a_0 = \frac{1-3q}{(q-1)^3} \qquad a_1 = -1 - a_0, \tag{18a}$$
$$b_0 = \frac{q}{(q-1)^2} \qquad b_1 = qb_0. \tag{18b}$$

The condition $q \neq 1$ is equivalent to $f(x_{n+1}) \neq f(x_n)$. This condition is not very restrictive, as stronger conditions are needed to ensure convergence.

The above method may also be derived from the inverse polynomial interpolation perspective, using the ansatz

$$H(y) = h_3\big(y - f(x_{n+1})\big)^3 + h_2\big(y - f(x_{n+1})\big)^2 + h_1\big(y - f(x_{n+1})\big) + h_0, \tag{19}$$

where h_i, $i = 0, 1, 2, 3$ are undetermined coefficients. The coefficients are fixed by demanding that H interpolates f^{-1} and its derivative at $y = f(x_{n+1})$ and $y = f(x_n)$, i.e.,

$$H\big(f(x_n)\big) = x_n, \tag{20a}$$
$$H\big(f(x_{n+1})\big) = x_{n+1}, \tag{20b}$$
$$H'\big(f(x_n)\big) = \frac{1}{f'(x_n)}, \tag{20c}$$
$$H'\big(f(x_{n+1})\big) = \frac{1}{f'(x_{n+1})}. \tag{20d}$$

Solving for h_i, $i = 0, 1, 2, 3$ and setting $y = 0$, we find the same update x_{n+2} as (15).

The stability of the $s = 2$ LMM method depends on the coefficients of the LMM in much the same way as an ODE solver. Indeed, we can set the sequence $\tilde{x}_n = x_n + z_n$ where x_n is the sequence generated by exact arithmetic we wish to find while z_n is a parasitic mode. It can be shown that the parasitic mode satisfies

$$z_{n+2} + a_1 z_{n+1} + a_0 z_n = 0, \tag{21}$$

so that it may grow unbounded if the roots are greater than 1 in modulus. Using the ansatz $z_n = B\lambda^n$, we find the characteristic polynomial of the $s = 2$ method, i.e.,

$$\rho(\lambda) = \lambda^2 - \lambda\,(1 + a_0) + a_0 = (\lambda - 1)\,(\lambda - a_0)\,, \tag{22}$$

where the roots cans simply be read off. Stability of the root-finder is ensured if the stability polynomial of the method has a single root with $\lambda = 1$, while the other roots satisfy $|\lambda| < 1$. This property is called zero-stability for linear multistep ODE solvers. Thus, to suppress parasitic modes we need

$$|a_0| = \left| \frac{1-3q}{(q-1)^3} \right| < 1. \tag{23}$$

This reduces to q being either $q < 0$, or $q > 3$, so that $|q| > 3$ is a sufficient condition. Thus, if the sequence $\{|f(x_n)|\}_{n=1}^{\infty}$ is decreasing fast enough, any parasitic mode is suppressed. We may estimate q as a ratio of errors, since $f(x_n) = f'(\alpha)\varepsilon_n + \mathcal{O}(\varepsilon_n^2)$, so that

$$q \approx \frac{\varepsilon_n}{\varepsilon_{n+1}}. \tag{24}$$

Using $\varepsilon_{n+1} = C\varepsilon_n^p$ with $p = 1 + \sqrt{3}$, we find that

$$\varepsilon_n < C_1, \tag{25}$$

with $C_1 = \left(\frac{1}{3C}\right)^{\frac{1}{\sqrt{3}}}$. We conclude that the method will be stable if the initial errors are smaller than the constant C_1, which depends on the details of the function f in the vicinity of the root. This condition translates to having the starting values sufficiently close to the root. This is a rather typical stability condition for root-finders.

