GRID-INDEPENDENT CONSTRUCTION OF MULTISTEP METHODS

Arévalo, Carmen; Söderlind, Gustaf

Published in:
Journal of Computational Mathematics

DOI:
10.4208/jcm.1611-m2015-0404

2017

Document Version:
Publisher's PDF, also known as Version of record

Journal of Computational Mathematics
Vol.35, No.5, 2017, 672–692.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1611-m2015-0404

GRID-INDEPENDENT CONSTRUCTION OF MULTISTEP METHODS*

Carmen Arévalo and Gustaf Söderlind
Numerical Analysis, Centre for Mathematical Sciences, Box 118, SE-221 00 Lund, Sweden.
Email: Carmen.Arevalo@na.lu.se, Gustaf.Soderlind@na.lu.se

Abstract

A new polynomial formulation of variable step size linear multistep methods is presented, where each k-step method is characterized by a fixed set of $k-1$ or k parameters. This construction includes all methods of maximal order ($p=k$ for stiff, and $p=k+1$ for nonstiff problems). Supporting time step adaptivity by construction, the new formulation is not based on extending classical fixed step size methods; instead classical methods are obtained as fixed step size restrictions within a unified framework. The methods are implemented in MATLAB, with local error estimation and a wide range of step size controllers. This provides a platform for investigating and comparing different multistep method in realistic operational conditions. Computational experiments show that the new multistep method construction and implementation compares favorably to existing software, although variable order has not yet been included.

Mathematics subject classification: 65L06, 65L05, 65L80
Key words: Linear multistep methods, Variable step size, Adaptive step size, Step size control, Explicit methods, Implicit methods, Nonstiff methods, Stiff methods, Initial value problems, Ordinary differential equations, Differential-algebraic equations, Implementation.

1. Introduction

Linear multistep methods for solving ordinary differential equations

$$\dot{y} = f(t,y), \qquad y(t_0) = 0, \qquad t \in [t_0, t_f], \tag{1.1}$$

consist of a discretization formula and a pointwise representation of the differential equation,

$$\sum_{i=0}^{k} \alpha_{k-i} x_{n-i} = h \sum_{i=0}^{k} \beta_{k-i} x'_{n-i} \tag{1.2}$$

$$x'_{n-i} = f(t_{n-i}, x_{n-i}). \tag{1.3}$$

Here x_{n-i} approximates $y(t_{n-i})$, k is the step number, and the step size is assumed to be constant, $h = t_n - t_{n-1}$. There is a well established theory for such methods, covering all essential aspects such as order of convergence and stability, [9, 10]. Other classes of problems, such as differential-algebraic equations of the form $F(t,y,\dot{y}) = 0$, can, at least in principle, be treated in a similar manner, by replacing (1.3) by $F(t_{n-i}, x_{n-i}, x'_{n-i}) = 0$. Here, a dot denotes the *time derivative of a function* as in (1.1), while a prime denotes a *sample of the vector*

* Received September 24, 2015 / Revised version received July 27, 2016 / Accepted November 21, 2016 / Published online July 1, 2017 /

field that defines the ODE. This distinction is motivated by the fact that in the computational process, the vector field samples are not located on a single trajectory, whence a dot notation would be misleading in (1.2 – 1.3).

In practice, a multistep method must be adaptive and use *variable step size*. There are several well-known and efficient implementations. Most of these are based on predictor-corrector schemes (Adams-Bashforth, Adams–Moulton) for nonstiff problems, or on backward differentiation formulas (BDF) for stiff problems, e.g. the codes VODE [5] and LSODE [12]. All of these use variable step size as well as variable order.

An established methodology for the construction of variable step size multistep methods, other than extending classical constant step-size formulas case by case, is still missing, [14]. One of the more general approaches is the one by Nordsieck [11], who developed a theory showing the equivalence of k-step methods of higher order to polynomials defined by a vector of dimension k. Although this approach also identifies a set of parameters with each multistep method, these parameters vary with the step sizes. Among other notable extensions we find fixed leading coefficient implementations and divided difference implementations. Some have better stability and computational properties than others, but the theoretical understanding of variable step size multistep methods remains incomplete.

This is a well recognized problem, and different approaches are taken depending on particular preferences. In recent years, there has been an increasing interest in solving this problem in various, demanding special contexts. For example, in [21] the special needs of PDE's are taken into account, when explicit and implicit methods are mixed in a splitting scheme. The approach is to identify an additive decomposition, distinguishing a nonstiff and a stiff part, to be treated with dedicated methods, while incorporating time step adaptivity. The multistep methods are extended to variable step size on a case-by-case basis, deriving step size ratio dependent method coefficients, although the order remains fairly low.

In [8], the special requirements of strong stability preserving schemes are considered. In nonlinear conservation laws in PDE's it is essential that the scheme does not add numerical energy errors; for this reason it is important to maintain interpolation conditions when the step size varies, and methods are again extended to variable step size on a case-by-case basis. This is a tedious approach, and the order is severely restricted. Finally in [6], a general attempt is made to extend multistep methods to variable step size using exponential methods, a technique that, just like splitting methods, has received new attention in recent years.

These examples are by no means exhaustive, but they point to the diversity of difficulties that one encounters when multistep methods are the preferred integration methods and adaptivity is crucial.

The objective of this paper is to develop a new, general methodology for variable step size multistep methods. Our approach addresses several well known problems and opens a new avenue of research in adaptive multistep methods. The main idea is to construct multistep methods that approximate the solution to the ODE by a polynomial, and where specific methods are characterized in terms of a fixed set of interpolation and collocation conditions. This will cover all k-step methods of orders $p = k$ and $p = k + 1$; these methods are of maximal order for stiff and nonstiff problems, respectively.

