

*Communications in  
Applied  
Mathematics and  
Computational  
Science*

vol. 9 no. 2 2014

# Communications in Applied Mathematics and Computational Science

msp.org/camcos

## EDITORS

### MANAGING EDITOR

John B. Bell  
Lawrence Berkeley National Laboratory, USA  
jbbell@lbl.gov

### BOARD OF EDITORS

Marsha Berger	New York University berger@cs.nyu.edu	Ahmed Ghoniem	Massachusetts Inst. of Technology, USA ghoniem@mit.edu
Alexandre Chorin	University of California, Berkeley, USA chorin@math.berkeley.edu	Raz Kupferman	The Hebrew University, Israel raz@math.huji.ac.il
Phil Colella	Lawrence Berkeley Nat. Lab., USA pcolella@lbl.gov	Randall J. LeVeque	University of Washington, USA rjl@amath.washington.edu
Peter Constantin	University of Chicago, USA const@cs.uchicago.edu	Mitchell Luskin	University of Minnesota, USA luskin@umn.edu
Maksymilian Dryja	Warsaw University, Poland maksymilian.dryja@acn.waw.pl	Yvon Maday	Université Pierre et Marie Curie, France maday@ann.jussieu.fr
M. Gregory Forest	University of North Carolina, USA forest@amath.unc.edu	James Sethian	University of California, Berkeley, USA sethian@math.berkeley.edu
Leslie Greengard	New York University, USA greengard@cims.nyu.edu	Juan Luis Vázquez	Universidad Autónoma de Madrid, Spain juanluis.vazquez@uam.es
Rupert Klein	Freie Universität Berlin, Germany rupert.klein@pik-potsdam.de	Alfio Quarteroni	Ecole Polytech. Féd. Lausanne, Switzerland alfio.quarteroni@epfl.ch
Nigel Goldenfeld	University of Illinois, USA nigel@uiuc.edu	Eitan Tadmor	University of Maryland, USA etadmor@cscamm.umd.edu
		Denis Talay	INRIA, France denis.talay@inria.fr

## PRODUCTION

production@msp.org

Silvio Levy, Scientific Editor

---

See inside back cover or [msp.org/camcos](http://msp.org/camcos) for submission instructions.

---

The subscription price for 2014 is US \$75/year for the electronic version, and \$105/year (+\$15, if shipping outside the US) for print and electronic. Subscriptions, requests for back issues from the last three years and changes of subscribers address should be sent to MSP.


---

Communications in Applied Mathematics and Computational Science (ISSN 2157-5452 electronic, 1559-3940 printed) at Mathematical Sciences Publishers, 798 Evans Hall #3840, c/o University of California, Berkeley, CA 94720-3840, is published continuously online. Periodical rate postage paid at Berkeley, CA 94704, and additional mailing offices.

---

CAMCoS peer review and production are managed by EditFLOW® from MSP.

PUBLISHED BY

 **mathematical sciences publishers**  
nonprofit scientific publishing

<http://msp.org/>

© 2014 Mathematical Sciences Publishers

## A COMPARISON OF HIGH-ORDER EXPLICIT RUNGE–KUTTA, EXTRAPOLATION, AND DEFERRED CORRECTION METHODS IN SERIAL AND PARALLEL

DAVID I. KETCHESON AND UMAIR BIN WAHEED

We compare the three main types of high-order one-step initial value solvers: extrapolation, spectral deferred correction, and embedded Runge–Kutta pairs. We consider orders four through twelve, including both serial and parallel implementations. We cast extrapolation and deferred correction methods as fixed-order Runge–Kutta methods, providing a natural framework for the comparison. The stability and accuracy properties of the methods are analyzed by theoretical measures, and these are compared with the results of numerical tests. In serial, the eighth-order pair of Prince and Dormand (DOP8) is most efficient. But other high-order methods can be more efficient than DOP8 when implemented in parallel. This is demonstrated by comparing a parallelized version of the well-known ODEX code with the (serial) DOP853 code. For an  $N$ -body problem with  $N = 400$ , the experimental extrapolation code is as fast as the tuned Runge–Kutta pair at loose tolerances, and is up to two times as fast at tight tolerances.

### 1. Introduction

The construction of very high-order integrators for initial value ordinary differential equations (ODEs) is challenging: very high-order Runge–Kutta (RK) methods are subject to vast numbers of order conditions, while very high-order linear multistep methods tend to have poor stability properties. Both extrapolation [9; 16] and deferred correction [7; 10] can be used to construct initial value ODE integrators of arbitrarily high order in a straightforward way. Both are usually viewed as iterative methods, since they build up a high-order solution based on lower order approximations. However, when the order is fixed, methods in both classes can be viewed as Runge–Kutta methods with a number of stages that grows quadratically with the desired order of accuracy.

It is natural to ask how these methods compare with standard Runge–Kutta

---

*MSC2010*: primary 65L06; secondary 65Y05.

*Keywords*: Runge–Kutta methods, extrapolation, deferred correction, ordinary differential equations, high-order methods, parallel.

methods. Previous studies have compared the relative (serial) efficiency of explicit extrapolation and Runge–Kutta (RK) methods [18; 32; 17], finding that extrapolation methods have no advantage over moderate to high-order Runge–Kutta methods, and may well be inferior to them [32; 17]. Consequently, extrapolation has received little attention in the last two decades. It has long been recognized that extrapolation methods offer excellent opportunities for parallel implementation [9]. Nevertheless, to our knowledge no parallel implementation has appeared, and comparisons of extrapolation methods have not taken parallel computation into account, even from a theoretical perspective. It seems that no work has thoroughly compared the efficiency of explicit spectral deferred correction methods with that of their extrapolation and RK counterparts. See [37] for a comparison of semi-implicit deferred correction and additive RK methods).

In this paper we compare the efficiency of explicit Runge–Kutta, extrapolation, and spectral deferred correction (DC) methods based on their accuracy and stability properties. The methods we study are introduced in Section 2 and range in order from four to twelve. In Section 4 we give a theoretical analysis based on metrics that are independent of implementation details. This section is similar in spirit and in methodology to the work of Hosea and Shampine [17]. In Section 5 we validate the theoretical predictions using simple numerical tests. These tests indicate, in agreement with our theoretical analysis and with previous studies, that extrapolation methods do not have a significant advantage over high-order Runge–Kutta methods, and may in fact be significantly less efficient. Spectral deferred correction methods based on explicit Euler generally fare even worse than extrapolation.

In Section 3 we analyze the potential of parallel implementations of extrapolation and deferred correction methods. We only consider parallelism “across the method”. Other approaches to parallelism in time often use parallelism “across the steps”; for instance, the parareal algorithm. Some hybrid approaches include PFASST [28; 11] and RIDC [5]; see also [14]. Our results should not be used to infer anything about those methods, since we focus on a simpler approach that does not involve parallelism across multiple steps.

For both extrapolation and (appropriately chosen) deferred correction methods, the number of stages that must be computed sequentially grows only linearly with the desired order of accuracy. Based on simple algorithmic analysis, we extend our theoretical analysis to parallel implementations of extrapolation and deferred correction. This analysis suggests that extrapolation should be more efficient than traditional RK methods, at least for computationally intensive problems. We investigate this further in Section 6 by performing a simple OpenMP parallelization of the ODEX extrapolation code. The observed computational speedup is very near the theoretical estimates, and the code outperforms the DOP853 (serial) code on some test problems.

No study of numerical methods can claim to yield conclusions that are valid for all possible problems. Our intent is to give some broadly useful comparisons and draw general conclusions that can serve as a guide to further studies. The analysis presented here was performed using the NodePy (Numerical ODEs in Python) package, which is freely available from [github.com/ketch/nodepy](https://github.com/ketch/nodepy). Additional code for reproducing the experiments in this work can be found online at [github.com/ketch/high\\_order\\_RK\\_RR](https://github.com/ketch/high_order_RK_RR).

## 2. High-order one-step embedded pairs

*... for high order RK formulas the construction of an embedding RK formula may be beyond human possibilities...*

P. Deuffhard, 1985

We are concerned with one-step methods for the solution of the initial value ODE

$$y'(t) = f(y), \quad y(t_0) = y_0, \tag{1}$$

where  $y \in \mathbb{R}^m$ ,  $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ . For simplicity of notation, we assume the problem has been written in autonomous form. An explicit Runge–Kutta pair computes approximations  $y_n, \hat{y}_n \approx y(t_n)$  as follows:

$$Y_i = y_n + h \sum_{j=1}^{i-1} a_{ij} f(Y_j), \quad 1 \leq j \leq s, \tag{2}$$

$$y_{n+1} = y_n + h \sum_{j=1}^s b_j f(Y_j), \tag{3}$$

$$\hat{y}_{n+1} = y_n + h \sum_{j=1}^s \hat{b}_j f(Y_j). \tag{4}$$

Here  $h$  is the step size,  $s$  denotes number of stages, the *stages*  $Y_i$  are intermediate approximations, and one evaluation of  $f$  is required for each stage. The coefficients  $A, b, \hat{b}$  determine the accuracy and stability of the method. The coefficients are typically chosen so that  $y_{n+1}$  has local error  $\tau = \mathcal{O}(h^p)$ , and  $\hat{y}_{n+1}$  has local error  $\hat{\tau} = \mathcal{O}(h^{\hat{p}})$  for some  $1 < \hat{p} < p$ . Here  $p$  is referred to as the order of the method, and sometimes such a method is referred to as a  $p(\hat{p})$  pair. The value  $\|y_{n+1} - \hat{y}_{n+1}\|$  is used to estimate the error and determine an appropriate size for the next step.

The theory of Runge–Kutta order conditions gives necessary and sufficient conditions for a Runge–Kutta method to be consistent to a given order [16; 3]. For order  $p$ , these conditions involve polynomials of degree up to  $p$  in the coefficients  $A, b$ . The number of order conditions increases dramatically with  $p$ : only eight conditions are required for order four, but order ten requires 1,205 conditions and

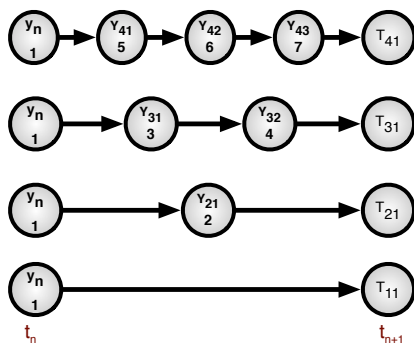
order fourteen requires 53,263 conditions. Although the order conditions possess a great deal of structure and certain simplifying assumptions can be used to facilitate their solution, the design of efficient Runge–Kutta pairs of higher than eighth order by direct solution of the order conditions remains a challenging area. Some methods of order as high as 14 have been constructed [12].

**2.1. Extrapolation.** Extrapolation methods provide a straightforward approach to the construction of high-order one-step methods; they can be viewed as Runge–Kutta methods, which is the approach taken here. For the mathematical foundations of extrapolation methods we refer the reader to [16, Section II.9]. The algorithmic structure of extrapolation methods has been considered in detail in previous works, including [36; 31]; we review the main results here. Various sequences of step numbers have been proposed, but we consider the harmonic sequence as it is usually the most efficient [8; 17]. We do not consider the use of smoothing, as previous studies have shown that it reduces efficiency [17].

**2.1.1. Euler extrapolation (Ex-Euler).** Extrapolation is most easily understood by considering the explicit Euler method

$$y_{n+1} = y_n + hf(y_n) \quad (5)$$

as a building block. The order  $p$  Ex-Euler algorithm computes  $p$  approximations to  $y(t_{n+1})$  by using the explicit Euler method, first breaking the interval into one step, then two steps, and so forth. The approximations to  $y(t_{n+1})$  computed in this manner are all first order accurate and are labeled  $T_{11}, T_{21}, \dots, T_{p1}$ . These values are combined using the Aitken–Neville interpolation algorithm to obtain a higher order approximation to  $y(t_{n+1})$ . The algorithm is depicted in Figure 1. For error estimation, we use the approximation  $T_{p-1,p-1}$  whose accuracy is one order less.



**Figure 1.** Structure of an Euler extrapolation step using the harmonic sequence 1, 2, 3, 4. Each numbered circle represents a function evaluation, and the numbers indicate the order in which they are performed.

---

```

for  $k = 1 \rightarrow p$  do                                     ▷ Compute first order approximations
     $Y_{k0} = y_n$ 
    for  $j = 1 \rightarrow k$  do
         $Y_{kj} = Y_{k,j-1} + \frac{h}{k} f(Y_{k,j-1})$ 
    end for
     $T_{k1} = Y_{kk}$ 
end for
for  $k = 2 \rightarrow p$  do                                     ▷ Extrapolate to get higher order
    for  $j = k \rightarrow p$  do
         $T_{jk} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{j/(j-k+1) - 1}$            ▷ Aitken–Neville formula for
    end for                                               extrapolation to order  $k$ 
end for
 $y_{n+1} = T_{pp}$                                            ▷ New solution value
 $\hat{y}_{n+1} = T_{p-1,p-1}$                                    ▷ Embedded method solution value

```

---

**Algorithm 1.** Explicit Euler extrapolation (Ex-Euler).

Simply counting the number of evaluations of  $f$  in Algorithm 1 shows that this is an  $s$ -stage Runge–Kutta method, where

$$s = \frac{p^2 - p + 2}{2}. \tag{6}$$

The quadratic growth of  $s$  as the order  $p$  is increased leads to relative inefficiency of very high-order extrapolation methods when compared to directly constructed Runge–Kutta methods, as we will see in later sections.

**2.1.2. Midpoint extrapolation (Ex-Midpoint).** It is common to perform extrapolation based on an integration method whose error function contains only even terms, such as the midpoint method [16; 36]. In this case, each extrapolation step raises the order of accuracy by two. We refer to this approach as Ex-Midpoint and give the algorithm below. Using midpoint extrapolation to obtain order  $p$  requires about half as many stages, compared to Ex-Euler:

$$s = \frac{p^2 + 4}{4}. \tag{7}$$

Again, the number of stages grows quadratically with the order.