3.2. $s = 3$ method

We may again apply (13) - (14) to find a method with $s = 3$, this time we have 6 coefficients, given by

$$a_0 = \frac{q_1^2(q_0(3+3q_1-5q_0)-q_1)}{(q_0-1)^3(q_0-q_1)^3}, \qquad b_0 = \frac{q_0 q_1^2}{(q_0-1)^2(q_0-q_1)^2}, \tag{26a}$$

$$a_1 = \frac{q_0^2(q_1(5q_1-3q_0-3)+q_0)}{(q_1-1)^3(q_0-q_1)^3}, \qquad b_1 = \frac{q_0^2 q_1}{(q_0-q_1)^2(q_1-1)^2}, \tag{26b}$$

$$a_2 = \frac{q_0^2 q_1^2(3q_1-q_0(q_1-3)-5)}{(q_0-1)^3(q_1-1^3)}, \qquad b_2 = \frac{q_0^2 q_1^2}{(q_0-1)^2(q_1-1)^2}, \tag{26c}$$

where $q_0 = \frac{f(x_n)}{f(x_{n+2})}$ and $q_1 = \frac{f(x_{n+1})}{f(x_{n+2})}$. Here, we have the conditions $q_0 \neq 1$ and $q_1 \neq 1$, reducing to the condition that all y-values must be unique. Again, this condition is not very restrictive for reasons detailed above.

Methods with a greater number of history points are possible, however, the gain in convergence rate from $s = 3$ to $s = 4$ is rather slim, as indicated by Table 3. If such methods are desirable, they can be derived by selecting coefficients that satisfy (13) - (14).

4. Results

Like the secant method, the $s = 2$ full LMM root-finding method needs two starting points for the iteration. However, as the analytical derivative is available, we choose to simply start Newton's method with one point, say x_0, giving x_1. The $s = 3$ method needs three starting values, therefore the next value x_2 is obtained from the $s = 2$ LMM method. The LMM-based methods can be efficiently implemented by storing the function value and derivative value of the previous step, thus resulting in a need for only one function and one derivative evaluation per iteration.

The efficiency measure is defined as $p^{\frac{1}{w}}$ with p the order of convergence and w the number of evaluations per iteration [4]. Assuming the function itself and the derivative cost the same to evaluate, the $s = 3$ LMM-based method has an efficiency measure of $\sqrt{2.91} \approx 1.71$. Compared to Newton's method, with an efficiency measure of $\sqrt{2} \approx 1.41$, this is certainly an improvement. Even compared to more recent root-finders, such as the fourth-order method of Shengguo et al.[11] with $\sqrt[3]{4} \approx 1.59$, our method holds up.

4.1. Numerical examples

Here, we provide a number of test cases and show how many iterations LMM-based root-finders take versus the number needed by Newton's method, see Table 4. We have used a selection of different test cases with polynomials, exponentials, trigonometric functions, square roots and combinations thereof. For each of the test problems shown, the methods converged within a few iterations. Some problems were deliberately started near a maximum or minimum to see the behaviour when the derivatives are small.

The test computations were performed using the variable-precision arithmetic of MATLAB's Symbolic Math Toolbox. The number of digits was set to 300 while the convergence criterion used was

$$|x_{l+1} - x_l| \leq 10^{-\eta}, \tag{27}$$

with $\eta = 250$. The numerical convergence rates were computed with the error behaviour

$$|\varepsilon_{l+1}| = C|\varepsilon_l|^p, \tag{28}$$

asymptotically as $p \to \infty$. The limiting value of the estimates for p is displayed in Table 4.

Overall, Newton's method consistently displayed a quadratic convergence rate, which is the reason we did not display it in the table. The LMM-based methods, on the other hand, generally have a higher convergence rate that may vary somewhat from problem to problem. This is due to the fact that the step sizes may vary slightly while the convergence rates of the LMM-based methods only holds asymptotically, even with so many digits.

Table 4: Test cases with iterations taken for Newton's method (subscript N) and the LMM-based method with $s = 2$ (subscript 2) and $s = 3$ method (subscript 3).

function	root	x_0	$\#\text{its}_N$	$\#\text{its}_2$	$\#\text{its}_3$	p_2	p_3
$x + e^x$	-0.57	1.50	11	8	8	2.73	2.93
$\sqrt{x} - \cos(x)$	0.64	0.50	9	7	8	2.74	2.91
$e^x - x^2 + 3x - 2$	0.26	0.00	10	8	7	2.72	2.94
$x^4 - 3x^2 - 3$	1.95	1.30	17	14	14	2.73	2.92
$x^3 - x - 1$	1.32	1.00	12	9	9	2.73	2.64
$e^{-x} - x^3$	0.77	2.00	13	10	10	2.73	2.92
$5\big(\sin(x) + \cos(x)\big) - x$	2.06	1.50	11	9	9	2.73	2.92
$x - \cos(x)$	0.74	1.00	9	7	7	2.72	2.93
$\log(x-1) + \cos(x-1)$	1.40	1.60	12	9	9	2.73	2.92
$\sqrt{1+x} - x$	1.62	1.00	9	7	7	2.73	2.92
$\sqrt{e^x - x} - 2x$	0.54	1.00	11	8	7	2.73	2.92
Total number of iterations			124	96	95		