Our approach is not based on extending classical methods; on the contrary, classical methods are obtained as fixed step size restrictions within a general interpolation representation, which, for each method, can be characterized in terms of a set of *fixed parameters*, even in the presence of varying step size. Further, the approach provides a continuous extension of

the solution, making the methods better suited to practical needs, as intermediate output is directly generated within the polynomial representation of each method.

With this approach, we establish both a methodological and a software platform for investigating every high order multistep method in a general purpose context, both with respect to method properties and computational performance. The approach does not include exponential methods and splitting. However, it does include some of the special splitting methods used in differential-algebraic equations, and utilizes advanced step size control, [16, 19] to enhance computational stability, [18].

Disregarding weakly stable methods, higher order k-step methods fall in three different categories:

- **Type $\mathbf{E}_k$** Explicit methods of order $p = k$ e.g. Adams–Bashforth
- **Type $\mathbf{I}_k$** Implicit methods of order $p = k$ e.g. BDF methods
- **Type $\mathbf{I}_k^+$** Implicit methods of order $p = k + 1$ e.g. Adams–Moulton

An implicit k-step formula has $2k+1$ coefficients, and an explicit formula ($\beta_k = 0$ in (1.2)) has $2k$ coefficients. Methods of order p will have a polynomial of degree p. As there are $p+1$ order conditions to be satisfied, there will be $k-1$ degrees of freedom for methods of Types E_k and I_k^+, while methods of Type I_k have k degrees of freedom. These will be determined by parameter sets that uniquely specify the method. The new representation has been implemented in MATLAB codes, one for each category stated above. Several ODE problems are then solved as a proof of concept, and to demonstrate the feasibility of the approach. Results are also compared to those obtained from standard solvers.

Apart from the classical methods mentioned above, there is a wide range of alternatives. In [3,4] we developed the β-blocking approach to solving differential-algebraic equations (DAEs) using various combinations of multistep methods. New, dedicated families of non-stiff implicit and explicit methods were derived in this context, in particular the implicit and explicit difference correction methods (IDC and EDC, respectively), [4]. All are of maximal order and include, as special cases, the Adams and BDF methods as well as the difference corrected BDFs (dcBDF, [15]). While the classical methods have been implemented for variable step size, to this date no general technique for implementing the IDC and EDC families has been developed; the methods were originally constructed using the classical approach with fixed step size difference operator series expansions, [4].

2. Basic Definitions and Polynomial Representation

For simplicity of notation, we only describe the formalism for a scalar equation (1.1). The extension to vector systems is technically straightforward, but as the required additional notation tends to obscure the formalism without adding crucial insight, we prefer a simpler exposition. The code, however, implements the formalism for systems of first order ODE's.

Let Π_p denote the space of polynomials of degree p. In the sequel, we define a multistep formula as a linear combination of state values, x_n together with their corresponding vector field samples, $x'_n = f(t_n, x_n)$. We assume that the sequence $\{t_n\}_{n=0}^{\infty}$ is strictly increasing, with step sizes $h_{n-1} = t_n - t_{n-1} > 0$. Further, we assume that $\{x_n\}_{n=0}^{\infty}$ represents some approximation to a sequence $\{y(t_n)\}_{n=0}^{\infty}$, sampled from the solution to the initial value problem $\dot{y} = f(t, y)$ with $y(t_0) = x_0$.

We construct k-step methods as *polynomial methods*. These approximate the solution $y(t)$ on each subinterval $[t_{n-1}, t_n]$ by a polynomial P_n, generating both the discrete solution $x_n = P_n(t_n)$

at t_n, as well as a continuous extension; $P_n(t)$ can be evaluated at any point in $[t_{n-1}, t_n]$, and $P_{n-1}(t)$ can be evaluated at t_n in order to produce an error estimate.

This allows us to implement any explicit or implicit higher order multistep formula with variable step-size. Ideally, P_n would interpolate $\{x_{n-j}\}_{j=0}^k$ while $\dot{P}_n$ interpolates $\{x'_{n-j}\}_{j=0}^k$. As this is known to lead to unstable methods we restrict the order, and assume that $P_n \in \Pi_p$, with $k \le p \le k+1$. Hence we cannot interpolate at all points, leaving a "slack" at a few of them.

Definition 2.1. *Let the sequences* $\{x_{n-j}\}_{j=0}^k$ *and* $\{x'_{n-j}\}_{j=0}^k$ *be given for a fixed* n*. Further, let* $P_n \in \Pi_p$ *with* $k \le p \le k+1$*. The* state slack s_{n-j} *and* derivative slack s'_{n-j} *at* t_{n-j} *are defined by*

$$s_{n-j} = P_n(t_{n-j}) - x_{n-j}\,, \qquad s'_{n-j} = \dot{P}_n(t_{n-j}) - x'_{n-j}\,; \qquad j = 0:k. \tag{2.1}$$

We are now ready to define parametric multistep polynomial methods. Noting that there are three structurally different types of high order methods, the methods will be characterized by a *structural condition* that uniquely identifies the method type, and additional *parametric conditions*. These are all expressed in terms of the state and derivative slacks.

Because there are one-step methods of all three different types, the structural condition must necessarily be composed of the state slack s_{n-1} and the two derivative slacks s'_{n-1} and s'_n,usually in terms of interpolation and collocation conditions. The parametric condition, on the other hand, is concerned with how a prototypical one-step method extends to various families of methods.

For example, among one-step methods, there is a unique Type E_k method – the explicit Euler method. Within the polynomial context, it is characterized by a first degree polynomial P_n, satisfying the *interpolation condition* $s_{n-1} = 0$, and the *explicit collocation condition* $s'_{n-1} = 0$.