**2.2. Deferred correction (DC-Euler).** Like extrapolation, deferred correction has a long history; its application to initial value problems goes back to [7]. Recently it has been revived as an area of research; see [10; 15] and subsequent works. Here we focus on the class of methods introduced in [10], with a modification introduced

---

```

r = p/2
for k = 1 → r do                                ▷ Compute second-order approximations
  Yk0 = yn
  Yk1 = Yk,0 +  $\frac{h}{2k} f(Y_{k,0})$                     ▷ Initial Euler step
  for j = 2 → 2k do
    Ykj = Yk,j-2 +  $\frac{h}{k} f(Y_{k,j-1})$                 ▷ Midpoint steps
  end for
  Tk1 = Yk,2k
end for
for k = 2 → r do                                ▷ Extrapolate to get higher order
  for j = k → r do
    Tjk = Tj,k-1 +  $\frac{T_{j,k-1} - T_{j-1,k-1}}{j^2/(j-k+1)^2 - 1}$     ▷ Aitken–Neville formula for
    extrapolation to order 2k
  end for
end for
yn+1 = Trr                                        ▷ New solution value
ŷn+1 = Tr-1,r-1                                  ▷ Embedded method solution value

```

---

**Algorithm 2.** Explicit midpoint extrapolation (Ex-Midpoint).

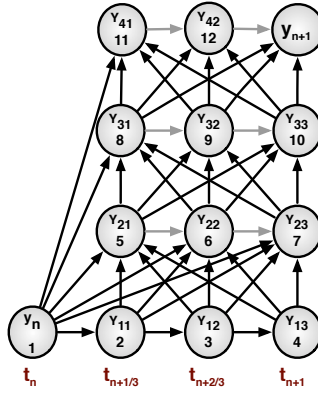
in [26]. These spectral DC methods are one-step methods and can be constructed for any order of accuracy.

Spectral DC methods start like extrapolation methods, by using a low-order method to step over subintervals of the time step; the subintervals can be equally sized, or Chebyshev nodes can be used. We consider methods based on the explicit Euler method and Chebyshev nodes. Subsequently, high-order polynomial interpolation of the computed values is used to approximate the integral of the error, or defect. Then the method steps over the same nodes again, applying a correction. This procedure is repeated until the desired accuracy is achieved.

A modification of the spectral DC method appears in [26], in which a parameter  $\theta$  is used to adjust the dependence of the correction steps on previous iterations. The original scheme corresponds to  $\theta = 1$ ; by taking  $\theta \in [0, 1]$  the stability of the method can be improved. Given a fixed order of accuracy and a predictor method, the resulting spectral DC method can be written as a Runge–Kutta method [13]. The algorithm is defined below (the values  $c_j$  denote the locations of the Chebyshev nodes) and depicted in Figure 2. For error estimation, we use the solution from the next-to-last correction iteration, whose order is one less than that of the overall method.

In Algorithm 3,  $\mathcal{I}_{j-1}^j(f(Y_{k-1},:))$  represents the integral of the degree  $p - 1$  polynomial that interpolates the points  $Y_{k-1,j}$  for  $j = 1, \dots, p - 1$ , over the interval





**Figure 2.** Structure of a fourth-order spectral DC step using 3 Euler substeps. Each numbered circle represents a function evaluation, and the numbers indicate the order in which they are performed. The black arrows represent dependencies; the gray arrows are dependencies that vanish when  $\theta = 0$ . Note that node 1 is connected to all other nodes; some of those arrows have been omitted for clarity. Thus the solution at each node depends on all solutions from the previous iteration and, unless  $\theta = 0$ , on its predecessor in the current iteration.

---

```

 $Y_{10} = y_n$ 
for  $k = 1 \rightarrow p - 1$  do ▷ Compute initial prediction
     $Y_{1k} = Y_{1,k-1} + (c_{k+1} - c_k)hf(Y_{1,k-1})$ 
end for
for  $k = 2 \rightarrow p$  do ▷ Compute successive corrections
     $Y_{k0} = y_n$ 
    for  $j = 1 \rightarrow p - 1$  do
         $Y_{kj} = Y_{k,j-1} + h\theta(f(Y_{k,j-1}) - f(Y_{k-1,j-1})) + \mathcal{P}_{j-1}^j(f(Y_{k-1,:}))$ 
    end for
end for
 $y_{n+1} = Y_{p,p-1}$  ▷ New solution value
 $\hat{y}_{n+1} = Y_{p-1,p-1}$  ▷ New solution value

```

---

**Algorithm 3.** Explicit Euler-based deferred correction (DC-Euler).

$[t_n + c_j h, t_n + c_{j+1} h]$ . Thus, for  $\theta = 0$ , the algorithm becomes a discrete version of Picard iteration.

The number of stages per step is

$$s = p(p - 1) \tag{8}$$

unless  $\theta = 0$ , in which case the stages  $Y_{p,j}$  (for  $j < p - 1$ ) need not be computed at all since  $Y_{p,p-1}$  depends only on the  $Y_{p-1,j}$ . Then the number of stages per step reduces to  $(p - 1)^2 + 1$ .

**2.3. Reference Runge–Kutta methods.** In this work we use the following existing Runge–Kutta pairs as benchmarks for evaluating extrapolation and deferred correction methods:

- Fourth order: the embedded formula of Merson 4(3) [16, p. 167]
- Sixth order: the 6(5) pair of Calvo et al. [4], which was found to be the most efficient out of those considered by Hosea and Shampine [17]
- Eighth order: the well-known Prince–Dormand 8(7) pair [30]
- Tenth order: the 10(8) pair of Curtis [6]
- Twelfth order: The 12(9) pair of Ono [29]

It should be stressed that finding pairs of order higher than eight is still very challenging, and the tenth- and twelfth-order pairs here are not expected to be as efficient as that of Prince–Dormand.

### 3. Concurrency

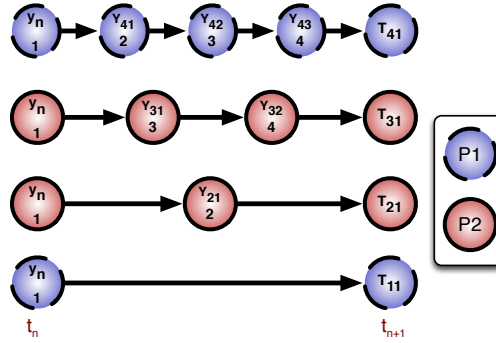
*In view of an implementation on parallel computers, extrapolation methods (as opposed to RKp methods or multistep methods) have an important distinguishing feature: the rows can be computed independently.*

P. Deuffhard, 1985

If a Runge–Kutta method includes stages that are mutually independent, then those stages may be computed concurrently [20]. In this section we investigate theoretically achievable parallel speedup and efficiency of extrapolation and deferred correction methods. Our goal is to determine hardware- and problem- independent upper bounds based purely on algorithmic concerns. We do not attempt to account for machine-specific overhead or communication, although the simple parallel tests in Section 5.5 suggest that the bounds we give are realistically achievable for at least some classes of moderate-sized problems. Previous works that have considered concurrency in explicit extrapolation and deferred correction methods include [33; 36; 31; 2; 14; 11; 21; 27; 28].

**3.1. Computational model and speedup.** As in the serial case, our computational model is based on the assumption that evaluation of  $f$  is sufficiently expensive so that all other operations (e.g., arithmetic, step size selection) are negligible by comparison.

Typically, stage  $y_j$  of an explicit Runge–Kutta method depends on all the previous stages  $y_1, y_2, \dots, y_{j-1}$ . However, if  $y_j$  does not depend on  $y_{j-1}$ , then these two stages may be computed simultaneously on a parallel computer. More generally, by interpreting the incidence matrix of  $A$  as the adjacency matrix of a directed graph  $G(A)$ , one can determine precisely which stages may be computed concurrently and



**Figure 3.** Exploiting concurrency in an Euler extrapolation step using 2 processes. The blue circles with broken border are computed by process 1 and the red circles with solid border are computed by process 2. Observe that only  $s_{\text{seq}} = 4$  sequential function evaluations are required for each process, as opposed to the  $s = 7$  sequential evaluations required in serial.

how much speedup may be achieved. For extrapolation methods, the computation of each  $T_{k1}$  may be performed independently in parallel [9], as depicted in Figure 3. Unlike some previous authors, we do not consider parallel implementation of the extrapolation process (i.e., the second loop in Algorithm 1) since it does not include any evaluations of  $f$  (so our computational model assumes its cost is negligible anyway).

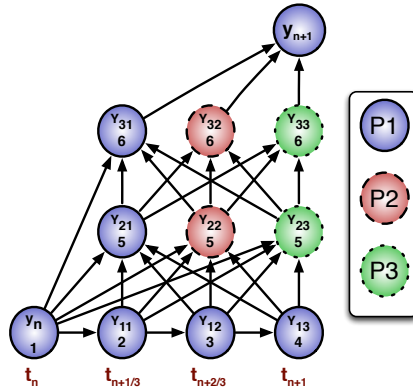
For the deferred correction methods we consider, parallel computation is advantageous only if  $\theta = 0$ ; the resulting parallel algorithm is depicted in Figure 4. A different approach to parallelism in DC methods is taken by the RIDC method [5]; see also [14]. Deferred correction has also been combined with the parareal algorithm to achieve parallel speedup [28; 11; 34].

We define the *minimum number of sequential stages*  $s_{\text{seq}}$  as the minimum number of sequential function evaluations that must be made when parallelism is taken into account. To make this more precise, let us label each node in the graph  $G(A)$  by the index of the stage it corresponds to, with the node corresponding to  $y_{n+1}$  labeled  $s + 1$ . Then

$$s_{\text{seq}} = \max_j \{ \text{path length from node 1 to node } s + 1 \}. \tag{9}$$

The quantity  $s_{\text{seq}}$  represents the minimum time required to take one step with a given method on a parallel computer, in units of the cost of a single derivative evaluation. For instance, the maximum path length for the method shown in Figure 3 is equal to 4; for the method in Figure 4 it is 6. The maximum potential parallel speedup is

$$S = s/s_{\text{seq}}. \tag{10}$$



**Figure 4.** Exploiting concurrency in a fourth-order spectral DC step (with  $\theta = 0$ ) using 3 Euler substeps and 3 processes. The color and border of each circle indicate which process evaluates it. Observe that only  $s_{\text{seq}} = 6$  sequential function evaluations are required for each process, as opposed to the  $s = 10$  sequential evaluations required in serial (12 in serial when  $\theta \neq 0$ ). Node 1 is connected to all other nodes; some of those arrows have been omitted for clarity. More synchronization is required than for a similar extrapolation step.

The minimum number of processes required to achieve speedup  $S$  is denoted by  $P$  (equivalently,  $P$  is the maximum number of processes that can usefully be employed by the method). Finally, let  $E$  denote the theoretical parallel efficiency (here we use the term in the sense that is common in the parallel computing literature) that could be achieved by spreading the computation over  $P$  processes:

$$E = \frac{s}{Ps_{\text{seq}}} = \frac{S}{P}. \quad (11)$$

Note that  $E$  is an upper bound on the achievable parallel efficiency; it accounts only for inefficiencies due to load imbalancing. It does not, of course, account for additional implementation-dependent losses in efficiency due to overhead or communication.

Table 1 shows the parallel algorithmic properties of fixed-order extrapolation and deferred correction methods. Note that for deferred correction methods with  $\theta \neq 0$ , we have  $s_{\text{seq}} = s$ ; that is, no parallel computation of stages is possible.

To our knowledge, no parallel implementation has been made of the deferred correction methods we consider here. However, the parallel iterated RK methods of [35] have a similar flavor. For parallel implementation of a *revisionist* DC method, see [5].

#### 4. Theoretical measures of efficiency

Here we describe the theoretical metrics we use to evaluate the methods. Our metrics are fairly standard; a useful and thorough reference is [22]. The overarching

Method	$s$	$s_{\text{seq}}$	$S$	$P$	$E$
Ex-Euler	$\frac{p^2-p+2}{2}$	$p$	$\frac{p^2-p+2}{2p}$	$\lceil \frac{p}{2} \rceil$	$\frac{p^2-p+2}{2p\lceil \frac{p}{2} \rceil}$
Ex-Midpoint	$\frac{p^2+4}{4}$	$p$	$\frac{p^2+4}{4p}$	$\lceil \frac{p+2}{4} \rceil$	$\frac{p^2+4}{4p\lceil \frac{p+2}{4} \rceil}$
DC-Euler, $\theta = 0$	$(p - 1)^2 + 1$	$2(p - 1)$	$\frac{(p-1)^2+1}{2(p-1)}$	$p - 1$	$\frac{(p-1)^2+1}{2(p-1)}$
DC-Euler, $\theta \neq 0$	$p(p - 1)$	$p(p - 1)$	1	1	—

**Table 1.** Parallel implementation properties of extrapolation and deferred correction methods.  $s$ : number of stages;  $s_{\text{seq}}$ : number of sequentially dependent stages;  $S = s/s_{\text{seq}}$ : optimal speedup;  $P$ : number of processes required to achieve optimal speedup;  $E = S/P$ : parallel efficiency. Note that DC-Euler with  $\theta = 0$  is discrete Picard iteration.

metric for comparing methods is efficiency: the number of function evaluations required to integrate a given problem over a specified time interval to a specified accuracy. We assume that function evaluations are relatively expensive so that other arithmetic operations and overhead for things like step size selection are not significant.

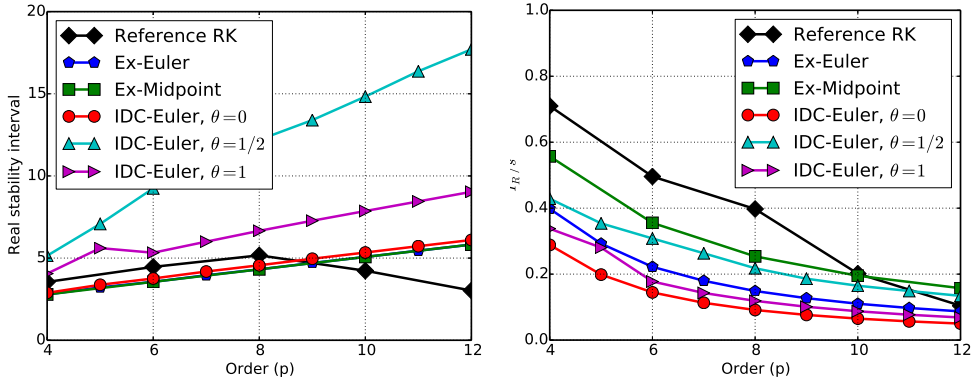
The number of function evaluations is the product of the number of stages of the method and the number of steps that must be taken. The number of steps to be taken depends on the step size  $h$ , which is usually determined adaptively to satisfy accuracy and stability constraints:

$$h = \min(h_{\text{stab}}, h_{\text{acc}}), \tag{12}$$

where  $h_{\text{stab}}, h_{\text{acc}}$  are the maximum step sizes that ensure numerical stability and prescribed accuracy, respectively. Since the cost of a step is proportional to the number of stages of the method,  $s$ , then a fair measure of efficiency is  $h/s$ . A simple observation that partially explains results in this section is as follows: extrapolation and deferred correction are straightforward approaches to creating methods that satisfy the huge numbers of order conditions for very high-order Runge–Kutta methods. However, this straightforward approach comes with a cost: they use many more than the minimum necessary number of stages to achieve a particular order, leading to relatively low efficiency.