Table 5: Convergence history for $\tanh(x)$ of the three methods: Newton (subscript N) and the LMM-based methods with $s = 2$ and $s = 3$. Note that the root of $\tanh(x)$ is at $x = 0$.

x_N	$x_{(s=2)}$	$x_{(s=3)}$
1.239	1.239	1.239
-1.719	-1.719	-1.719
6.059	0.8045	0.8045
$-4.583 \cdot 10^4$	0.7925	-0.6806
Inf	-0.7386	1.377
	$-6.783 \cdot 10^{-3}$	-0.7730
	$9.323 \cdot 10^{-6}$	$3.466 \cdot 10^{-2}$
	$\lvert x_{(s=2)} \rvert < \epsilon$	$-3.032 \cdot 10^{-4}$
		$1.831 \cdot 10^{-11}$
		$\lvert x_{(s=3)} \rvert < \epsilon$

4.2. Pathological functions

A classical example of a pathological function for Newton's method is the hyperbolic tangent $\tanh(x)$. Students often believe that Newton's method converges for any monotone function until they are asked to find the root of $\tanh(x)$. Hence, we have used this function as a test case using standard double precision floating point arithmetic and a convergence criterion reading

$$|x_{l+1} - x_l| \leq 2\epsilon, \tag{29}$$

with ϵ the machine precision. Newton's method fails to converge for starting values with approximately $|x_0| \geq 1.089$, see Table 5. Our $s = 2$ LMM-based method extends this range somewhat more and converges for any starting value with roughly $|x_0| \leq 1.239$. The behaviour of the $s = 3$ LMM-based method is similar, though it does take two more iterations to converge.

Starting at $x_0 = 1.239$, Newton's method diverges quickly, returned as $-\inf$ after only 4 iterations. The LMM-based method, on the other hand, bounces around positive and negative values for about 5 iterations until it is close enough to the root. After that, the asymptotic convergence rate sets in and the root is quickly found, reaching the root within machine precision at 7 iterations.

Donovan et al.[12] developed another test function for which Newton's method fails, it gives a false convergence result to be precise. The test function is given by

$$h(x) = \sqrt[3]{x}e^{-x^2}, \tag{30}$$

which is, in fact, infinitely steep near the root $x = 0$, yet smooth, see Figure 1. Again, we used double precision arithmetic and (29) as a stopping criterion. Newton's method diverges for any starting value except the exact root itself. However, Newton's method eventually gives a false convergence result as the increment $|x_{l+1} - x_l|$ falls below the tolerance. The $s = 2$ and $s = 3$ LMM-based methods converge when starting with $|x_0| \leq 0.1147$ for this problem, see Table 6.

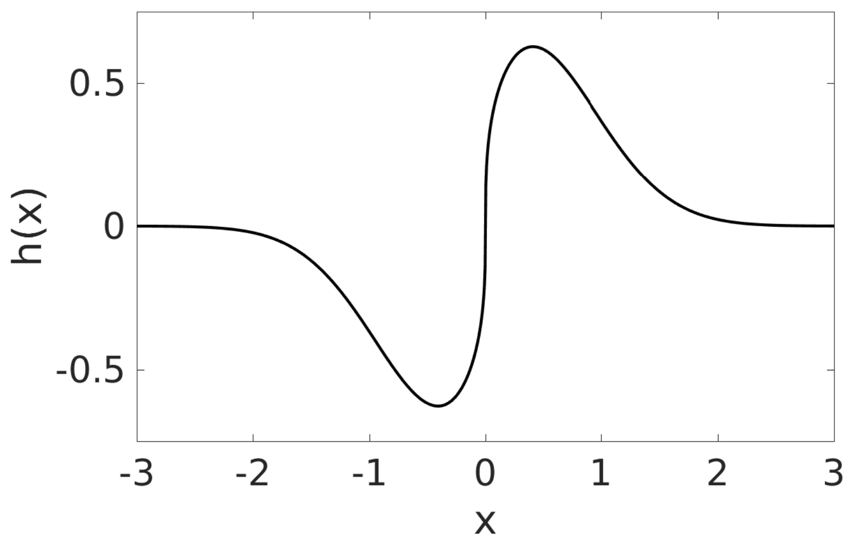

Figure 1: Pathological test function $h(x)$ from (30).