Similarly, there is a unique one-step Type I_k^+ method – the trapezoidal rule. It is characterized by a second degree polynomial P_n, satisfying $s_{n-1} = 0$ and $s'_{n-1} = 0$, together with the *implicit collocation condition* $s'_n = 0$, also written $\dot{P}_n(t_n) = f(t_n, P_n(t_n))$. These three conditions uniquely determine the polynomial's three coefficients.

Implicit one-step methods of order 1 (Type I_k) are not unique, however; there is a one-parameter family of such methods known as *theta methods*. Here, a first degree polynomial is characterized by the implicit collocation condition $\dot{P}_n(t_n) = f(t_n, P_n(t_n))$, together with a *slack balance condition*, expressed as the linear combination

$$s_{n-1}\cos\theta_0 + h s'_{n-1}\sin\theta_0 = 0 \quad \text{for} \quad \theta_0 \in (-\pi, \pi].$$

This is obviously not the usual parametrization of the theta methods, but it conforms to the parametrization we use for k-step methods below. The implicit Euler method, which is the prototypical Type I_k method, is obtained when the slack balance condition degenerates to the interpolation condition $s_{n-1} = 0$, i.e., for $\theta_0 = 0$.

Interestingly, k-step methods of all three types are obtained by augmenting the one-step conditions above by the same set of $k-1$ slack balance conditions, of the form

$$s_{n-j-1}\cos\theta_j + h_{n-j-1} s'_{n-j-1}\sin\theta_j = 0; \quad \theta_j \in (-\pi/2, \pi/2], \text{ for } j = 1:k-1. \tag{2.2}$$

Thus the parameter set $\Theta = \{\theta_j\}_{j=0}^{k-1}$ characterizes k-step methods, where the parameter θ_0 is only used in Type I_k methods and is otherwise void. We summarize in a definition.

Definition 2.2. *A k-step* parametric multistep method *of order p (with $k \le p \le k+1$) is defined by a polynomial P_n, satisfying $k-1$* slack balance conditions (2.2) *together with the following* structural conditions*:*

***Type E_k** (Explicit methods of order $p = k$) $P_n \in \Pi_k$ satisfies an interpolation condition and an explicit collocation condition,*

$$s_{n-1} = 0, \tag{2.3}$$
$$s'_{n-1} = 0. \tag{2.4}$$

***Type I_k^+** (Implicit methods of order $p = k+1$) $P_n \in \Pi_{k+1}$ satisfies an interpolation condition, and explicit as well as implicit collocation conditions,*

$$s_{n-1} = 0, \tag{2.5}$$
$$s'_{n-1} = 0, \tag{2.6}$$
$$\dot{P}_n(t_n) = f(t_n, P_n(t_n)). \tag{2.7}$$

***Type I_k** (Implicit methods of order $p = k$) $P_n \in \Pi_k$ satisfies a slack balance condition, as well as the implicit collocation condition,*

$$s_{n-1}\cos\theta_0 + h_{n-1}s'_{n-1}\sin\theta_0 = 0; \qquad \theta_0 \in (-\pi/2, \pi/2], \tag{2.8}$$
$$\dot{P}_n(t_n) = f(t_n, P_n(t_n)). \tag{2.9}$$

A method is implicit if and only if the implicit collocation condition (2.7) or (2.9) is included. The approximation at the next time point, t_n, is in all cases defined by $x_n = P_n(t_n)$. In the explicit case, x_n can be evaluated directly once the polynomial P_n has been constructed from "past data". Otherwise, the construction of P_n involves solving a nonlinear equation dependent on the vector field f, as indicated by (2.7) and (2.9).

In [2] it was shown that all *implicit multistep methods of maximal convergence order* are characterized by a multistep polynomial $P_n \in \Pi_{k+1}$, whose $k+2$ coefficients are determined by collocation and interpolation conditions. This approach led to a less advantageous parameter representation and did not allow for the inclusion of other types of maximal order methods. Here we will show that *all linear multistep methods of practical interest can be represented by a multistep polynomial method* in the manner devised above. In other words, the construction of parametric multistep methods is completely defined in terms of slack balance conditions specified by a parameter set $\{\theta_j\}_{j=1}^k$ that spans the entire space of linear multistep methods.

3. Parametric Formulation

We will show that all three types of methods mentioned above satisfy the required consistency order conditions. We will then discuss basic properties of some particular I_k^+ methods in the light of this formulation. Because of the slack balance conditions that each polynomial must satisfy, its coefficients depend on the last *step size ratios*, defined as

$$\omega_{n-1} = \frac{h_n}{h_{n-1}}.$$

Theorem 3.1. *Every k-step method with $P_n \in \Pi_{k+1}$ defined by the conditions*

$$I_k^+ : \begin{cases} s'_n = 0, \\ s_{n-1} = 0, \;\; s'_{n-1} = 0, \\ s_{n-j}\cos\theta_{j-1} + h_{n-j}s'_{n-j}\sin\theta_{j-1} = 0; \quad j = 2:k \end{cases}$$

is implicit and has order of consistency $p = k + 1$.