**4.1. Absolute stability.** The stable step size  $h_{\text{stab}}$  is typically the limiting factor when a very loose error tolerance is applied. A method’s region of absolute stability (in conjunction with the spectrum of  $f'$ ) typically dictates  $h_{\text{stab}}$ .

In order to make broad comparisons, we measure the size of the and real-axis interval that is contained in the absolute stability region. Specifically, let  $S \subset \mathbb{C}$



**Figure 5.** Comparison of stability regions for reference methods, Euler extrapolation, midpoint extrapolation and deferred correction. Real stability interval (left) and scaled real stability interval (right).

denote the region of absolute stability; then we measure

$$I_{\text{real}} = \max\{r \geq 0 : [-r, 0] \subset S\}, \tag{13}$$

$$I_{\text{imag}} = \max\{r \geq 0 : [-ir, ir] \subset S\}. \tag{14}$$

Determination of the stability region for very high-order methods can be numerically delicate; for instance, the stability function for the eighth-order deferred correction method is a polynomial of degree 56! Because of this, all stability calculations presented here have been performed using exact (rational) arithmetic, not in floating point.

Figure 5 (left) and Table 2 show real and imaginary stability interval sizes for Ex-Euler, Ex-Midpoint, and DC-Euler methods of orders 4–12. We show the real stability intervals of the deferred correction methods with three different values of  $\theta$ , because this interval has a strong dependence on  $\theta$ . For all classes of methods,

Order	Reference RK	Ex-Euler	Ex-Midpoint	DC-Euler, $\theta = 0$
4	3.46	2.83	2.83	2.93
5	—	0	—	0
6	2.61	0	0	0
7	—	1.76	—	1.82
8	0	3.40	3.40	3.52
9	—	0	—	0
10	0	0	0	0
11	—	1.70	—	1.75

**Table 2.** Imaginary stability intervals.

the overall size of the stability region grows with increasing order. However, many methods have  $I_{\text{imag}} = 0$ . Note that the stability regions for Ex-Euler and Ex-Midpoint are identical since both have stability polynomial

$$\sum_{k=0}^p \frac{z^k}{k!}, \quad (15)$$

the degree- $p$  Taylor polynomial of the exponential function.

A fair metric for efficiency is obtained by dividing these interval sizes by the number of stages in the method. The result is shown in Figure 5 (right). Higher-order methods have smaller relative stability regions. For orders  $p \leq 10$ , the reference RK methods have better real stability properties. We caution that, for high-order methods, the boundary of the stability region typically lies very close to the imaginary axis, so values of the amplification factor may differ from unity by less than roundoff over a large interval. For instance, the tenth-order extrapolation method has  $I_{\text{imag}} = 0$ , but the magnitude of its stability polynomial differs from unity by less than  $1.4 \times 10^{-15}$  over the interval  $[-i/4, i/4]$ . It is not clear whether precise measures of  $I_{\text{imag}}$  are relevant for such methods in practical situations. We have omitted the values for DC methods with  $\theta > 0$  because they exhibited extreme sensitivity to small perturbations in  $\theta$ .

Here for simplicity we have considered only the stability region of the principal method; in the design of embedded pairs, it is important that the embedded method have a similar stability region. All the pairs considered here seem to have fairly well matched stability regions.

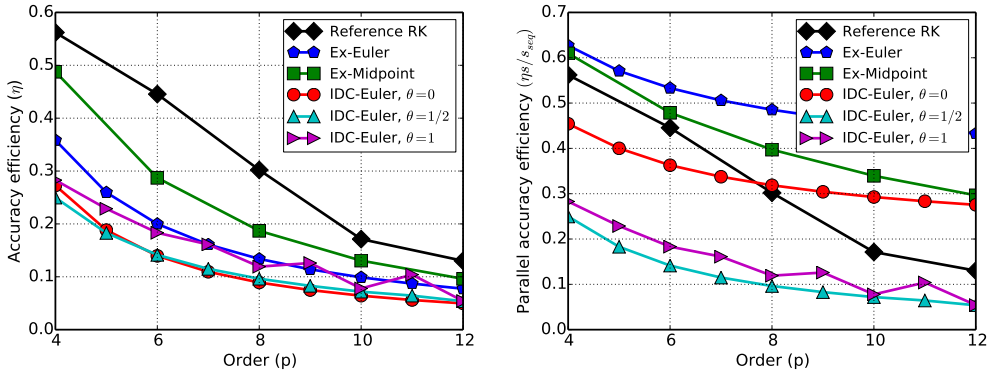
**4.2. Accuracy efficiency.** Typically, the local error is controlled by requiring that  $\|y_{n+1} - \hat{y}_{n+1}\| < \epsilon$  for some tolerance  $\epsilon > 0$ . When the maximum stable step size does not yield sufficient accuracy, the accuracy constraint determines the step size. This is typically the case when the error tolerance is reasonably small. In theoretical analyses, the *principal error norm* [22]

$$C_{p+1} = \left( \sum_k (\tau_k^{(p+1)})^2 \right)^{\frac{1}{2}} \quad (16)$$

is often used as a way to compare accuracy between two methods of the same order. Here the constants  $\tau_k^{(p+1)}$  are the coefficients appearing in the leading order truncation error terms.

Assuming that the one-step error is proportional to  $C_{p+1}h^{p+1}$  leads to a fair comparison of accuracy efficiency given by the *accuracy efficiency index*, introduced in [17]:

$$\eta = \frac{1}{s} \left( \frac{1}{C_{p+1}} \right)^{1/p+1}. \quad (17)$$



**Figure 6.** Accuracy efficiency: serial accuracy (left) and ideal parallel accuracy (right).

Figure 6 (left) plots the accuracy efficiency index for the methods under consideration. Interestingly, a ranking of methods based on this metric gives the same ordering as that based on  $I_{real}/s$ .

**4.3. Accuracy and stability metrics.** In order to determine idealized accuracy and stability efficiency measures, we take the speedup factor  $s/s_{seq}$  into account. In other words, we consider

$$\frac{s}{s_{seq}}\eta = \frac{1}{s_{seq}}\left(\frac{1}{C_{p+1}}\right)^{1/p+1}, \quad (18)$$

as a measure of accuracy efficiency. A similar scaling could be used to study stability efficiency of parallel implementations. We stress that in this context *efficiency* relates to the number of function evaluations required to advance to a given time, and is not related to the usual concept of parallel efficiency.

Figure 6 (right) shows the accuracy efficiency, rescaled by the speedup factor. Comparing with Figure 6 (left), we see a very different picture for methods of order 8 and above. Extrapolation methods are the most efficient, while the reference RK methods give the weakest showing — since they do not benefit from parallelism.

**4.4. Predictions.** The theoretical measures above indicate that fixed-order extrapolation and deferred correction methods are less efficient than traditional Runge–Kutta methods, at least up to order eight. At higher orders, the disadvantage of extrapolation and spectral DC are less pronounced, but they still offer no theoretical advantage. When parallelism is taken into account, extrapolation and deferred correction offer a significant theoretical advantage.



### 5. Performance tests

In this section we perform numerical tests, solving some initial value problems with the methods under consideration, to validate the theoretical predictions of the last section.

In this and all remaining sections of the paper, all results shown for DC methods are for the case  $\theta = 0$ . We focus on this case due to its potential for parallelization. In addition to the tests shown, we tested all methods on a collection of problems known as the nonstiff DETEST suite [18]. The results (not shown here) are broadly consistent with those seen in the test problems below.

**5.1. Verification tests.** For each of the pairs considered, we performed convergence tests using a sequence of fixed step sizes with several nonlinear systems of ODEs, in order to verify that the expected rate of convergence is achieved in practice. We also checked that the coefficients of each method satisfy the order conditions exactly (in rational arithmetic).

**5.2. Step size control.** For step size selection, we use a standard I-controller [22]:

$$h_{n+1}^* = \kappa h_n \left( \frac{\epsilon}{\|\delta_{n+1}\|_\infty} \right)^\alpha. \quad (19)$$

Here  $\epsilon$  is the chosen integration tolerance and  $\delta_{n+1} = y_{n+1} - \hat{y}_{n+1}$ . We take  $\kappa = 0.9$  and  $\alpha = 0.7/p$ , where  $p$  is the order of the embedded method. The step size is not allowed to increase or decrease too suddenly; we use [16]

$$h_{n+1} = \min(\kappa_{\max} h_n, \max(\kappa_{\min} h_n, h_{n+1}^*)) \quad (20)$$

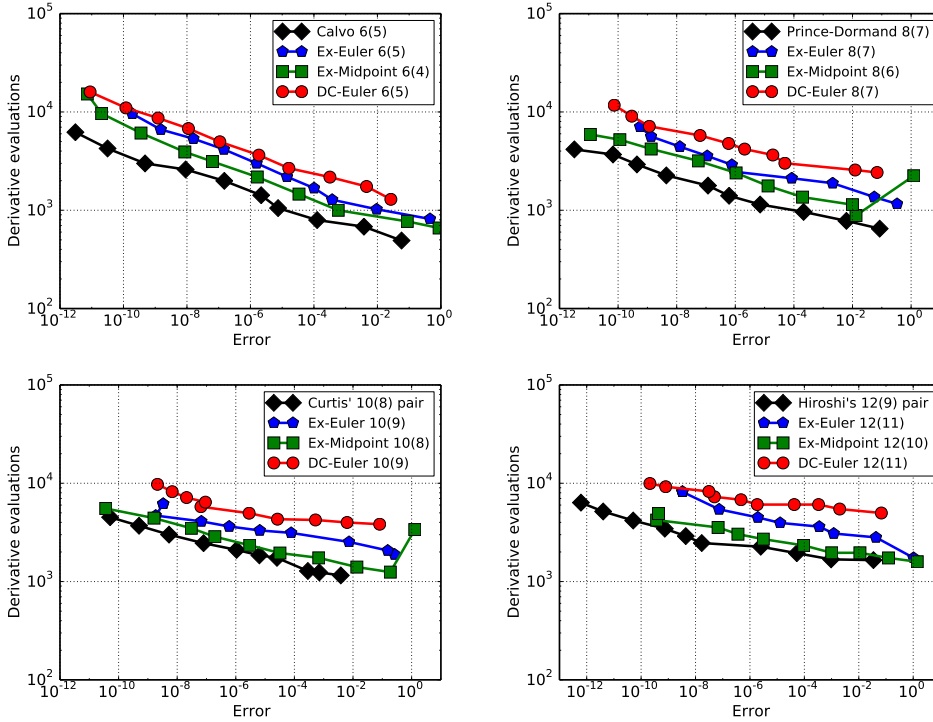
with  $\kappa_{\min} = 0.2$  and  $\kappa_{\max} = 5$ . A step is rejected if the error estimate exceeds the tolerance, i.e., if  $\|\delta_n\|_\infty > \epsilon$ .

All tests in this work were also run with a PI-controller, and very similar results were obtained.

### 5.3. Test problems and results.

**5.3.1. Three-body problem.** We consider the first three-body problem from [32]:

$$\begin{aligned} \text{SB1 : } \quad & y_1' = y_3, \\ & y_2' = y_4, \\ & y_3' = y_1 + 2y_4 - \mu' \frac{y_1 + \mu}{((y_1 + \mu)^2 + y_2^2)^{3/2}} - \mu \frac{y_1 - \mu'}{((y_1 - \mu')^2 + y_2^2)^{3/2}}, \\ & y_4' = y_2 + 2y_3 - \mu' \frac{y_2}{((y_1 + \mu)^2 + y_2^2)^{3/2}} - \mu \frac{y_2}{((y_1 - \mu')^2 + y_2^2)^{3/2}}, \end{aligned} \quad (21)$$



**Figure 7.** Efficiency tests on problem SB1 (Section 5.3.1). Top row: 6th order (left); 8th order (right). Bottom row: 10th order (left); 12th order (right).

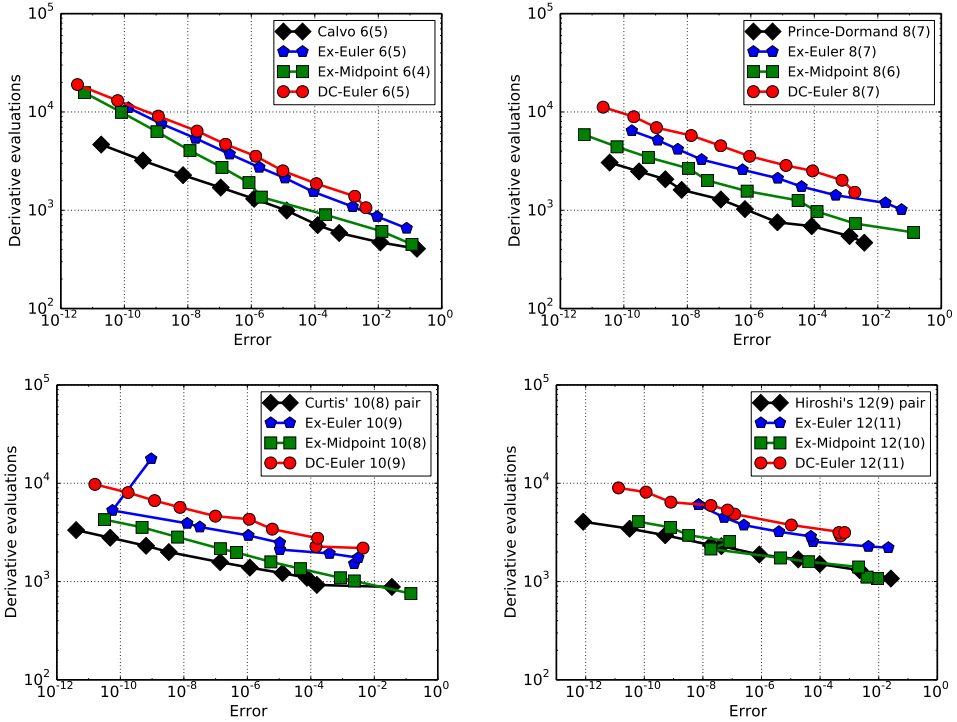
Here  $\mu' = 1 - \mu$ , the final time is  $T = 6.192169331319639$ , and the initial values are

$$\begin{aligned}
 y_1(0) &= 1.2, & y_2(0) &= 0, & y_3(0) &= 0, \\
 y_4(0) &= -1.049357509830319, & \mu &= 0.0121285627653123.
 \end{aligned}
 \tag{22}$$

Figure 7 plots number of function evaluations (cost) against the absolute error for this problem. The absolute error is

$$\text{Error} = |y_N - y(T)|,
 \tag{23}$$

where  $T$  is the final time and  $y_N$  is the numerical solution at that time, while  $y(T)$  is a reference solution computed using a fine grid and the method of Bogacki and Shampine [1]. The initial step size is 0.01. In every case, the method efficiencies follow the ordering predicted by the accuracy efficiency index, and are consistent with previous studies.