Starting at the maximal $x_0 = 0.1147$ for instance, the LMM-based methods bounce several times between positive and negative x-values without making much headway. After that, the root is close enough and the asymptotic convergence rate sets in, reaching the root to within machine precision in a few steps.

We believe the reason that the LMM-based method has increased stability is due to the fact that it uses two points to evaluate the function and its derivative.

Table 6: Convergence history for $h(x)$ from (30) of the three methods: Newton (subscript N) and the LMM-based methods with $s = 2$ and $s = 3$. Note that the root is at $x = 0$.

x_N	$x_{(s=2)}$	$x_{(s=3)}$
0.1147	0.1147	0.1147
0.2589	-0.2589	-0.2589
1.0402	0.1016	0.1016
1.6084	$9.993 \cdot 10^{-2}$	$-5.648 \cdot 10^{-2}$
1.9407	-0.2581	0.1959
2.2102	$9.840 \cdot 10^{-2}$	-0.1611
2.4445	$9.810 \cdot 10^{-2}$	$5.021 \cdot 10^{-2}$
2.6549	-0.2344	$-7.190 \cdot 10^{-2}$
2.8478	$6.602 \cdot 10^{-2}$	$4.947 \cdot 10^{-2}$
3.0270	$6.021 \cdot 10^{-2}$	$-3.777 \cdot 10^{-3}$
3.1953	$-4.939 \cdot 10^{-2}$	$3.027 \cdot 10^{-4}$
3.3543	$-4.019 \cdot 10^{-4}$	$-6.875 \cdot 10^{-6}$
3.5056	$1.288 \cdot 10^{-4}$	$1.216 \cdot 10^{-9}$
3.6502	$2.028 \cdot 10^{-10}$	$-4.652 \cdot 10^{-15}$
3.7889	$-5.308 \cdot 10^{-15}$	$\lvert x_{(s=2)} \rvert < \epsilon$
3.9225	$\lvert x_{(s=2)} \rvert < \epsilon$	

In both cases, the iterations jump between positive and negative values, enclosing the root. In this fashion, the LMM-based method acts much like the *regula falsi* method. Once the iterates are close enough to the root, the asymptotic convergence rate sets in and the iterates converge in but a few steps.

5. A robust implementation

As with most open root-finding algorithms, the conditions under which the method is guaranteed to converge are rather restrictive. Therefore, we have designed a bracketing version of the LMM-based method that is guaranteed to converge. The algorithm is based on Brent's method, using similar conditions to catch either slow convergence or runaway divergence. This version of the LMM-based method does, however, require an enclosing bracket $[a, b]$ on which the function changes sign, i.e., $f(a)f(b) < 0$. Alternatively, such a method can start out as an open method, switching to the bracketing method when a sign change is detected.

The algorithm consists of a cascade of methods increasing in accuracy but decreasing in robustness, similar to Brent's method. At the lowest level stands the most robust method, bisection, guarding against steps outside the current search bracket. Bisection is guaranteed to converge, but does so rather slowly. On the highest level we use the full $s = 3$ LMM-based method discussed in the previous section. Thus, in the best possible case, the method will converge with a rate of 2.91. The method is, by virtue of the bisection method, guaranteed to converge to a root.

Like Brent's method and Dekker's method, the LMM-based method keeps track of three points a, b and c. Here, b is the best estimate of the root so far, c is the previous value for b while a is the contrapoint so that $\mathrm{int}[a,b]$ encloses the root. Ideally, all three values are used to compute the next value for b. However, extra conditions are added to ensure the inverse actually makes sense on the interval $\mathrm{int}[a,c]$.

Consider the case where the sign of $f'(c)$ is not equal to the sign of $\frac{f(b)-f(a)}{b-a}$, but the sign of $f'(b)$ is. It follows that there is an extremum between b and c, and the inverse function does not exist in the interval $\mathrm{int}[a,c]$, leading to an error if we were to compute the inverse interpolation.