Proof. Implicitness follows from the implicit collocation condition. Let

$$P_n(t) = c_{k+1}(t - t_{n-1})^{k+1} + \cdots + c_1(t - t_{n-1}) + c_0.$$

The explicit structural conditions imply that $c_1 = x'_{n-1}$, $c_0 = x_{n-1}$, and the coefficients $c_{k+1}, c_k, \ldots, c_2$ are the solution of the system $Mc = g$ where M is a $k \times k$ matrix $\{m_{ij}\}_{i,j=0}^{k}$ defined as follows, in terms of the last $k - 1$ step-size ratios:

$$\begin{aligned} m_{1j} &= (k - j + 2)h_{n-1}^{k-j+2} && (j = 1:k) \\ m_{ij} &= (-1)^{k-j}\, h_{n-1}^{k-j+2} \left(\sum_{l=0}^{i-2} \frac{1}{\Pi_{d=n-l-2}^{n-2}\omega_d} \right)^{k-j+1} \\ &\quad \cdot \left(\cos\theta_{i-1} \left(\sum_{l=0}^{i-2} \frac{1}{\Pi_{d=n-l-2}^{n-2}\omega_d} \right) - \sin\theta_{i-1} \frac{k-j+2}{\Pi_{d=n-i}^{n-2}\omega_d} \right) \\ &&& (i \neq 1, j = 1:k) \end{aligned} \tag{3.1}$$

and g is the $k \times 1$ vector with components

$$\begin{aligned} g_1 &= -h_{n-1}x'_{n-1} \\ g_i &= \cos\theta_{i-1} \left(-x_{n-1} + x_{n-i} - \left(\sum_{l=0}^{i-2} \frac{1}{\Pi_{d=n-l-2}^{n-2}\omega_d} \right) h_{n-1}x'_{n-1} \right) \\ &\quad + \sin\theta_{i-1} \left(\frac{1}{\Pi_{d=n-i}^{n-2}\omega_d} h_{n-1}(-x'_{n-1} + x'_{n-i}) \right) \qquad (i = 2:k). \end{aligned}$$

Suppose the solution of the differential equation is a polynomial, $Q \in \Pi_{k+1}$, such that $x_{n-i} = Q(t_{n-i}), (i = 1:k)$ and $x'_{n-i} = \dot{Q}(t_{n-1}), (i = 0:k)$. This corresponds to the standard approach of inserting "exact data." We need to show that the formulation yields $P_n(t_n) = Q(t_n)$. Define

$$\Lambda(t) = P_n(t) - Q(t).$$

Then $\Lambda \in \Pi_{k+1}$ satisfies the following conditions:

$$\dot{\Lambda}(t_n) = 0 \tag{3.2}$$

$$\Lambda(t_{n-1}) = 0 \tag{3.3}$$

$$\dot{\Lambda}(t_{n-1}) = 0 \tag{3.4}$$

$$\cos\theta_{j-1}\Lambda(t_{n-j}) + \sin\theta_{j-1}h_{n-j}\dot{\Lambda}(t_{n-j}) = 0; \quad j = 2:k. \tag{3.5}$$

The system matrix is identical to M, the matrix for the system that defined the multistep polynomial P_n. Then the system (3.2)–(3.5) also has a unique solution, viz., $\Lambda \equiv 0$. Hence $P_n(t_n) = Q(t_n)$. □

We state the following theorems without proof as all proofs follow a pattern similar to that of Theorem 3.1. In the following two cases, the polynomial $Q(t)$ is of degree k.

Theorem 3.2. *Every k-step method with polynomials $P_n \in \Pi_k$ defined by*

$$I_k\colon \begin{cases} s'_n = 0 \\ s_{n-j}\cos\theta_{j-1} + h_{n-j}s'_{n-j}\sin\theta_{j-1} = 0; \quad j = 1:k. \end{cases}$$

is implicit and has order of consistency $p = k$.

Theorem 3.3. *Every k-step method with polynomials $P_n \in \Pi_k$ defined by*

$$E_k\colon \begin{cases} s_{n-1} = 0, \;\; s'_{n-1} = 0, \\ s_{n-j}\cos\theta_{j-1} + h_{n-j}s'_{n-j}\sin\theta_{j-1} = 0; \quad j = 2:k \end{cases}$$

is explicit and has order of consistency $p = k$.

The parametric multistep formulation thus allows the construction of methods of a given order of consistency, regardless of stability. Only those methods satisfying appropriate stability (hence convergence) conditions are of significance. To illustrate the new parametric formulation of multistep methods, we will take implicit I_k^+ methods of step numbers $k = 2$ and $k = 3$, with corresponding orders of consistency $p = 3$ and $p = 4$. We will focus on the analysis of basic method properties, such as zero-stability, order of convergence, error constant and stability region.

3.1. Properties of two-step methods of order 3

Disregarding weakly stable methods, such as the two-step Milne method, the maximum convergence order attained by two-step implicit methods is $p = 3$. The multistep polynomial must be of degree 3 and have four coefficients. Such a method employs two step sizes, h_{n-1} and h_{n-2}, which potentially pose a stability restriction on the step size ratio, ω_{n-2}.

An I_2^+ method has multistep polynomials $P_n \in \Pi_3$ defined by

$$s'_n = 0, \tag{3.6a}$$
$$s_{n-1} = 0, \;\; s'_{n-1} = 0, \tag{3.6b}$$
$$s_{n-2}\,\cos\theta + h_{n-2}\,s'_{n-2}\,\sin\theta = 0. \tag{3.6c}$$

Let f_j represent $f(t_j, x_j)$. From (3.6b), we can write

$$P_n(t) = a(t - t_{n-1})^3 + b(t - t_{n-1})^2 + f_{n-1}(t - t_{n-1}) + x_{n-1} \tag{3.7}$$

and from the remaining conditions we get two nonlinear equations,

$$3h_{n-1}^3 a + 2h_{n-1}^2 b = h_{n-1}(f_n - f_{n-1}), \tag{3.8a}$$
$$\begin{aligned}(-c + 3s)h_{n-2}^3 a + (c - 2s)h_{n-2}^2 b \\ = c(-x_{n-1} + x_{n-2}) + h_{n-2}[(c - s)f_{n-1} + sf_{n-2}],\end{aligned} \tag{3.8b}$$

where $c = \cos\theta$, $s = \sin\theta$ and $x_n = ah_{n-1}^3 + bh_{n-1}^2 + f_{n-1}h_{n-1} + x_{n-1}$. The Jacobian of this system is regular for