**Figure 8.** Efficiency tests on problem B1 (Section 5.3.2). Top row: 6th order (left); 8th order (right). Bottom row: 10th order (left); 12th order (right).

**5.3.2. A two-population growth model.** Next we consider problem B1 of [18], which models the growth of two conflicting populations:

$$y_1' = 2(y_1 - y_1 y_2), \quad y_1(0) = 1, \quad (24a)$$

$$y_2' = -(y_2 - y_1 y_2), \quad y_2(0) = 3. \quad (24b)$$

Results, shown in Figure 8, are consistent with those of the previous test. The effect of internal instability (see Section 5.4) can be seen for the high-order Euler extrapolation methods.

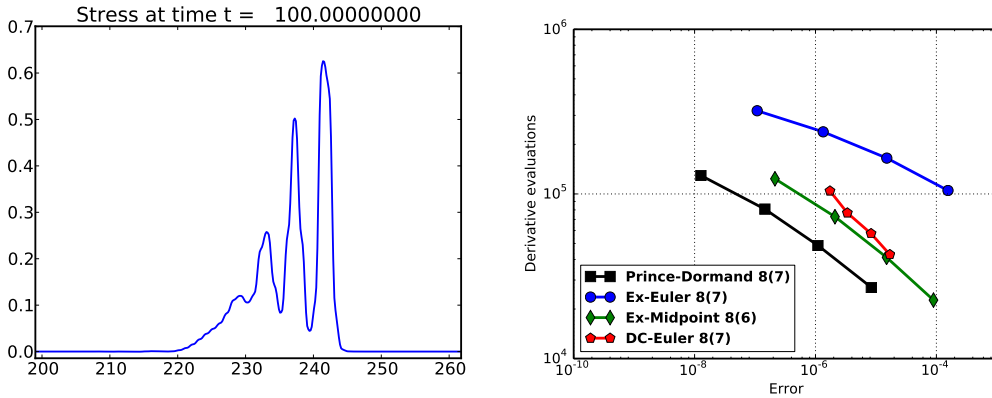
**5.3.3. A nonlinear wave PDE.** Finally, we consider the integration of a high-order PDE semidiscretization from [23]. We solve the 1D elasticity equations

$$\epsilon_t(x, t) - u_x(x, t) = 0, \quad (25a)$$

$$\rho(x)u(x, t)_t - \sigma(\epsilon(x, t), x)_x = 0 \quad (25b)$$

with nonlinear stress-strain relation

$$\sigma(\epsilon, x) = \exp(K(x)\epsilon) - 1, \quad (26)$$



**Figure 9.** Solution (left) and efficiency (right) for methods applied to the stegoton problem (Section 5.3.3).

and a simple periodic medium composed of alternating homogeneous layers:

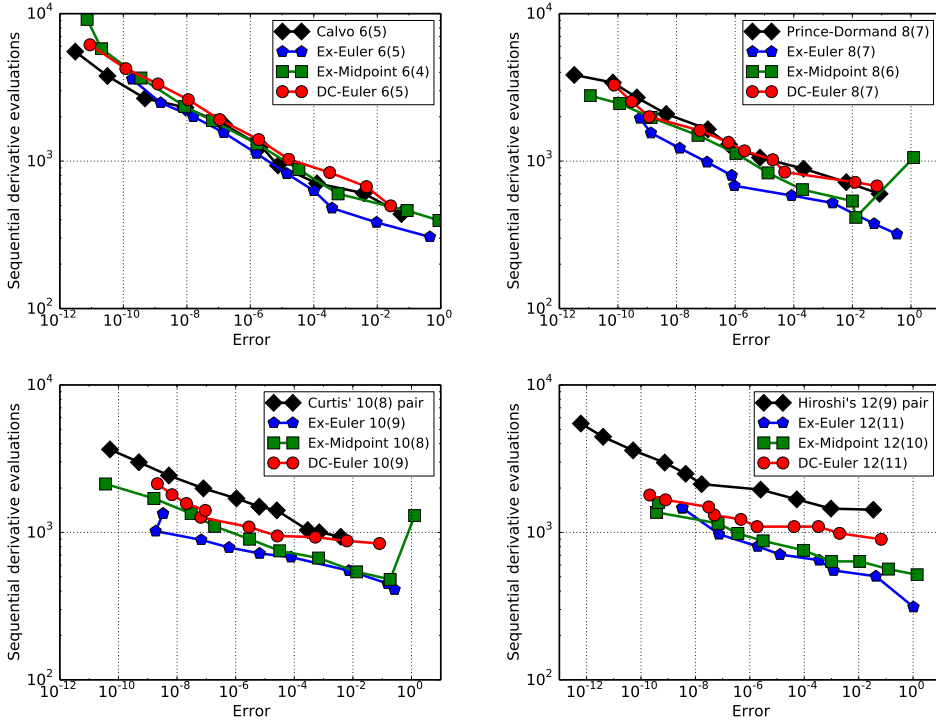
$$\rho(x) = K(x) = \begin{cases} 4 & \text{if } j < x < j + \frac{1}{2} \text{ for some integer } j, \\ 1 & \text{otherwise.} \end{cases} \quad (27)$$

We consider the domain  $0 \leq x \leq 300$ , an initial Gaussian perturbation to the stress, and final time  $T = 100$ . The solution consists of two trains of emerging solitary waves; one of them is depicted in Figure 9 (left). The semidiscretization is based on the WENO wave-propagation method implemented in SharpClaw [25].

Efficiency results for eighth-order methods are shown in Figure 9 (right). The spatial grid is held fixed across all runs, and the time step is adjusted automatically to satisfy the imposed tolerance. The error is computed with respect to a solution computed with tolerance  $10^{-13}$  using the 5(4) pair of Bogacki and Shampine. For the most part, these are consistent with the results from the smaller problems above. However, the Euler extrapolation method performs quite poorly on this problem. The reason is not clear, but this underscores the fact that performance on particular problems can be very different from the “average” performance of a method.

**5.4. Failure of integrators.** Some failure of the integrators was observed in testing. These failures fall into two categories. First, at very tight tolerances, the high-order Euler extrapolation methods were sometimes unable to finish because the time step size was driven to zero. This is a known issue related to internal stability; for a full explanation see [24].

Second, the deferred correction methods sometimes gave global errors much larger than those obtained with the other methods. This indicates a failure of the error estimator. Upon further investigation, we found that the natural embedded error estimator method of order  $p - 1$  satisfies nearly all (typically all but one) of

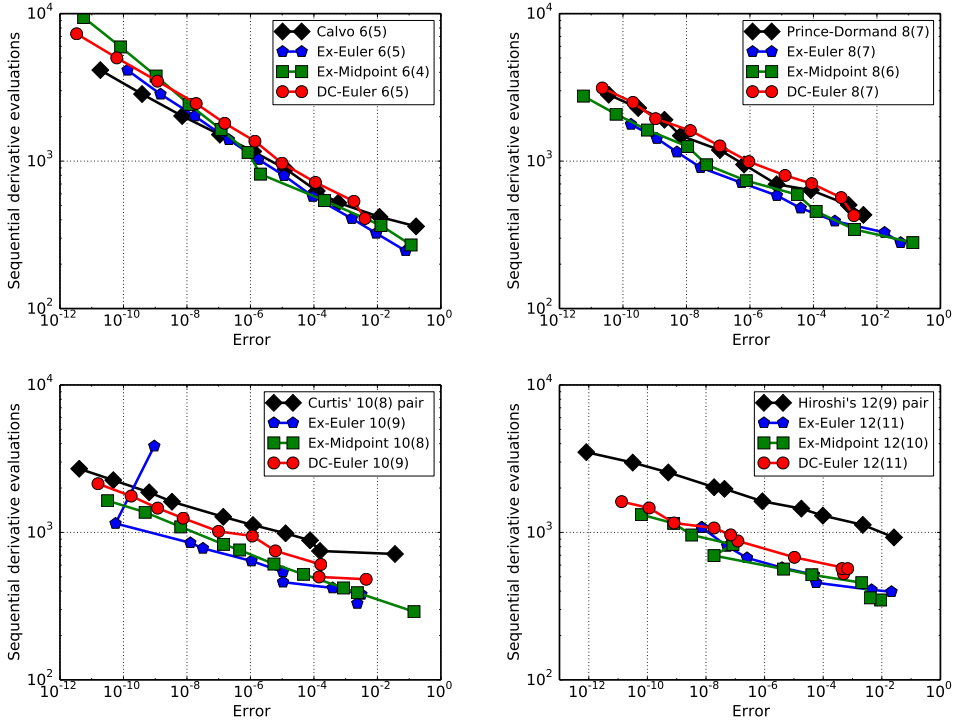


**Figure 10.** Efficiency for the problem SB1 (Section 5.3.1) based on sequential derivative evaluations. Top row: 6th-order methods (left); 8th-order methods (right). Bottom row: 10th-order methods (left); 12th-order methods (right).

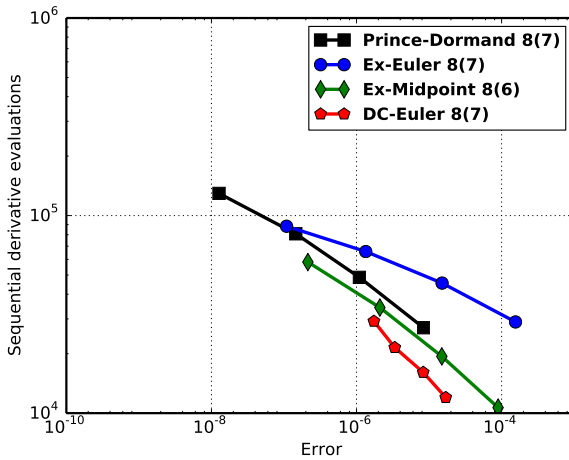
the order conditions for order  $p$ . Hence these estimators may be said to be *defective*, and it would be advisable to employ a more robust approach like that discussed in [10]. Since our focus is purely on Runge–Kutta pairs, we do not pursue this issue further here.

**5.5. Ideal parallel performance.** Figures 10–12 show efficiency for the same three test problems but now based on the number of sequential evaluations. That is, the vertical axis is  $N_{s_{\text{seq}}}$ , where  $N$  denotes the number of steps taken. The same measure of efficiency was used in [35]. We see that the parallelizable methods—especially extrapolation—outperform traditional methods, especially at higher orders. Similar results were obtained for parallel iterated RK methods in [35]. Remarkably, the deferred correction method performs the best by this measure for the stegoton problem.

This measure of efficiency may be viewed with some skepticism since it neglects the cost of communication. This concern is addressed with a true parallel implementation in the next section.



**Figure 11.** Efficiency for the problem B1 (Section 5.3.2) based on sequential derivative evaluations. Top row: 6th-order methods (left); 8th-order methods (right). Bottom row: 10th-order methods (left); 12th-order methods (right).



**Figure 12.** Parallel efficiency for the stegoton problem (Section 5.3.3).

## 6. A shared-memory implementation of extrapolation

Development and testing of a tuned parallel extrapolation or deferred correction code is beyond the scope of this paper, but in this section we run a simple example to demonstrate that it is possible in practice to achieve speedups like those listed in Table 1, and to outperform even the best highly tuned traditional RK methods, at least on problems with an expensive right-hand-side. We focus on speedup with an eye to providing efficient black-box parallel ODE integrators for multicore machines, noting that the number of available cores is often more than can be advantageously used by the methods considered.

Previous studies have implemented explicit extrapolation methods in parallel and achieved parallel efficiencies of up to about 80% [19; 21; 27]. As those studies were conducted about twenty years ago, it is not clear that their conclusions are relevant to current hardware.

In order to test the achievable parallel speedup, we took the code ODEX [16], downloaded from [unige.ch/haier/software.html](http://unige.ch/haier/software.html), and modified it as follows:

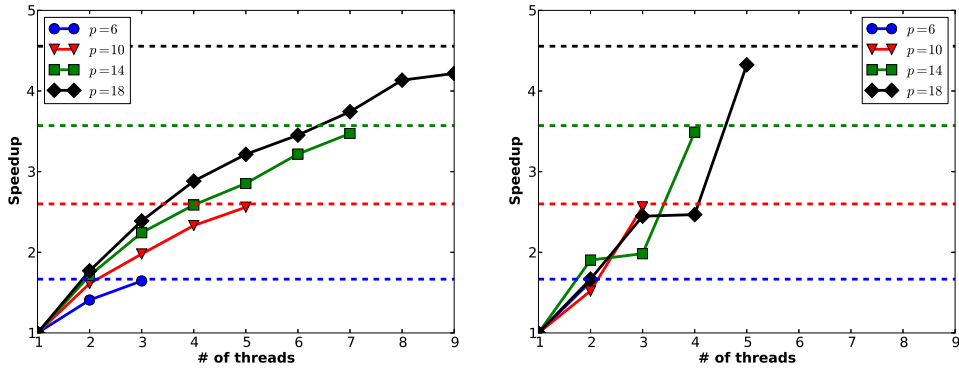
- Fixed the order of accuracy (disabling adaptive order selection)
- Inserted an OMP PARALLEL pragma around the extrapolation loop
- Removed the smoothing step

We refer to the modified code as ODEX-P.

Figure 13 (left) shows the achieved speedup based on dynamic scheduling for  $p = 6, 10, 14, 18$ , applying the code to an  $N$ -body gravitational problem with 400 bodies. Results for other orders are similar. The dotted lines show the theoretical maximum speedup  $S = (p^2 + 4)/(4p)$  based on our earlier analysis. The tests were run on a workstation with two 2.66 GHz quad-core Intel Xeon processors, and the code was compiled using GFortran. Using  $p/2$  threads, the measured speedup is very close to the theoretical maximum. However, the speedup is significantly below the theoretical value when only  $P$  threads are used. We interpret this to mean that the dynamic scheduler is not able to optimally allocate the work among threads unless there are enough threads to give just one loop iteration to each.