Thus, the following condition should be applied to each derivative value: the sign of the derivative needs to be the same as the sign of the secant slope on $\mathrm{int}[a,b]$, i.e.,

$$\mathrm{sgn}\,(f'(z)) = \mathrm{sgn}\left(\frac{f(b)-f(a)}{b-a}\right), \tag{31}$$

where $z = a, b, c$. Only when the derivative at a point satisfies (31) can the derivative sensibly contribute to the inverse. Otherwise the derivative information should be discarded, leading to lower-order interpolation. If all derivatives are discarded, the resulting interpolation is inverse quadratic or the secant method.

Ultimately, the method provides an interval on which the function f changes sign with a relative size of some given tolerance δ, i.e.,

$$|a-b| \leq \delta|b|. \tag{32}$$

We shall use $\delta = 2\epsilon$ in all our examples, with ϵ the machine precision. As an input, the algorithm has f, f', a and b such that $f(a)f(b) < 0$. The algorithm can be described in the following way:

1. If all three function values are different, use $s = 3$, otherwise use $s = 2$.
2. Check the sign of the derivatives at points a, b and c, include the derivatives that have the proper sign.
3. If the interpolation step is worse than a bisection step, or outside the interval $[a,b]$, use bisection.
4. If the step is smaller than the tolerance, use the tolerance as step size.
5. If the convergence criterion is met, exit, otherwise go to 1.

The first step determines the number of history points that can be used. The second step determines which derivative values should be taken into account. In effect, only the second step is essentially different from Brent's method, with all the following steps exactly the same [8]. The extra conditions on the derivatives gives rise to a selection of 12 possible root-finders, including inverse quadratic interpolation and the secant method. Our method can therefore be seen as building another 10 options on top of Brent's method. Naturally, sufficiently close to the root, the derivative conditions will be satisfied at all three points and the method will use the full LMM method with $s = 3$.

5.1. Comparison with Brent's method

Here we give a few examples of the robust LMM-based root-finding algorithm discussed above compared to Brent's method. As a performance measure, we use the number of iterations. Standard double precision arithmetic is employed, as that provides sufficient material for comparison. For both methods, the stopping criterion is given by (32), the relative size of the interval must be sufficiently small.

Table 7: Test cases with iterations taken for Brent's method and the LMM-based method. Subscript B represents Brent while subscript LMM represents the LMM-based method.

function	root	$[a, b]$	$\#\text{its}_B$	$\#\text{its}_{LMM}$
$x + e^x$	-0.57	$[-1, 1]$	6	4
$\sqrt{x} - \cos(x)$	0.64	$[0, 2]$	8	4
$e^x - x^2 + 3x - 2$	0.26	$[-1, 1]$	5	3
$x^4 - 3x^2 - 3$	1.95	$[1, 3]$	10	8
$x^3 - x - 1$	1.32	$[0, 2]$	29	6
$e^{-x} - x^3$	0.77	$[0, 2]$	9	4
$5\big(\sin(x) + \cos(x)\big) - x$	2.06	$[0, 4]$	45	6
$x - \cos(x)$	0.74	$[0, 1]$	7	3
$\log(x - 1) + \cos(x - 1)$	1.39	$[1.2, 1.6]$	31	4
$\sqrt{1 + x} - x$	1.62	$[0, 2]$	5	3
$\sqrt{e^x - x} - 2x$	0.54	$[-1, 2]$	9	4
Total number of iterations			164	49
Total number of function evaluations			164	98

Table 7 shows that for most functions, both Brent's method and the LMM-based method take a comparable number of iterations. However, in some cases, the difference is considerable. In the worst case considered here, Brent's method takes 7.5 times as many iterations to converge. In terms of efficiency index, Brent's method should be superior with an efficiency index of 1.84 against 1.71 of the LMM-based method. Taken over the whole set of test functions, however, Brent's method takes more than three times as many iterations in total, leading to a significant increase in function evaluations. We conclude therefore that practically, the LMM-based root-finder is a better choice.

6. Conclusions

We have discussed root-finders based on full linear multistep methods. Such LMM-based methods may be interpreted as inverse polynomial (Hermite) interpolation methods, resulting in a simple and general convergence analysis. Furthermore, we have proven a fundamental barrier for LMM-based root-finders: their convergence rate cannot exceed $d+2$, where d is the number of derivatives used in the method.