$$\omega_{n-2} \neq -\frac{2(3\sin\theta - \cos\theta)}{(3 - h_{n-1}f_x)(\cos\theta - 2\sin\theta)}, \tag{3.9}$$

where $f_x = \partial f/\partial x$ is evaluated at (t_n, x_n). Recall that our derivations above are carried out for scalar systems. The operator $(3 - h_{n-1}f_x)^{-1}$ corresponds to the Newton iteration matrix that would be used in tandem with the implicit method. For small enough values of h_{n-1}, that operator is positive. The θ-dependent factor, however, is non-positive when $\theta \in [-\frac{\pi}{2}, \arctan\frac{1}{3}) \cup (\arctan\frac{1}{2}, \frac{\pi}{2}]$, and as the step size ratios ω_n are positive, the Jacobian remains regular in this interval. Therefore, with the exception of the 4th order Milne method with $\theta = \arctan(1/3)$, all 2-step methods defined by (3.6) with

$$\theta \in (-\frac{\pi}{2}, \arctan\frac{1}{3}) \cup [\arctan\frac{1}{2}, \frac{\pi}{2}]$$

are implicit and of consistency order $p = 3$, while those with

$$\theta \in (\arctan\frac{1}{3}, \arctan\frac{5}{12}) \cup (\arctan\frac{5}{12}, \arctan\frac{1}{2})$$

have consistency order $p = 3$ as long as the step size ratio satisfies

$$\omega_{n-2} \neq -\frac{2(3\sin\theta - \cos\theta)}{(3 - h_{n-1}f_x)(\cos\theta - 2\sin\theta)}. \tag{3.10}$$

The two two-step IDC methods, Adams-Moulton and dcBDF, are obtained for $\theta = \pi/2$ and $\theta = \arctan 2/3$, respectively. Our variable step size Adams-Moulton formulation coincides with the extension given in [9]. For $\theta = \arctan 1/3$ we obtain Milne's method, a weakly stable 4th order method. However, as that method has order $p = k + 2$, and we only employ a polynomial $P_n \in \Pi_3$, the 4th order has a "superconvergence" character only achieved locally, in the immediate vicinity of the new point x_n, while the 3rd degree polynomial does not offer a continuous extension of order $p = 4$. Finally, with $\theta = 0$ we obtain a non-stiff method with a stability region about 2/3 the size of that of the corresponding Adams-Moulton method.

For constant step size ($\omega_n = 1$), it is straightforward to determine zero stability. The bridge from consistency order to convergence order is a complicated issue, however, when variable step sizes are used. Applying the variable step size k-step method to the problem $\dot{y} = 0$, stability is defined in [9, p. 403], in terms of the resulting time-stepping matrix A_n. This leads to a recursion

$$X_{n+1} = A_n X_n, \tag{3.11}$$

with $X_n = (x_{n+k-1}, \ldots, x_n)^T$. Some operator norm of $A_{n+l}A_{n+l-1}\cdots A_n$ must be uniformly bounded for all $l \geq 0$ and all n. As this matrix contains the step size dependent coefficients $\alpha_{k-i,n}$ of the form

$$\sum_{i=0}^{k} \alpha_{k-i,n} x_{n-i} = 0, \tag{3.12}$$

and these coefficients depend continuously on the step size ratios $\omega_{n-2}, \ldots, \omega_{n-k}$, a method will also be stable for variable steps in some step size ratio neighborhood

$$\omega_\nu \in (1 - \psi, 1 + \Psi)$$

with $0 < \psi < 1$ and $\Psi > 0$, provided the method is zero-stable for $\omega_{n-2} = \cdots = \omega_{n-k} = 1$.

For zero-stable methods, errors in the starting values do not grow without bound. In constant step size theory, this requires that the zeros of $\rho(\zeta)$ satisfy the root condition. Several

attempts have been made to determine similar conditions for fixed step size ratios $h_{n-1}/h_{n-2} = \omega > 1$. This corresponds to ramping up the step size at a constant exponential rate. While such a situation does not occur in practical computations, except at a finite number of steps, it does justify the attempts of finding the maximum stable value of ω. This procedure can also be used for two-step implicit order 3 methods, where the stability condition requires the existence of a number $0 < q < 1$ such that

$$\left| \frac{-\omega^3 \cos\theta}{(3\cos\theta - 6\sin\theta)\omega + 2\cos\theta - 6\sin\theta} \right| \le q. \tag{3.13}$$

As the parameter for the two-step third order Adams-Moulton method is $\theta = \pi/2$, this condition holds for any ω. Hence there is no need to restrict the step size ratios for this method. For all other methods, condition (3.13) reduces to

$$\frac{\omega^3}{|(3 - 6\tan\theta)\omega + 2 - 6\tan\theta|} \le q. \tag{3.14}$$

The two-step third order dcBDF method is defined by $\tan\theta = 2/3$. Therefore, the condition becomes

$$\frac{\omega^3}{\omega + 2} \le q < 1.$$

This method will be stable for $\omega \in (0,\ 1.5)$. On the other hand, Milne's method, with $\tan\theta = 1/3$, must satisfy the stability condition $\omega^3 - q\omega \le 0$, but there is no $q < 1$ that satisfies the condition when $\omega = 1$. Finally, for the I_2^+ method with parameter $\theta = \pi/10$, the stability condition is $\omega^3 - 1.0505q\omega + 0.0505q \le 0$. This implies that the method is guaranteed to be stable if θ is kept in $(0,\ 1.048]$. In Figure 3.1 we generalize this result, by plotting the maximum ω for which two-step 3rd order methods are stable, for every feasible value of the parameter θ.