Figure 13 (right) and Table 3 show the result of a more intelligent parallel implementation, using static scheduling with the code modified so that both  $T_{k1}$  and  $T_{r-k,1}$  are computed in a single loop iteration. This load balancing scheme is optimal when using on the optimal number of threads  $P$ , and the results agree almost perfectly with theory.

**6.1. Comparison with DOP853.** We now compare actual runtimes of our experimental ODEX-P with the DOP853 code ([unige.ch/haier/prog/nonstiff/dop853.f](http://unige.ch/haier/prog/nonstiff/dop853.f)). These two codes have been compared in [16, Section II.10], but using the original



**Figure 13.** Measured speedup of the midpoint extrapolation code ODEX-P on a 400-body gravitation problem by insertion of a single OMP parallel pragma in the code. Dynamic scheduling (left) and manual load-balancing (right). The ratio of runtime with multiple threads to runtime using a single thread is plotted. The dotted lines show the theoretical maximum speedup  $S = (p^2 + 4)/(4p)$  based on our earlier analysis.

Order ( $p$ )	$P$	Runtime (seconds)		Max. speedup		Parallel efficiency	
		1 thread	$P$ threads	Theory ( $S$ )	Observed	Theory ( $E$ )	Observed
6	2	13.140	7.977	1.67	1.65	0.83	0.82
10	3	17.370	6.770	2.60	2.57	0.87	0.86
14	4	19.508	5.573	3.57	3.50	0.89	0.88
18	5	25.876	5.827	4.56	4.44	0.91	0.89

**Table 3.** Runtime, speedup and efficiency of manually load-balanced runs of the modified ODEX-P code with  $P$  threads. The observed speedup (and efficiency) are close to the theoretically optimal values ( $S$  and  $E$ ).

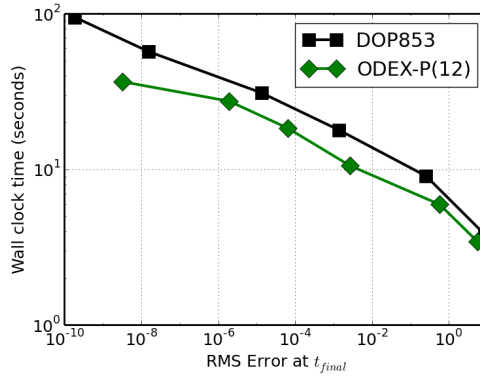
ODEX code (with order-adaptivity and without parallelism). In that reference, DOP853 was shown to be superior to ODEX at all but the most strict tolerances.

Table 4 shows runtimes versus prescribed tolerance for a 400-body problem for the two codes, using fixed order 12 (with 4 threads) in the ODEX-P code. Figure 14 shows the achieved relative root-mean-square global error versus runtime. Perhaps

Code	Tolerance				
	$10^{-3}$	$10^{-5}$	$10^{-7}$	$10^{-9}$	$10^{-11}$
DOP853	3.81	9.02	17.80	30.93	56.75
ODEX-P(12)	3.31	5.87	10.32	18.07	26.76

**Table 4.** Runtimes (in seconds) for Dormand–Prince and modified 12th-order ODEX-P code. The tests were run on a workstation with two 2.66 GHz quad-core Intel Xeon processors using four threads.





**Figure 14.** Runtime versus achieved relative error of the midpoint extrapolation code ODEX-P on a 400-body gravitation problem. The tests were run on a workstation with two 2.66 GHz quad-core Intel Xeon processors using four threads.

surprisingly, the parallel extrapolation code is no worse even at loose tolerances. At moderate to strict tolerances, it substantially outperforms the RK code.

## 7. Discussion

This study is intended to provide a broadly useful characterization of the properties of explicit extrapolation and spectral deferred correction methods. Of course, no study like this can be exhaustive. Our approach handicaps extrapolation and deferred correction methods by fixing the order throughout each computation; practical implementations are order-adaptive and should achieve somewhat better efficiency. We have investigated only the most generic versions of each class of methods; other approaches (e.g., using higher order building blocks or exploiting concurrency in different ways) may give significantly different results. Such approaches could be evaluated using the same kind of analysis employed here. Finally, our parallel computational model is valid only when evaluation of  $f$  is relatively expensive—but that is when efficiency and concurrency are of most interest.

The most interesting new conclusions from the present study is that parallel extrapolation methods of very high order outperform sophisticated implementations of the best available RK methods for problems with an expensive right hand side. This is true even for a relatively naive non-order-adaptive code. We have shown that near-optimal speedup can be achieved in practice with simple modification of an existing code. The resulting algorithm is faster (at least for some problems) than the highly regarded DOP853 code.

Our serial results are in line with those of previous studies. New here is the evidence that, in serial, spectral deferred correction based on explicit Euler seems

(like extrapolation) inferior to well-designed RK methods. However, we have tested only one of the many possible variants of these methods.

High order Euler extrapolation methods suffer from dramatic amplification of roundoff errors. This leads to the loss of several digits of accuracy (and failure of the automatic error control) for very high-order methods, and is observed in practice on most problems. Fortunately, midpoint extrapolation does not exhibit this amplification.

The theoretical and preliminary experimental results we have presented suggest that a carefully designed parallel code based on midpoint extrapolation could be very efficient. Such a practical implementation is the subject of current efforts.

### Acknowledgments

We thank one of the referees, who pointed out a discrepancy that revealed an important bug in our implementation of some spectral deferred correction methods when  $\theta \neq 0$ .

Research reported in this publication was supported by the King Abdullah University of Science and Technology (KAUST).

### References

- [1] P. Bogacki and L. F. Shampine, *An efficient Runge–Kutta (4, 5) pair*, *Comput. Math. Appl.* **32** (1996), no. 6, 15–28. MR 1409687 Zbl 0857.65077
- [2] K. Burrage, *Parallel and sequential methods for ordinary differential equations*, The Clarendon Press, New York, NY, 1995. MR 97f:65021 Zbl 0838.65073
- [3] J. C. Butcher, *Numerical methods for ordinary differential equations*, 2nd ed., John Wiley & Sons, Ltd., Chichester, 2008. MR 2009b:65002 Zbl 1167.65041
- [4] M. Calvo, J. I. Montijano, and L. Rández, *A new embedded pair of Runge–Kutta formulas of orders 5 and 6*, *Comput. Math. Appl.* **20** (1990), no. 1, 15–24. MR 1051749 Zbl 0712.65070
- [5] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, *SIAM J. Sci. Comput.* **32** (2010), no. 2, 818–835. MR 2011g:65105 Zbl 1211.65089
- [6] A. R. Curtis, *High-order explicit Runge–Kutta formulae, their uses, and limitations*, *J. Inst. Math. Appl.* **16** (1975), no. 1, 35–55. MR 52 #4630 Zbl 0317.65024
- [7] J. W. Daniel, V. Pereyra, and L. L. Schumaker, *Iterated deferred corrections for initial value problems*, *Acta Ci. Venezolana* **19** (1968), 128–135. MR 40 #8270
- [8] P. Deuffhard, *Order and stepsize control in extrapolation methods*, *Numer. Math.* **41** (1983), no. 3, 399–422. MR 85b:65062 Zbl 0543.65049
- [9] ———, *Recent progress in extrapolation methods for ordinary differential equations*, *SIAM Rev.* **27** (1985), no. 4, 505–535. MR 86m:65075 Zbl 0602.65047
- [10] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, *BIT* **40** (2000), no. 2, 241–266. MR 2001e:65104 Zbl 0959.65084
- [11] M. Emmett and M. L. Minion, *Toward an efficient parallel in time method for partial differential equations*, *Commun. Appl. Math. Comput. Sci.* **7** (2012), no. 1, 105–132. MR 2979518 Zbl 1248.65106

- [12] T. Feagin, *High-order explicit Runge–Kutta methods using  $m$ -symmetry*, Neural Parallel Sci. Comput. **20** (2012), no. 3–4, 437–458. MR 3057741 Zbl 1278.65107
- [13] S. Gottlieb, D. I. Ketcheson, and C.-W. Shu, *High order strong stability preserving time discretizations*, J. Sci. Comput. **38** (2009), no. 3, 251–289. MR 2010b:65161 Zbl 1203.65135
- [14] D. Guibert and D. Tromeur-Dervout, *Cyclic distribution of pipelined parallel deferred correction method for ODE/DAE*, Parallel Computational Fluid Dynamics 2007, Springer, 2009, pp. 171–178.
- [15] B. Gustafsson and W. Kress, *Deferred correction methods for initial value problems*, BIT **41** (2001), no. 5, suppl., 986–995. MR 2005c:65053
- [16] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations, I: Nonstiff problems*, 2nd ed., Springer Series in Computational Mathematics, no. 8, Springer, Berlin, 1993. MR 94c:65005
- [17] M. E. Hosea and L. F. Shampine, *Efficiency comparisons of methods for integrating ODEs*, Comput. Math. Appl. **28** (1994), no. 6, 45–55. MR 95d:65053 Zbl 0807.65083
- [18] T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick, *Comparing numerical methods for ordinary differential equations*, SIAM J. Numer. Anal. **9** (1972), 603–637; errata, *ibid.* **11**, 681. MR 50 #3577 Zbl 0221.65115
- [19] T. Ito and T. Fukushima, *Parallelized extrapolation method and its application to the orbital dynamics*, Astron. J. **114** (1997), 1260.
- [20] K. R. Jackson and S. P. Nørsett, *The potential for parallelism in Runge–Kutta methods, I: RK formulas in standard form*, SIAM J. Numer. Anal. **32** (1995), no. 1, 49–82. MR 95k:65066
- [21] M. Kappeller, M. Kiehl, M. Perzl, and M. Lenke, *Optimized extrapolation methods for parallel solution of IVPs on different computer architectures*, Appl. Math. Comput. **77** (1996), no. 2–3, 301–315. MR 97b:65157 Zbl 0859.65070
- [22] C. A. Kennedy, M. H. Carpenter, and R. M. Lewis, *Low-storage, explicit Runge–Kutta schemes for the compressible Navier–Stokes equations*, Appl. Numer. Math. **35** (2000), no. 3, 177–219. MR 2001k:65111 Zbl 0986.76060
- [23] D. I. Ketcheson and R. J. LeVeque, *Shock dynamics in layered periodic media*, Commun. Math. Sci. **10** (2012), no. 3, 859–874. MR 2911200 Zbl 1273.35186
- [24] D. I. Ketcheson, L. Lóczi, and M. Parsani, *Internal error propagation in explicit Runge–Kutta discretization of PDEs*, preprint, 2013. arXiv 1309.1317
- [25] D. I. Ketcheson, M. Parsani, and R. J. LeVeque, *High-order wave propagation algorithms for hyperbolic systems*, SIAM J. Sci. Comput. **35** (2013), no. 1, A351–A377. MR 3033052 Zbl 1264.65151
- [26] Y. Liu, C. W. Shu, and M. Zhang, *Strong stability preserving property of the deferred correction time discretization*, J. Comput. Math. **26** (2008), no. 5, 633–656. Zbl 1174.65036
- [27] L. Lustman, B. Neta, and W. Gragg, *Solution of ordinary differential initial value problems on an intel hypercube*, Comput. Math. Appl. **23** (1992), no. 10, 65–72. Zbl 0765.65070
- [28] M. L. Minion, *A hybrid parareal spectral deferred corrections method*, Commun. Appl. Math. Comput. Sci. **5** (2010), no. 2, 265–301. MR 2012e:65118 Zbl 1208.65101
- [29] H. Ono, *On the 25 stage 12th order explicit Runge–Kutta method*, Trans. Japan Soc. Ind. Appl. Math. **16** (2006), no. 3, 177, In Japanese.
- [30] P. J. Prince and J. R. Dormand, *High order embedded Runge–Kutta formulae*, J. Comput. Appl. Math. **7** (1981), no. 1, 67–75. MR 82f:65080 Zbl 0449.65048

- [31] T. Rauber and G. Rünger, *Load balancing schemes for extrapolation methods*, *Concurrency: Practice and Experience* **9** (1997), no. 3, 181–202.
- [32] L. F. Shampine and L. S. Baca, *Fixed versus variable order Runge–Kutta*, *ACM Trans. Math. Software* **12** (1986), no. 1, 1–23. Zbl 0594.65047
- [33] H. H. Simonsen, *Extrapolation methods for ODE's: continuous approximations, a parallel approach*, Ph.D. thesis, University of Trondheim, Norway, 1990.
- [34] R. Speck, D. Ruprecht, R. Krause, M. Emmett, M. Minion, M. Winkel, and P. Gibbon, *A massively space-time parallel N-body solver*, *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (Los Alamitos, CA)*, IEEE Computer Society Press, 2012, pp. 92:1–92:11.
- [35] P. J. van der Houwen and B. P. Sommeijer, *Parallel iteration of high-order Runge–Kutta methods with stepsize control*, *J. Comput. Appl. Math.* **29** (1990), no. 1, 111–127. MR 91a:65179 Zbl 0682.65039
- [36] P. J. van der Houwen and B. P. Sommeijer, *Parallel ODE solvers*, *ACM SIGARCH Computer Architecture News* **18** (1990), no. 3, 71–81.
- [37] Y. Xia, Y. Xu, and C.-W. Shu, *Efficient time discretization for local discontinuous Galerkin methods*, *Discrete Contin. Dyn. Syst. Ser. B* **8** (2007), no. 3, 677–693. MR 2008e:65307 Zbl 1141.65076

Received November 4, 2013. Revised May 4, 2014.

DAVID I. KETCHESON: david.ketcheson@kaust.edu.sa  
*Division of Computer, Electrical, and Mathematical Sciences and Engineering,  
King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia*

UMAIR BIN WAHEED: umairbin.waheed@kaust.edu.sa  
*Division of Physical Sciences and Engineering, King Abdullah University of Science and Technology,  
Thuwal 23955-6900, Saudi Arabia*

## A NEW CLASS OF SECANT-LIKE METHODS FOR SOLVING NONLINEAR SYSTEMS OF EQUATIONS

JOSÉ A. EZQUERRO, ANGELA GRAU, MIQUEL GRAU-SÁNCHEZ  
AND MIGUEL A. HERNÁNDEZ-VERÓN

Applying twice an idea of Hernández and Rubio (2002) for constructing a one-parameter family of secant-like methods, we define a two-parameter family of secant-like methods for solving nonlinear systems of equations. We analyze the efficiency of this new family and conclude that the Kurchatov method, which is one member of the family, is the most efficient. We illustrate this with Troesch's problem.