The results indicate that compared to the Adams-Bashforth root-finder methods of Grau-Sánchez et al. [5], any full LMM-based method with $s \geq 2$

has a higher convergence rate. As ODE solvers, full LMMs are typically not zero-stable and special choices of the coefficients have to be made. Employed as root-finders on the other hand, it turns out that LMMs are stable, due to the rapid decrease of the step size. This allows the usage of full LMMs that are otherwise not zero-stable.

Contrary to the Adams-type methods, the full LMM-based root-finders can achieve the convergence rate of 3 in the limit that all history points are used. The $s = 2$ and $s = 3$ methods, s being the number of history points, were explicitly constructed and provide a convergence rate of 2.73 and 2.91, respectively. Numerical experiments confirm these predicted convergence rates. Furthermore, application to pathological functions where Newton's method diverges show that the LMM-based methods also have enhanced stability properties.

Finally, we have implemented a robust LMM-based method that is guaranteed to converge when provided with an enclosing bracket of the root. The resulting robust LMM root-finder algorithm is a cascade of twelve root-finders increasing in convergence rate but decreasing in reliability. At the base sits bisection, so that the method is indeed guaranteed to converge to the root. At the top resides the $s = 3$ LMM-based root-finder, providing a maximal convergence rate of 2.91.

In terms of efficiency index, Brent's method is theoretically the preferred choice with 1.84 compared to 1.71 for the LMM-based method. However, numerical examples show that the increased convergence rate leads to a significant decrease in the total number of function evaluations over a range of test functions. Therefore, in practical situations, provided the derivative is available, the LMM-based method performs better.

Acknowledgements

This work was generously supported by Philips Lighting and the Intelligent Lighting Institute.

[1] A. S. Glassner, *An Introduction to Ray Tracing.* London: Academic Press, 1991.

[2] J. Chaves, *Introduction to Non-Imaging Optics.* CRC Press, 2008.

[3] J. C. Butcher, *The Numerical Analysis of Ordinary Differential Equations.* Wiley, 1987.

[4] W. Gautschi, *Numerical Analysis.* Birkhäuser, Boston, 2012.

[5] M. Grau-Sánchez, M. Noguera, and J. L. Díaz-Barrero, "Adams-like techniques for zero-finder methods," *Applied Mathematics and Computation*, vol. 211, pp. 130–136, 2009.

[6] M. Grau-Sánchez and J. L. Díaz-Barrero, "Zero-finder methods derived using Runge-Kutta techniques," *Applied Mathematics and Computation*, vol. 217, pp. 5366–5376, 2011.

[7] M. Grau-Sánchez and J. M. Gutirrez, "Zero-finder methods derived from Obreshkovs techniques," *Applied Mathematics and Computation*, vol. 215, no. 8, pp. 2992 – 3001, 2009.

[8] R. P. Brent, *Algorithms for minimization without derivatives.* Prentice-Hall, 1973.

[9] E. Hairer, G. Wanner, and S. P. Nørsett, *Solving Ordinary Differential Equations I: nonstiff problems.* Springer-Verlag, 1987.

[10] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics.* Springer, 2007.

[11] L. Shengguo, L. Xiangke, and C. Lizhi, "A new fourth-order iterative method for finding multiple roots of nonlinear equations," *Applied Mathematics and Computation*, vol. 215, no. 3, pp. 1288 – 1292, 2009.

[12] G. C. Donovan, A. R. Miller, and T. J. Moreland, "Pathological functions for Newton's method," *The American Mathematical Monthly*, vol. 100, no. 1, pp. 53–58, 1993.

PREVIOUS PUBLICATIONS IN THIS SERIES:

Number	Author(s)	Title	Month
17-01	N. Kumar J.H.M. ten Thije Boonkkamp B. Koren A. Linke	A Nonlinear Flux Approximation Scheme for the Viscous Burgers Equation	January '17
17-02	T.G.J. Beelen	Finding all convex tangrams	January '17
17-03	A.J. Vromans A.A.F. van de Ven A. Muntean	Existence of weak solutions for a pseudo-parabolic system coupling chemical reactions, diffusion and momentum equations	February '17
17-04	S. Vovchak T.G.J. Beelen	Interference in wireless networks A game theory approach	February '17
17-05	B.S. van Lith J.H.M. ten Thije Boonkkamp W.L. IJzerman	Full linear multistep methods as root-finders	February '17

Ontwerp: de Tantes,
Tobias Baanders, CWI