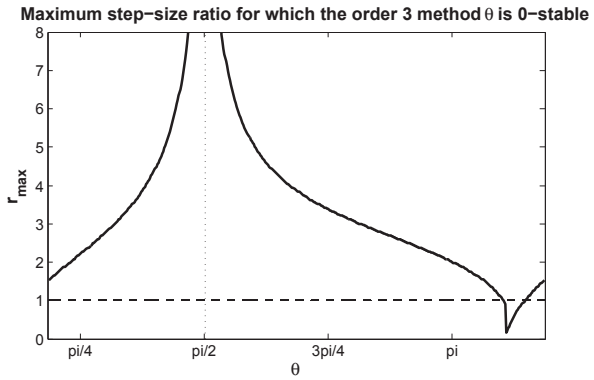

Fig. 3.1. *Zero stability.* Only Adams-Moulton ($\theta = \pi/2$) is 0-stable for any step-size ratio. Methods with $\theta \in [\arctan 1/3, \arctan 1/2]$ are unstable as they are not 0-stable for $\omega = 1$.

Thus step size ratios must be restricted, and the step size should vary smoothly, see also [1]. This can be accomplished by well designed step size controllers (see below). These do not keep the step size piecewise constant, only allowing the occasional substantial change (usually by a factor of 2). Instead, the step size changes continually, even when ω_n deviates from 1 by very small amounts. This increases the the stability of the method, without harming its efficiency; in fact, we often use fewer steps than standard codes of the same order.

If the step size ratio is restricted to the interval $[0.8, 1.2]$, we can cover parametric methods with $\theta \in (-\frac{\pi}{2}, \arctan(0.2934)] \cup [\arctan(0.5553), \frac{\pi}{2}]$. This is a perfectly useful range for practical computations.

The principal error term also varies with the step size ratio. Figure 3.2 shows the dependency of error coefficients on ω for three different methods. As shown in Figures 3.2 and 3.3, the stability region of the same methods are larger for larger error coefficients.

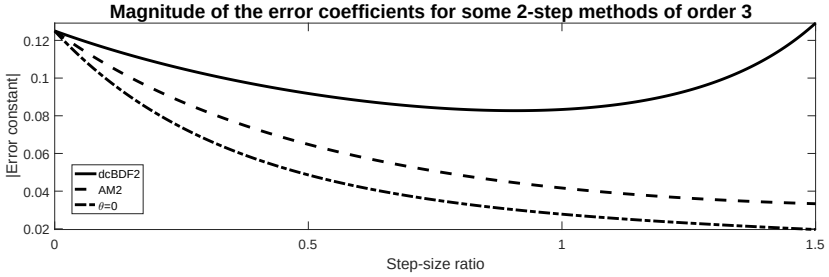

Fig. 3.2. *Error coefficients.* Error coefficients are shown for dcBDF2 ($\theta = \arctan 2/3$), Adams-Moulton2 ($\theta = \pi/2$), and the method with $\theta = 0$. dcBDF2 is 0-stable for $\omega \leq 1.52$.

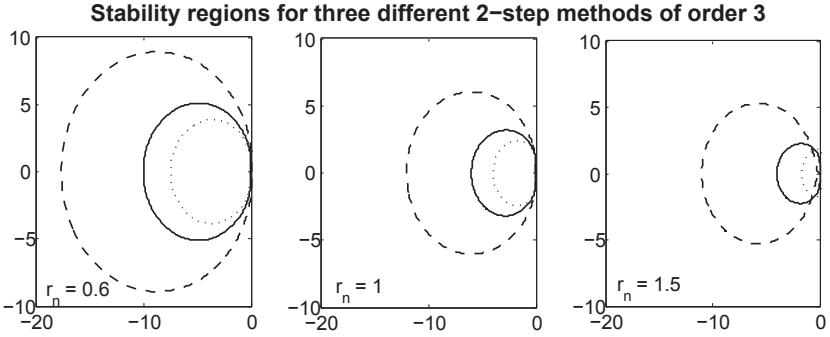

Fig. 3.3. *Stability regions.* Stability regions for dcBDF (solid), Adams-Moulton (dashed) and the method with $\theta = 0$ (dotted). Larger stability regions correspond to larger error coefficients. The dcBDF2 method is zero-stable for $\omega \leq 1.52$.

3.2. Three-step methods of order 4

As the complexity of the multistep method increases, so does the number of parameters needed to define its collocation polynomial. The structure of the interpolation conditions will remain the same: k-step implicit methods of order $p = k+1$ will require $k-1$ slack conditions.

For instance, the fourth degree collocation polynomial for I_3^+ is defined by the following conditions:

$$
\begin{aligned}
& s'_n = 0, \\
& s_{n-1} = 0, \quad s'_{n-1} = 0, \\
& s_{n-2} \cos\theta_1 + h_{n-2}\, s'_{n-2} \sin\theta_1 = 0, \\
& s_{n-3} \cos\theta_2 + h_{n-3}\, s'_{n-3} \sin\theta_2 = 0.
\end{aligned}
$$

The coefficients of the methods will depend on the last two step-size ratios, $\omega_{n-3} = h_{n-2}/h_{n-3}$ and $\omega_{n-2} = h_{n-1}/h_{n-2}$.

The domain of the parameters will be a subset of $(-\pi/2, \pi/2] \times (-\pi/2, \pi/2]$. For example, $\theta_1 = \arctan(7/6)$ and $\theta_2 = \pi/2$ gives IDC23, a method that can be used in conjunction with β-blocking for solving index 2 DAEs. From the method's representation one can work out a step size ratio condition for constant ramp-up. The stability condition for this method depends on the boundedness of products of matrices of the form

$$A_n^* = \begin{bmatrix} -\dfrac{\omega_{n-2}^3(\omega_{n-2}\omega_{n-3} + 2\omega_{n-3} + 2)}{2\omega_{n-2}\omega_{n-3} - 8\omega_{n-2} + \omega_{n-3} - 10} & 0 \\ 1 & 0 \end{bmatrix}.$$

A sufficient condition for strong stability is that the step ratios are restricted by

$$\frac{\omega_{n-2}^3(\omega_{n-2}\omega_{n-3} + 2\omega_{n-3} + 2)}{|2\omega_{n-2}\omega_{n-3} - 8\omega_{n-2} + \omega_{n-3} - 10|} < 1.$$

If, e.g., during the initial phase of the integration, the step size needs to be ramped up quickly, one can find the maximum permissible step size increase by investigating for what value of

Table 3.1: Parameters of selected high order methods.