### 1. Introduction

Iterative methods are typically used for approximating a simple root  $\alpha$  of a nonlinear system of equations, say  $F(x) = 0$ , where  $F \equiv (F_1, F_2, \dots, F_m)$  — each component  $F_i : D \subseteq \mathbb{R}^m \rightarrow \mathbb{R}$ ,  $i = 1, 2, \dots, m$ , being defined on a nonempty open convex domain  $D$  of  $\mathbb{R}^m$ . The choice of a method for solving  $F(x) = 0$  usually depends on its efficiency, which links the order of convergence of the method to its computational cost. Two classic measurements of the efficiency, in the sense defined by Traub [25] and Ostrowski [20], are the efficiency index (EI) and the computational efficiency (CE), by

$$\text{EI} = \rho^{1/a} \quad \text{and} \quad \text{CE} = \rho^{1/p}, \quad (1)$$

where  $\rho$  is the  $R$ -order of convergence of the method [21],  $a$  represents the number of function evaluations necessary to apply the method and  $p$  is the number of multiplications and divisions needed to compute each iteration of the method.

For one-point iterative methods without memory, it is known that the order of convergence  $\rho$  is a natural number and can be achieved for methods that depend explicitly on the first  $\rho - 1$  derivatives of  $F$ . However, the computational cost increases when it is necessary to calculate successive derivatives.

---

This work was supported in part by project MTM2011-28636-C02-01 of the Spanish Ministry of Science and Innovation.

*MSC2010:* 47H99, 65H10.

*Keywords:* nonlinear equations, iterative methods, divided difference, secant method, Kurchatov method, secant-like method, order of convergence, efficiency index, computational efficiency.

In this paper, we are interested in numerical methods that avoid the expensive computation of derivatives of  $F$  at each step. Among such methods, a popular one is the secant method [2; 3], whose algorithm we recall. Given two points  $u = (u_1, \dots, u_m)$  and  $v = (v_1, \dots, v_m)$  in  $\mathbb{R}^m$ , with  $u_i \neq v_i$  for each  $i$ , define the (first-order) *divided difference* of  $F$  with respect to  $u$  and  $v$  as the linear map  $[u, v; F] : \mathbb{R}^m \rightarrow \mathbb{R}^m$  given by the matrix with the following entries:

$$[u, v; F]_{ij} = \frac{1}{u_j - v_j} \left( F_i(u_1, \dots, u_{j-1}, u_j, v_{j+1}, \dots, v_m) - F_i(u_1, \dots, u_{j-1}, v_j, v_{j+1}, \dots, v_m) \right), \quad i, j = 1, 2, \dots, m.$$

The secant method prescribes

$$\begin{cases} x_0, x_{-1} \text{ given in } D, \\ x_{n+1} = x_n - [x_{n-1}, x_n; F]^{-1} F(x_n), \quad n \geq 0. \end{cases} \tag{2}$$

It is superlinearly convergent with  $R$ -order of convergence  $\frac{1}{2}(1 + \sqrt{5})$  [22].

In [13] the authors propose a one-parameter family of secant-like methods for solving  $F(x) = 0$ , containing the secant method and Newton’s method. For a given value of the parameter  $\lambda \in [0, 1]$ , the method prescribes

$$\begin{cases} x_0, x_{-1} \text{ given in } D, \\ y_n = \lambda x_n + (1 - \lambda)x_{n-1}, \quad n \geq 0, \\ x_{n+1} = x_n - [y_n, x_n; F]^{-1} F(x_n), \quad n \geq 0. \end{cases} \tag{3}$$

Clearly (3) reduces to the secant method if  $\lambda = 0$ ; and, if  $F$  is differentiable, (3) reduces to Newton’s method for  $\lambda = 1$ , since in this case  $[u, v; F]$  tends to  $F'(v)$  as  $u \rightarrow v$ . We know from [14; 15] that the  $R$ -order of convergence of (3) is at least the same as that of the secant method for all  $\lambda$ . In practice, the closer  $x_n$  and  $y_n$ , the higher the speed of convergence; indeed, it is shown in [13] that the speed of convergence of (3) increases with  $\lambda \in [0, 1]$ , approaching that of Newton’s method when  $\lambda$  is close to 1.

Following the above idea twice, we can generalize the method to two parameters, one for each component of the divided difference involved in the secant method. Given  $\gamma, \delta \in \mathbb{R}$ , the generalized method prescribes

$$\begin{cases} x_0, x_{-1} \text{ given in } D, \\ y_n = \gamma x_n + (1 - \gamma)x_{n-1}, \quad n \geq 0, \\ z_n = \delta x_n + (1 - \delta)x_{n-1}, \quad n \geq 0, \\ x_{n+1} = x_n - [y_n, z_n; F]^{-1} F(x_n), \quad n \geq 0. \end{cases} \tag{4}$$

As before we have as particular cases the secant method ( $\gamma = 0, \delta = 1$ ) and Newton’s method if  $F$  is differentiable ( $\gamma = 1, \delta = 1$ ). The family (4) also contains Kurchatov’s method [4; 5; 16; 24], which corresponds to the case  $\gamma = 0, \delta = 2$ ; explicitly, this

method prescribes

$$\begin{cases} x_0, x_{-1} \text{ given in } D, \\ x_{n+1} = x_n - [x_{n-1}, 2x_n - x_{n-1}; F]^{-1} F(x_n), \quad n \geq 0. \end{cases} \tag{5}$$

In the one-dimensional case, the Kurchatov method has a geometrical interpretation similar to the secant method [4].

The paper is organized as follows. In Section 2, we determine the order of convergence of (4) in terms of  $\gamma$  and  $\delta$ . In Section 3, we compute the efficiencies (EI and CE) and find the parameter values that maximize it. In Section 4.1 we repeat the analysis using a more general efficiency index, CEI, which takes into account both the number of function evaluations and the number of operations. Finally, in Section 4.2, we give an application to Troesch’s problem [26], illustrating the theoretical results presented in earlier sections.

To summarize, this paper presents a two-parameter family of iterative methods for solving nonlinear systems of equations that generalizes both the secant method and the Kurchatov method, and shows that, within this family, the Kurchatov method (or in some restricted cases the secant method) is the most efficient.

### 2. Order of convergence

*From now on we assume that  $F$  is continuously differentiable four times at  $\alpha \in D$ .*

In this section we state and prove Theorem 1, which gives the order of convergence of the family of iterations defined in (4). We start by writing down the development to fourth order of the divided difference of  $F$ ; this was introduced in [8], following ideas from [11; 12]. See [8] for details.

Thanks to our assumption on the differentiability of  $F$ , we can approximate the divided difference by the derivative of  $F$ , plus corrections up to the fourth derivative:

$$[y, x; F] = F'(\alpha) + \sum_{k=1}^3 \left( \frac{1}{(k+1)!} F^{(k+1)}(\alpha) \sum_{i=0}^k e^{k-i} \tilde{e}^i \right) + W(x, e, \tilde{e}), \tag{6}$$

where  $e = x - \alpha$ ,  $\tilde{e} = y - \alpha$ , the  $(k+1)$ -st derivative  $F^{(k+1)}(\alpha)$  is understood as the appropriate  $(k+1)$ -linear map acting on the  $k$  vectors whose “product” is written under the inner sum, together with the vector on which  $[y, x; F]$  acts, and finally  $W(x, e, \tilde{e})$  is a linear map  $\mathbb{R}^m \rightarrow \mathbb{R}^m$  satisfying  $\|[F'(\alpha)]^{-1} W(x, e, \tilde{e})\| = o(\|e\|^p \|\tilde{e}\|^q)$ , for all  $p, q = 0, 1, 2, 3$  such that  $p + q = 3$ .

*In the sequel we assume that  $F'(\alpha)$  is nonsingular.* We can then introduce the maps

$$A_k = \frac{1}{k!} [F'(\alpha)]^{-1} F^{(k)}(\alpha) \in \mathcal{L}(\mathbb{R}^m \times \underbrace{\cdots}_k \times \mathbb{R}^m, \mathbb{R}^m), \quad k = 2, 3, 4.$$

Also, it will be convenient to write

$$w_k(e)$$

for any vector-valued expression in  $e$  whose norm is  $o(\|e\|^k)$ ; similarly we write

$$w_{j,k}(e, \tilde{e})$$

for any expression in  $e, \tilde{e}$  such that whose norm is  $o(\|e\|^j \|\tilde{e}\|^k)$ . Here  $j, k$  are natural numbers.

**Theorem 1.** *The iterative procedure in (4) has R-order of convergence at least 2 if  $\gamma + \delta = 2$  and at least  $\frac{1}{2}(1 + \sqrt{5})$  if  $\gamma + \delta \neq 2$ . More precisely, if  $F'(\alpha)$  is nonsingular, then*

$$e_{n+1} = A_2 e_n^2 + (1 - \gamma)^2 A_3 e_{n-1}^2 e_n + w_{2,1}(e_{n-1}, e_n) \quad \text{if } \gamma + \delta = 2 \quad (7)$$

and

$$e_{n+1} = (2 - \gamma - \delta) A_2 e_{n-1} e_n + (\gamma + \delta - 1) A_2 e_n^2 + w_2(e_{n-1}) \quad \text{if } \gamma + \delta \neq 2. \quad (8)$$

*Proof.* We set  $y = y_n$  and  $x = z_n$  in (6) to obtain the expression of  $[y_n, z_n; F]$  in terms of  $e_n = x_n - \alpha$ . Then, by expanding in formal power series of  $e_{n-1}$  and  $e_n$  and taking into account that  $[y_n, z_n; F]^{-1}[y_n, z_n; F] = I$ , we obtain

$$\begin{aligned} & [y_n, z_n; F]^{-1} \\ &= \left( I - (2 - \gamma - \delta) A_2 e_{n-1} - (\gamma + \delta) A_2 e_n \right. \\ &\quad \left. - \left( ((1 - \gamma)^2 + (1 - \delta)^2 + (1 - \gamma)(1 - \delta)) A_3 - (2 - \gamma - \delta)^2 A_2^2 \right) e_{n-1}^2 + w_2(e_{n-1}) \right) \\ &\quad \times [F'(\alpha)]^{-1}. \end{aligned}$$

The highest local order of convergence for (4) is obtained when  $\gamma + \delta = 2$ , since then the term  $(2 - \gamma - \delta) A_2 e_{n-1}$  disappears. In this case,  $\delta = 2 - \gamma$  and

$$[y_n, z_n; F]^{-1} = (I - 2A_2 e_n - (1 - \gamma)^2 A_3 e_{n-1}^2 + w_2(e_{n-1})) [F'(\alpha)]^{-1},$$

so that (4) becomes

$$\begin{cases} x_0, x_{-1} \text{ given in } D, \\ x_{n+1} = x_n - [\gamma x_n + (1 - \gamma)x_{n-1}, (2 - \gamma)x_n + (\gamma - 1)x_{n-1}; F]^{-1} F(x_n), \quad n \geq 0. \end{cases} \quad (9)$$

By subtracting the root  $\alpha$  from both sides of (9), we deduce that

$$\begin{aligned} e_{n+1} &= e_n - (I - 2A_2 e_n - (1 - \gamma)^2 A_3 e_{n-1}^2 + w_2(e_{n-1})) [F'(\alpha)]^{-1} F'(\alpha) \\ &\quad \times (e_n + A_2 e_n^2 + w_2(e_n)), \end{aligned}$$

which leads to (7). Taking norms, we then have



$$\|e_{n+1}\| \leq \|A_2\| \|e_n\|^2 + (1 - \gamma)^2 \|A_3\| \|e_{n-1}\|^2 \|e_n\|.$$

Consequently the associated equation is  $t^2 - t - 2 = 0$  [20; 25], whose only positive root is 2. Thus the  $R$ -order of convergence of family (9) is at least 2.

In the other case,  $\gamma + \delta \neq 2$ , we argue as above and deduce (8) and the inequality

$$\|e_{n+1}\| \leq |2 - \gamma - \delta| \|A_2\| \|e_{n-1}\| \|e_n\| + |\gamma + \delta - 1| \|A_2\| \|e_n\|^2.$$

The associated equation is now  $t^2 - t - 1 = 0$ , whose unique positive root is  $\frac{1}{2}(1 + \sqrt{5})$ . Thus the  $R$ -order of convergence is at least  $\frac{1}{2}(1 + \sqrt{5})$ , which is that of the secant method.  $\square$

### 3. Efficiency

We next turn to the efficiency of the two-parameter family of iterative methods (4), comparing it with that of the one-parameter family (3). Having just determined the  $R$ -orders of convergence, we need to find the number of function evaluations and operations (multiplications and divisions) required at each step.

We denote by  $a_1(m)$  and  $p_1(m)$ , respectively, the number of function evaluations and operations (per step) for (3) in dimension  $m$ . For the two-parameter family (4), the corresponding numbers are denoted by  $a_2(m)$  and  $p_2(m)$  in the case  $\gamma + \delta \neq 2$ , and by  $a_3(m)$  and  $p_3(m)$  in the case  $\gamma + \delta = 2$ .

To determine  $a_1(m)$  and  $p_1(m)$ , we rewrite the last line of (3) as

$$x_{n+1} = x_n + b_n, \quad \text{where } [y_n, x_n; F]b_n = -F(x_n). \tag{10}$$

We see that  $m$  evaluations are needed for the  $F_i$  and  $m^2$  for functions in the divided difference matrix, so

$$a_1(m) = m^2 + m.$$

Also needed are  $m^2 + 2m$  operations to compute the divided difference matrix (counting  $y_n = \lambda x_n + (1 - \lambda)x_{n+1}$  as two multiplications),  $\frac{1}{3}(m^3 - m)$  operations for its LU decomposition, and  $m^2$  operations to solve two triangular linear systems. Therefore

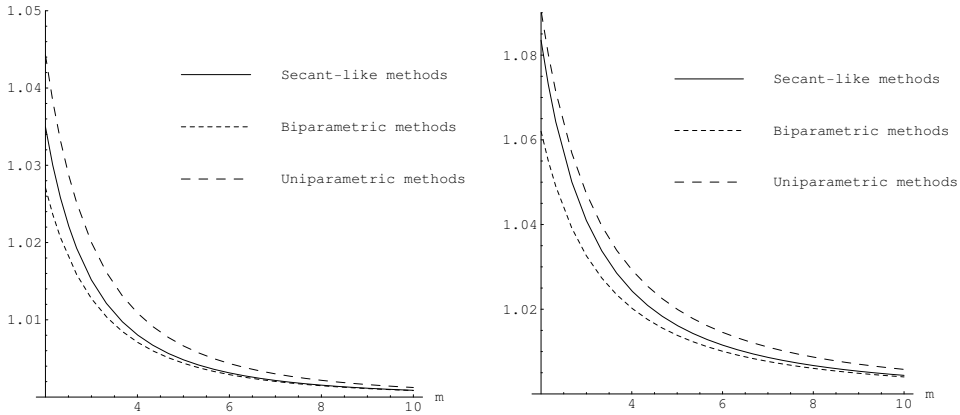
$$p_1(m) = \frac{1}{3}m(m^2 + 6m + 5).$$

With these two values we can then compute the two measures of efficiency,

$$EI = \left(\frac{1 + \sqrt{5}}{2}\right)^{1/a_1(m)} \quad \text{and} \quad CE = \left(\frac{1 + \sqrt{5}}{2}\right)^{1/p_1(m)}. \tag{11}$$

Similarly, for (4) with  $\gamma + \delta \neq 2$ , we can write

$$x_{n+1} = x_n + c_n, \quad \text{where } [y_n, z_n; F]c_n = -F(x_n). \tag{12}$$



**Figure 1.** Plots of EI (left) and CE (right) versus the dimension  $m$ . The bottom curves refer to the general two-parameter algorithm (4); the efficiency indices are given in (11). The middle curves refer to the “secant-like” specialization ( $\delta = 0$ ); see (13). The top curves refer to the specialization  $\gamma + \delta = 2$ ; see (14).

In this case we get  $a_2(m) = a_1(m) + m = m^2 + 2m$  function evaluations and  $p_2(m) = p_1(m) + 2m = \frac{1}{3}m(m^2 + 6m + 8)$  operations (because  $z_n$ , too, requires two multiplications). This leads to

$$EI = \left(\frac{1 + \sqrt{5}}{2}\right)^{1/(a_1(m)+m)} \quad \text{and} \quad CE = \left(\frac{1 + \sqrt{5}}{2}\right)^{1/(p_1(m)+2m)}. \quad (13)$$

Finally we take (4) with  $\gamma + \delta = 2$ . Equation (12) is still valid; however, since  $z_n = (2 - \gamma)x_n + (\gamma - 1)x_{n-1}$  shares a summand with  $y_n$ , it requires one fewer multiplication. Consequently,  $a_3(m) = a_1(m) + m = m^2 + 2m$  and  $p_3(m) = p_1(m) + m = \frac{1}{3}m(m^2 + 6m + 8)$ . In this case we obtain

$$EI = 2^{1/(a_1(m)+m)} \quad \text{and} \quad CE = 2^{1/(p_1(m)+m)}. \quad (14)$$

The results are summarized in Figure 1. We see that both measures of efficiency, the EI the the CE, are highest for the case  $\gamma + \delta = 2$  of the two-parameter family (4). Within this spacial case, the Kurchatov method ( $\gamma = 0, \delta = 2$ ) is the most efficient of all, since it saves  $m$  function evaluations and  $3m$  multiplications.

### 4. Applications

As already discussed, the EI and CE are based, respectively, on the number of function evaluations and the number of operations. To take both into account at once we can use what we call the *computational efficiency index*, defined as

$$CEI = \rho^{1/\epsilon}. \quad (15)$$

Here  $\rho$  is the  $R$ -order of convergence and  $\mathcal{C}$  is the computational cost per step, defined as the number of operations plus  $\mu$  times the number of function evaluations. The factor  $\mu$  reflects the cost of a function evaluation relative to that of an operation, and depends on the machine, the software and the arithmetic used. (Some discussion of the CEI can be found in [22].) In Section 4.1 we use the CEI to refine the analysis of the previous section. In Section 4.2 we illustrate with Troesch’s problem [26].

**4.1. Optimal computational efficiency.** We have seen in Section 3 that two special cases stand out for efficiency among the algorithm of the family (4): the secant method (2), with  $(\gamma, \delta) = (0, 1)$ , and the Kurchatov method (5), with  $(\gamma, \delta) = (0, 2)$ . Combining the definition of  $\mathcal{C}$  in the previous paragraph with the results of Section 3, we have

$$\begin{aligned} \mathcal{C}^{\text{sec}}(\mu, m) &= m^2 \mu + \frac{1}{3}m(m^2 + 6m - 1), & \rho^{\text{sec}} &= \frac{1}{2}(1 + \sqrt{5}), \\ \mathcal{C}^{\text{Kur}}(\mu, m) &= (m^2 + m)\mu + \frac{1}{3}m(m^2 + 6m - 1), & \rho^{\text{Kur}} &= 2. \end{aligned}$$

**Theorem 2.** *If  $m = 2$ , then  $\text{CEI}_{\text{sec}} > \text{CEI}_{\text{Kur}}$  for  $\mu > \mu_0 \approx 18.48023$ , and  $\text{CEI}_{\text{Kur}} > \text{CEI}_{\text{sec}}$  for  $\mu < \mu_0$ . If  $m \geq 3$ , then  $\text{CEI}_{\text{Kur}} > \text{CEI}_{\text{sec}}$ .*

*Proof.* It is enough to consider the borderline case of the ratio

$$\frac{\log \text{CEI}^{\text{sec}}}{\log \text{CEI}^{\text{Kur}}} = \frac{\mathcal{C}^{\text{Kur}} \log \rho^{\text{sec}}}{\mathcal{C}^{\text{sec}} \log \rho^{\text{Kur}}}.$$

Equating this ratio to 1 gives a curve in the  $(m, \mu)$  plane with vertical asymptote  $m = 2.270559\dots$ . For higher  $m$ , the ratio is always less than 1. For  $m = 2$ , the ratio is less than 1 if and only if  $\mu > \mu_0$ . □

Therefore, the CEI of the Kurchatov method is almost always better than that of the secant method.

**4.2. Troesch’s problem.** Troesch’s problem [26] is the following nonlinear two-point boundary value problem in one dimension:

$$u''(x) = \lambda \sinh(\lambda u(x)), \quad 0 \leq x \leq 1, \tag{16}$$

with boundary conditions  $u(0) = 0$  and  $u(1) = 1$  and the real positive parameter  $\lambda$ . It arises from modeling the confinement of a plasma column by radiation pressure. A closed-form solution to the problem is known [6; 7; 9; 17; 23]. It can be written in terms of the Jacobian elliptic function  $\text{sc}$  as

$$u(x) = \frac{2}{\lambda} \sinh^{-1} \left\{ 12u'(0) \text{sc}(\lambda x, 1 - \frac{1}{4}u'(0)^2) \right\}, \tag{17}$$

where  $u'(0)$  is the derivative at  $t = 0$ ; we have  $u'(0) = 2\sqrt{1 - \kappa}$ , where  $\kappa$  is the

digits	$x * y$	$x/y$	$\sqrt{x}$	$\exp(x)$
32	1.1 $\mu$ s	1	11	25

**Table 1.** Estimated computational cost of elementary functions computed with Maple@13 on an Intel® Core™ 2 Duo CPU P8800 (32-bit machine) running Microsoft Windows 7 Professional, where  $x = \sqrt{3} - 1$  and  $y = \sqrt{5}$ . The last three entries are relative to the second (multiplication).

solution to the equation

$$\frac{\sinh(\lambda/2)}{\sqrt{1 - \kappa}} = \text{sc}(\lambda, \kappa). \tag{18}$$

Given a value of  $\lambda$ , we can find  $\kappa$  from (18) and the defining equation  $\text{sc}(\lambda, \kappa) = \tan \phi$ , where  $\phi$  is determined by

$$\int_0^\phi \frac{d\theta}{\sqrt{1 - \kappa \sin^2 \theta}} = \lambda$$

(see [1]). Following [7; 9] we consider two cases:  $\lambda = 0.5$  and  $\lambda = 1$ .

In the remainder of this section we use two finite-difference schemes to solve Troesch’s problem numerically, using the closed-form solution for comparison. The numerical computations were performed using Maple with 32 digits. To specify the computational cost, an estimation of the factor  $\mu$  is necessary. We used the data in Table 1, based on [10; 19].

*A classic finite difference scheme.* We partition the interval  $[0, 1]$  as follows:

$$x_0 = 0 < x_1 < x_2 < \dots < x_{n-1} < x_n = 1, \quad x_{j+1} = x_j + h, \quad h = 1/n, \tag{19}$$

and define  $y_0 = y(x_0) = 0$ ,  $y_1 = y(x_1)$ ,  $\dots$ ,  $y_{n-1} = y(x_{n-1})$ ,  $y_n = y(x_n) = 1$ . If we discretize (16) by using the standard numerical formula for the second derivative,

$$y_k'' = \frac{y_{k-1} - 2y_k + y_{k+1}}{h^2} + O(h^2), \quad k = 1, 2, \dots, n - 1, \tag{20}$$

we obtain the following system of  $(n - 1) \times (n - 1)$  nonlinear equations:

$$y_{k-1} - (2y_k + h^2\lambda \sinh(\lambda y_k)) + y_{k+1} = 0, \quad k = 1, 2, \dots, n - 1. \tag{21}$$

Setting  $n = 20$ , the approximate solution is computed taking the initial points  $\mathbf{x}_{-1} = (1, 1, \dots, 1)$  and  $\mathbf{x}_0 = (0, 0, \dots, 0)$  and applying methods (2) and (5), the secant and Kurchatov methods.

The errors in the solution are shown in Table 2 and more information about efficiencies is shown in Table 3. The cost is computed in the following way: the cost of the function  $\sinh$  is 27 (25 for the exponential function plus 2 divisions); each component function  $F_k$  is equal to  $y_{k-1} - (2y_k + h^2\lambda \sinh(\lambda y_k)) + y_{k+1}$  with

x	$\lambda = 0.5$			$\lambda = 1.0$		
	$u(x)$	$ u(x) - y(x) $	$ u(x) - z(x) $	$u(x)$	$ u(x) - y(x) $	$ u(x) - z(x) $
0.1	0.095944349292	$4.1627 \cdot 10^{-7}$	$3.4372 \cdot 10^{-12}$	0.084661256551	$5.9888 \cdot 10^{-6}$	$5.6178 \cdot 10^{-11}$
0.2	0.192128747660	$8.0952 \cdot 10^{-7}$	$6.6447 \cdot 10^{-12}$	0.170171358178	$1.1732 \cdot 10^{-5}$	$1.0262 \cdot 10^{-10}$
0.3	0.288794400893	$1.1563 \cdot 10^{-6}$	$9.3965 \cdot 10^{-12}$	0.257393908080	$1.6965 \cdot 10^{-5}$	$1.3041 \cdot 10^{-10}$
0.4	0.386184846362	$1.4323 \cdot 10^{-6}$	$1.1475 \cdot 10^{-11}$	0.347222855110	$2.1385 \cdot 10^{-5}$	$1.3243 \cdot 10^{-10}$
0.5	0.484547164744	$1.6118 \cdot 10^{-6}$	$1.2675 \cdot 10^{-11}$	0.440599835168	$2.4626 \cdot 10^{-5}$	$1.0472 \cdot 10^{-10}$
0.6	0.584133248445	$1.6674 \cdot 10^{-6}$	$1.2810 \cdot 10^{-11}$	0.538534398077	$2.6221 \cdot 10^{-5}$	$4.8544 \cdot 10^{-11}$
0.7	0.685201148302	$1.5690 \cdot 10^{-6}$	$1.1717 \cdot 10^{-11}$	0.642128609191	$2.5561 \cdot 10^{-5}$	$2.6357 \cdot 10^{-11}$
0.8	0.788016522650	$1.2837 \cdot 10^{-6}$	$9.2672 \cdot 10^{-12}$	0.752608094046	$2.1818 \cdot 10^{-5}$	$9.6507 \cdot 10^{-11}$
0.9	0.892854216136	$7.7458 \cdot 10^{-7}$	$5.3721 \cdot 10^{-12}$	0.871362519798	$1.3843 \cdot 10^{-5}$	$1.1578 \cdot 10^{-10}$

**Table 2.** Exact and approximate solutions  $u(x)$ ,  $y(x)$  and  $z(x)$  defined in (17), (21) and (22), respectively.

$\lambda = 0.5$										
	$I$	$a$	$a\mu$	$\nu$	$\mathcal{E}$	EI	CE	CEI	TF	$\tau$
method (2)	3	$3m$	$87m$	$6m - 4$	1763	1.0084780	1.0043842	1.0002730	8435.91	0.024516
method (5)	2	$3m$	$87m$	$6m - 4$	1763	1.0122347	1.0063212	1.0003932	5856.56	0.018875
$\lambda = 1.0$										
	$I$	$a$	$a\mu$	$\nu$	$\mathcal{E}$	EI	CE	CEI	TF	$\tau$
method (2)	3	$3m$	$84m$	$6m - 4$	1706	1.0084780	1.0043842	1.0002821	8163.16	0.024438
method (5)	2	$3m$	$84m$	$6m - 4$	1706	1.0122347	1.0063212	1.0004064	5667.21	0.019000

**Table 3.** Numerical efficiency for system (21) with  $m = 19$ .

$h^2 = 1/400$  and  $h^2\lambda$  prefixed, so that we have an evaluation of sinh and 2 products if  $\lambda = 0.5$  (in total 29), whereas we have an evaluation of sinh and 1 product (in total 28) if  $\lambda = 1$ . In short,  $\mu_{\lambda=0.5} = 29$  and  $\mu_{\lambda=1} = 28$ .