Method	**Order**	I_k **method parameters** $\tan(\theta_j)$, $j = 0 : k-1$				
BDFk	$k \le 6$	$\{0\}_{0:k-1}$				
Kregel [7]	3	154/543	-11/78	0		
Rockswold [13]	3	1/3	2/3	1		
Method	**Order**	I_k^+ **method parameters** $\tan(\theta_j)$, $j = 1 : k-1$				
AMk	$k+1$	$\{\infty\}_{1:k-1}$				
dcBDFk	$k+1$	$\{(j+1)/(k+1)\}_{1:k-1}$				
Milne2	4	1/3				
Milne4	5	4/15	∞	∞		
IDC23	4	7/6	∞			
IDC24	5	26/15	∞	∞		
IDC34	5	4/5	33/20	∞		
IDC45	6	28/45	11/10	32/15	∞	
IDC56	7	43/84	6/7	29/21	55/21	∞
Method	**Order**	E_k **method parameters** $\tan(\theta_j)$, $j = 1 : k-1$				
ABk	k	$\{\infty\}_{1:k-1}$				
EDFk	k	$\{j+1\}_{1:k-1}$				
Nyström3	3	-2/3	∞			
Nyström4	4	-5/3	∞	∞		
Nyström5	5	-133/45	∞	∞	∞	
EDC22	3	14/3	∞			
EDC23	4	49/6	∞	∞		
EDC33	4	7/2	39/4	∞		
EDC24	5	1121/90	∞	∞	∞	
EDC34	5	53/10	219/10	∞	∞	
EDC45	6	193/45	121/10	692/15	∞	∞

$\omega = \omega_{n-2}\omega_{n-3}$ this condition would hold. In the implementation of the methods, such stability conditions may be monitored to warn against potential stability problems when the step size varies.

Parameters of the most commonly used multistep methods are shown in Table 3.1, together with the parameters of some special purpose methods that may be used in regular and singular β-blocking, [4]. Parameters of methods originally defined by fixed step size formulas were obtained using MAPLE. The coefficients of the polynomial method were obtained symbolically and then equaled to the particular method coefficients; as there is a one-to-one correspondence, the values of the parameters θ_j can be constructed. In Table 3.1, they are given implicitly in terms of $\tan\theta_j$; actual θ_j values are trivially computed from the given rational coefficients, where we use $\arctan\infty = \pi/2$. In the table, Milne2 refers to the classical weakly stable two-step Milne method of order $p = 4$, while Milne4 refers to the four-step Milne–Simpson method of order $p = 5$.

4. Implementation Issues

Three similar Matlab codes were implemented, for methods of type E_k, I_k, and I_k^+. A few important features of these programs will be discussed below.

4.1. Basic algorithmic details

For the solution of the initial value problem (1.1), the user must provide the right hand-side function, f; the integration interval; the initial value; and the parameter vector θ that defines the method. All algorithms have default values for several parameters, but they can be overridden by the user, who has the option of defining the following:

- Size of the initial step size;
- Choice of error controller;
- Maximum and minimum values allowed for step size ratios;
- Choice of either absolute error control, or combined absolute/relative error control, implemented internally using the usual RTOL and ATOL approach;
- Choice of controlling *error per step* or *error per unit step*, to support both stiff solvers and tolerance proportionality.

These aspects will be discussed in detail below.

The implementation of implicit I_k^+ methods for non-stiff problems is done in a $\mathrm{P(EC)^2E}$ scheme, where the prediction (P) is made by evaluating the previous multistep polynomial at the new point,

$$x'_{n,\mathrm{est}} = f(t_n, P_{n-1}(t_n)).$$

Following the evaluation (E) of the vector field, a correction (C) polynomial P_n^{c} is calculated by solving the linear system resulting from Theorem 3.1 (see Section 3.1). This entails a straightforward generalization of the equation system (3.8a – 3.8b) to the k-step case. The final vector field evaluation (E) of the corrected derivative

$$x'_{n,\mathrm{c}} = f(t_n, P_n^{\mathrm{c}}(t_n))$$

is then used to construct the multistep polynomial P_n, which advances the solution by

$$x_n = P_n(t_n), \quad x'_n = f(t_n, x_n).$$

Implicit I_k methods for stiff problems are implemented using the equations of Theorem 3.2. These are non-linear equations that are solved by a simplified Newton iteration, in which the Jacobian is updated at each step but kept constant throughout the Newton iteration.

In the case of explicit E_k methods, the equations in Theorem 3.3 give a system of k linear equations which must be solved at every step. The multistep polynomial is obtained and evaluated at t_n to advance the solution.

In all cases, the error estimate is constructed by comparing $P_n(t_n)$ to $P_{n-1}(t_n)$. This measures how much the method's polynomial changes over a single step and essentially corresponds to how different the solution would have been, had a polynomial of higher degree been used. This is akin to using an "embedded" pair of methods.

4.2. Initial step size

The choice of an appropriate initial step size is of great importance to fully exploit the potential of a high order method. It is particularly important that the initial step size does not generate a too large error, as it may be impossible to recover lost accuracy. It is therefore better to employ a somewhat conservative initial step size, as the step size controller will quickly ramp up the step size to meet the accuracy requirement.