*A nonstandard finite difference scheme.* As a consequence of the low accuracy obtained in the previous section, we now discretize (16) in a different way. We again consider the partition of the interval  $[0, 1]$  given in (19), define  $z_0 = z(x_0) = 0$ ,  $z_1 = z(x_1)$ ,  $\dots$ ,  $z_{n-1} = z(x_{n-1})$ ,  $z_n = z(x_n) = 1$  and discretize (16) by using the following smart numerical formula for the second derivative [7]:

$$z''_k = \frac{w_k^2(z_{k-1} - 2z_k + z_{k+1})}{2(\cosh(w_k h) - 1)} + O(h^4), \quad k = 1, 2, \dots, n - 1,$$

where

$$w_k = \lambda \sqrt{\frac{(z_{k+1} - z_{k-1})^2}{4h^2} + \cosh(\lambda z_k)}.$$

Next, we obtain the following system of  $(n - 1) \times (n - 1)$  nonlinear equations:

$$w_k^2(z_{k+1} - 2z_k + z_{k-1}) - 2\lambda \sinh(\lambda z_k)(\cosh(w_k h) - 1) = 0, \quad k = 1, \dots, n-1. \quad (22)$$

Setting  $n = 20$ , the approximate solution is computed taking the initial points

$$\mathbf{x}_{-1} = (.0480, .0959, .144, .192, .240, .289, .337, .386, .435, .485, \\ .534, .584, .634, .685, .736, .788, .840, .893, .946)^t,$$

and

$$\mathbf{x}_0 = (.047957, .095944, .14399, .19213, .24039, .28879, .33738, .38618, .43523, .48455, \\ .53417, .58413, .63447, .68520, .73637, .78802, .84016, .89285, .94612)^t,$$

and applying again methods (2) and (5). The errors in the solution are shown in Table 2 and more information about efficiencies is given in Table 4. The cost of the function  $\cosh$  is the same as that of  $\sinh$  if this function is not computed before. In this case, the cost is 1. Every component function  $F_k$  is equal to  $w_k^2(z_{k+1} - 2z_k + z_{k-1}) - 2\lambda \sinh(\lambda z_k)(\cosh(hw_k) - 1)$ ,  $w_k^2$  with cost equal to 31 or 29,  $w_k$  with cost equal to 11,  $\sinh(\lambda z_k)$  with cost equal to 28 or 27,  $\cosh(w_k h)$  with cost equal to 39 (27 (cosh) + 11 (sqrt) + 1 (prod)), and some isolated products. Finally, we obtain  $\mu_{\lambda=0.5} = 73$  and  $\mu_{\lambda=1} = 72$ .

In Table 2, for  $\lambda = 0.5$  and  $\lambda = 1$ , we present the exact solution  $u(x_\ell)$ , the numerical solution  $y(x_\ell)$  of (21) and the numerical solution  $z(x_\ell)$  of (22), where  $\ell = 1, 2, \dots, 9$  and  $k = 2\ell$ , which  $k$  is given in (21) and (22). In both cases the results are independent of the application of methods (2) and (5).

Table 2 confirms the theoretical results. It is interesting that inaccurate tabulated “exact” solutions are given in [9; 18], but those numerical results would approximate exact results more closely if their calculations of the later were properly done.

Tables 3 and 4 show the results obtained for both methods. In each table we show the number of iterations,  $I$ , needed to get the required precision, the computational cost  $\mathcal{C}$ , the computational efficiency index CEI defined in (15) and the time factor  $TF$  defined by  $1/\log(\text{CEI})$ . If the values of the CEI are so close as to be almost indistinguishable in practice, we can then observe the  $TF$  that tell better the difference between iterative methods. While in the definition of the CEI we have considered functions with the divided difference full of terms, we observe that the two given discretizations to solve Troesch’s problem provide a tridiagonal operator.

If both methods are applied to solve systems of  $m$  nonlinear equations, we have to solve a triangular linear system per iteration,  $2(m - 1)$  operations (products and divisions) are required in the LU decomposition,  $2m - 1$  operations in the backward substitution and  $m - 1$  operations in the forward substitution. Therefore, the number

$\lambda = 0.5$										
	$I$	$a$	$a\mu$	$\nu$	$\mathcal{C}$	EI	CE	CEI	TF	$\tau$
method (2)	5	$5m-2$	$73(5m-2)$	$8m-6$	7028	1.005188	1.003301	1.000069	33183.78	.155234
method (5)	5	$5m-2$	$73(5m-2)$	$8m-6$	7028	1.007481	1.004759	1.000099	23163.80	.147970
$\lambda = 1.0$										
	$I$	$a$	$a\mu$	$\nu$	$\mathcal{C}$	EI	CE	CEI	TF	$\tau$
method (2)	5	$5m-2$	$72(5m-2)$	$8m-6$	6842	1.005188	1.003301	1.000070	32738.78	.151870
method (5)	4	$5m-2$	$72(5m-2)$	$8m-6$	6842	1.007481	1.004759	1.000101	22728.63	.131109

**Table 4.** Numerical efficiency for system (22) with  $m = 19$ .

of operations needed per iteration is  $5m - 4$  for both methods. The number of function evaluations is computed in the following way:

- We have  $m$  evaluations of the function  $F: F_1, F_2, \dots, F_m$ .
- For the classic finite difference scheme, we have  $2m$  evaluations and  $m$  divisions to evaluate the divided difference, so that  $\mathcal{C} = 3m\mu + 6m - 4$ .
- For the nonstandard finite difference scheme, we have to compute  $4m - 2$  evaluations and  $3m - 2$  divisions in the divided difference matrix, so that  $\mathcal{C} = (5m - 2)\mu + 8m - 6$ .

Tables 3 and 4 confirm the theoretical results. For Troesch’s problem, the costs are the same and we then observe that method (5) has the highest value of CEI in all cases, since  $CEI^{Kur} > CEI^{sec}$ , which confirms the results of Section 4.1.

### 5. Concluding remarks

We present a two-parameter family of iterative methods for solving nonlinear systems of equations with local  $R$ -order of convergence higher than other competitive iterative methods. Between the members of the family we point out the Kurchatov method and the secant method. Moreover, we analyze a generalization of the efficiency used in the one-dimensional case to several variables. Finally, we show an application, where Troesch’s problem is considered, which illustrates the theoretical results presented in the paper and conclude that the Kurchatov method is more efficient than the secant method for solving Troesch’s problem.

### References

[1] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, National Bureau of Standards Applied Mathematics Series, no. 55, U.S. Government Printing Office, Washington, DC, 1964, Reprinted by Dover, New York, 1974. MR 29 #4914 Zbl 0171.38503

- [2] I. K. Argyros, *The secant method and fixed points of nonlinear operators*, Monatsh. Math. **106** (1988), no. 2, 85–94. MR 90b:65111 Zbl 0652.65043
- [3] ———, *On the secant method*, Publ. Math. Debrecen **43** (1993), no. 3-4, 223–238. MR 95j:47077 Zbl 0796.65075
- [4] ———, *On a two-point Newton-like method of convergent order two*, Int. J. Comput. Math. **82** (2005), no. 2, 219–233. MR 2158994 Zbl 1068.65070
- [5] ———, *A Kantorovich-type analysis for a fast iterative method for solving nonlinear equations*, J. Math. Anal. Appl. **332** (2007), no. 1, 97–108. MR 2008g:65075 Zbl 1121.65061
- [6] J. P. Boyd, *One-point pseudospectral collocation for the one-dimensional Bratu equation*, Appl. Math. Comput. **217** (2011), no. 12, 5553–5565. MR 2770174 Zbl 1222.65070
- [7] U. Erdogan and T. Ozis, *A smart nonstandard finite difference scheme for second order nonlinear boundary value problems*, J. Comput. Phys. **230** (2011), no. 17, 6464–6474. MR 2012m:65218 Zbl 05992164
- [8] J. A. Ezquerro, M. Grau-Sánchez, A. Grau, M. A. Hernández, M. Noguera, and N. Romero, *On iterative methods with accelerated convergence for solving systems of nonlinear equations*, J. Optim. Theory Appl. **151** (2011), no. 1, 163–174. MR 2012j:65145 Zbl 1226.90103
- [9] X. Feng, L. Mei, and G. He, *An efficient algorithm for solving Troesch’s problem*, Appl. Math. Comput. **189** (2007), no. 1, 500–507. MR 2330227 Zbl 1122.65373
- [10] L. Fousse, G. Hanrot, V. Lefèvre, P. Péliissier, and P. Zimmermann, *MPFR: a multiple-precision binary floating-point library with correct rounding*, ACM Trans. Math. Software **33** (2007), no. 2, Article ID #13. MR 2008e:65157
- [11] M. Grau-Sánchez, À. Grau, and M. Noguera, *Frozen divided difference scheme for solving systems of nonlinear equations*, J. Comput. Appl. Math. **235** (2011), no. 6, 1739–1743. MR 2012a:65132 Zbl 1204.65051
- [12] M. Grau-Sánchez and M. Noguera, *A technique to choose the most efficient method between secant method and some variants*, Appl. Math. Comput. **218** (2012), no. 11, 6415–6426. MR 2879122 Zbl 06036020
- [13] M. A. Hernández and M. J. Rubio, *A uniparametric family of iterative processes for solving nondifferentiable equations*, J. Math. Anal. Appl. **275** (2002), no. 2, 821–834. MR 2003i:47077 Zbl 1019.65036
- [14] M. A. Hernández, M. J. Rubio, and J. A. Ezquerro, *Secant-like methods for solving nonlinear integral equations of the Hammerstein type*, J. Comput. Appl. Math. **115** (2000), no. 1-2, 245–254. MR 2000m:65157 Zbl 0944.65146
- [15] ———, *Solving a special case of conservative problems by secant-like methods*, Appl. Math. Comput. **169** (2005), no. 2, 926–942. MR 2006g:65078 Zbl 1080.65044
- [16] V. A. Kurčatov, *A certain linear interpolation method for solving functional equations*, Dokl. Akad. Nauk SSSR **198** (1971), 524–526, In Russian; translated in *Soviet Math. Dokl.* **12** (1971) 835–838. MR 45 #6211 Zbl 0252.65044
- [17] Y. Lin, J. A. Enszer, and M. A. Stadtherr, *Enclosing all solutions of two point boundary value problems for ODE’s*, Comput. Chem. Eng. **32** (2008), 1714–1725.
- [18] S. T. Mohyud-Din, *Solution of Troesch’s problem using He’s polynomials*, Rev. Un. Mat. Argentina **52** (2011), no. 1, 143–148. MR 2815720 Zbl 05965157
- [19] *The MPFR library 2.2.0: timings.*
- [20] A. M. Ostrowski, *Solutions of equations and systems of equations*, Pure and Applied Mathematics, no. 9, Academic Press, New York, 1960. MR 23 #B571 Zbl 0115.11201



- [21] F. A. Potra, *On  $Q$ -order and  $R$ -order of convergence*, J. Optim. Theory Appl. **63** (1989), no. 3, 415–431. MR 91d:65077 Zbl 0663.65049
- [22] F. A. Potra and V. Pták, *Nondiscrete induction and iterative processes*, Research Notes in Mathematics, no. 103, Pitman, Boston, MA, 1984. MR 86i:65003 Zbl 0549.41001
- [23] S. M. Roberts and J. S. Shipman, *On the closed form solution of Troesch's problem*, J. Comput. Phys. **21** (1976), no. 3, 291–304. MR 54 #4122 Zbl 0334.65062
- [24] S. M. Shakhno, *On a Kurchatov's method of linear interpolation for solving nonlinear equations*, Proc. Appl. Math. Mech. **4** (2004), 650–651.
- [25] J. F. Traub, *Iterative methods for the solution of equations*, Prentice-Hall, Englewood Cliffs, NJ, 1964. MR 29 #6607 Zbl 0121.11204
- [26] B. A. Troesch, *A simple approach to a sensitive two-point boundary value problem*, J. Comput. Phys. **21** (1976), no. 3, 279–290. MR 54 #4121 Zbl 0334.65063

Received March 29, 2012. Revised October 15, 2012.

JOSÉ A. EZQUERRO: [jezquer@unirioja.es](mailto:jezquer@unirioja.es)

*Department of Mathematics and Computation, University of La Rioja, 26004 Logroño, Spain*

ANGELA GRAU: [angela.grau@upc.edu](mailto:angela.grau@upc.edu)

*Department of Applied Mathematics II, Technical University of Catalonia, 08034 Barcelona, Spain*

MIQUEL GRAU-SÁNCHEZ: [miquel.grau@upc.edu](mailto:miquel.grau@upc.edu)

*Department of Applied Mathematics II, Technical University of Catalonia, 08034 Barcelona, Spain*

MIGUEL A. HERNÁNDEZ-VERÓN: [mahernan@unirioja.es](mailto:mahernan@unirioja.es)

*Department of Mathematics and Computation, University of La Rioja, 26004 Logroño, Spain*



## Guidelines for Authors

Authors may submit manuscripts in PDF format on-line at the Submission page at [msp.org/camcos](http://msp.org/camcos).

**Originality.** Submission of a manuscript acknowledges that the manuscript is original and is not, in whole or in part, published or under consideration for publication elsewhere. It is understood also that the manuscript will not be submitted elsewhere while under consideration for publication in this journal.

**Language.** Articles in CAMCoS are usually in English, but articles written in other languages are welcome.

**Required items.** A brief abstract of about 150 words or less must be included. It should be self-contained and not make any reference to the bibliography. If the article is not in English, two versions of the abstract must be included, one in the language of the article and one in English. Also required are keywords and subject classifications for the article, and, for each author, postal address, affiliation (if appropriate), and email address.

**Format.** Authors are encouraged to use L<sup>A</sup>T<sub>E</sub>X but submissions in other varieties of T<sub>E</sub>X, and exceptionally in other formats, are acceptable. Initial uploads should be in PDF format; after the refereeing process we will ask you to submit all source material.

**References.** Bibliographical references should be complete, including article titles and page ranges. All references in the bibliography should be cited in the text. The use of BibT<sub>E</sub>X is preferred but not required. Tags will be converted to the house format, however, for submission you may use the format of your choice. Links will be provided to all literature with known web locations and authors are encouraged to provide their own links in addition to those supplied in the editorial process.

**Figures.** Figures must be of publication quality. After acceptance, you will need to submit the original source files in vector graphics format for all diagrams in your manuscript: vector EPS or vector PDF files are the most useful.

Most drawing and graphing packages (Mathematica, Adobe Illustrator, Corel Draw, MATLAB, etc.) allow the user to save files in one of these formats. Make sure that what you are saving is vector graphics and not a bitmap. If you need help, please write to [graphics@msp.org](mailto:graphics@msp.org) with details about how your graphics were generated.

**White space.** Forced line breaks or page breaks should not be inserted in the document. There is no point in your trying to optimize line and page breaks in the original manuscript. The manuscript will be reformatted to use the journal's preferred fonts and layout.

**Proofs.** Page proofs will be made available to authors (or to the designated corresponding author) at a Web site in PDF format. Failure to acknowledge the receipt of proofs or to return corrections within the requested deadline may cause publication to be postponed.

# *Communications in Applied Mathematics and Computational Science*

vol. 9

no. 2

2014

---

A comparison of high-order explicit Runge–Kutta, extrapolation, and  
deferred correction methods in serial and parallel 175

DAVID I. KETCHESON and UMAIR BIN WAHEED

A new class of secant-like methods for solving nonlinear systems of equations 201

JOSÉ A. EZQUERRO, ANGELA GRAU, MIQUEL GRAU-SÁNCHEZ and  
MIGUEL A. HERNÁNDEZ-VERÓN



1559-3940(2014)9:2;1-#