Many existing codes are "self-starting" in the sense that they start with a short step size and a one-step method, after which they successively increase the order of the method. An alternative to this approach is to start directly with a high order method; this requires the generation of a sufficient number of starting values for the method. In both cases we need a procedure for determining a suitable initial step-size, whether it is used as a starting step size for a one-step method, or to construct the starting values of a multistep formula.

We generate the initial step size by the following algorithm.

1. Given the initial value $x_0 = x(t_0)$ we compute the derivative $f(t_0, x_0)$. We then use a small perturbation Δx of the initial value (usually selected randomly in the first orthant) to compute a perturbed derivative $f(t_0, x_0 + \Delta x)$ and obtain a crude estimate L_0 of the Lipschitz constant of f, as

$$L_0 = \frac{\|f(t_0, x_0 + \Delta x) - f(t_0, x_0)\|}{\|\Delta x\|}.$$

2. We next determine a preliminary initial step size Δt by the condition $\Delta t\, L_0 = 0.1$. The somewhat conservative choice of the number 0.1 reflects that L_0 underestimates the Lipschitz constant, and that we need to find a preliminary step size of such a scale that an explicit method is stable for the choice Δt.

3. We then use Δt to take a single forward Euler step, $x_1 = x_0 + \Delta t\, f(t_0, x_0)$. Next, we evaluate the vector field at this new point, to get $f(t_0 + \Delta t, x_1)$. This derivative is used to take a single forward Euler step *back* to t_0,

$$\tilde{x}_0 = x_1 - \Delta t\, f(t_0 + \Delta t, x_1).$$

4. The new perturbed initial value $\tilde{x}_0$ is used to compute an improved estimate L of the Lipschitz constant, as well as an estimate M of the logarithmic norm, [17]

$$L = \frac{\|f(t_0, \tilde{x}_0) - f(t_0, x_0)\|}{\|\tilde{x}_0 - x_0\|}; \qquad M = \frac{(\tilde{x}_0 - x_0)^{\mathrm{T}}(f(t_0, \tilde{x}_0) - f(t_0, x_0))}{\|\tilde{x}_0 - x_0\|^2}.$$

   We also compute a local error estimate for the explicit Euler step, $e_1 = \|\tilde{x}_0 - x_0\|$, where it is easily shown that $e_1 \approx \Delta t^2 \|\ddot{x}_0\|$.

5. Given the local error tolerance TOL and the local order $q = p+1$ of the method we intend to start with, we compute the scaling factors

$$\kappa_a = \frac{1}{\sqrt{e_1}}, \qquad \kappa_s = \frac{1}{\Delta t(L + M/2)},$$

   and then their average $\kappa = (\kappa_a + \kappa_s)/2$. This is used to compute the proposed initial step size, $h_0 = \kappa \cdot \mathrm{TOL}^{1/q} \cdot \Delta t$. Finally, to protect against a too large initial step, $h_0 = \min(h_0, 10^{-3}(t_f - t_0))$.

In the algorithm above, κ_a handles accuracy. Thus the 2nd order error estimate for the explicit Euler step, forward and back, is rescaled to order p by computing $\sqrt{e_1}$ and using the factor $\mathrm{TOL}^{1/q}$. Likewise, the factor κ_s handles stability requirements. Disregarding M for a moment, the construction requires that $(h_0 L)^q \approx 1$; this ensures that homogeneous solutions are stable and accurately represented at a step size of h_0. Finally, M assists in the handling of stability — in case $M < 0$ the factor κ_s becomes larger than if $M > 0$, thus allowing a somewhat larger initial step size for problems exhibiting dissipativity than for those that are unstable at the initial point. The averaging of κ_a and κ_s aims to make sure that both stability and accuracy requirements are accounted for.

The initial step size algorithm above uses a total of four function calls and can be replaced if necessary. In the software, we also add extra provisions to enable the algorithm to integrate problems both in forward and reverse time.

4.3. Starting values

After the initial step-size is chosen according to the algorithm above, an explicit Runge-Kutta method is used to generate the next $k-1$ starting values that are needed for a k-step method. Here we use MATLAB's solver, `ode45`, based on the Dormand–Prince (4,5) pair with constant step size h_0.

4.4. Error control and step size selection

The classical approach to controlling errors by varying the step size is multiplicative (linear) control,

$$h_n = \omega_{n-1} h_{n-1}, \tag{4.1}$$

where the step size change factor ω_{n-1} (the step ratio) is constructed from error estimates produced by the time stepping method. In the simplest approach, one takes

$$\omega_{n-1} = \left(\frac{\mathrm{TOL}}{e_{n-1}}\right)^{1/q},$$

where TOL is a given error tolerance, e_{n-1} is the (norm of the) local error on the current step, and q is related to the order of the method; the choice $q = p + 1$ is used for local *error per step* control, while $q = p$ for *error per unit step* control. The latter choice is preferred when it is essential that the adaptive method produces a *tolerance proportional* global error.

The classical error control is referred to as *integrating control*; taking logarithms, one finds that

$$\log h_n = \log h_{n-1} + \frac{1}{q}(\log \text{TOL} - \log e_{n-1}).$$

Hence the deviation between the estimated error and the tolerance (known as the control error) is summed up ("integrated") in the process of generating the step size sequence.

The classical error control has several shortcomings, and practical implementations typically include major restrictions on the step size selection, including exception handling. Many of these are potentially harmful to a multistep method, which requires step size changes to be small and smooth. Improved schemes, generating smooth, stable step size sequences, can be constructed using discrete control theory and digital filters, see [16] and the survey [19]. Still within the multiplicative format (4.1), a general and highly versatile control structure is given by $h_n = \omega_{n-1} h_{n-1}$ together with the dynamic step size ratio recursion

$$\omega_{n-1} = c_{n-1}^{\beta_1} \cdot c_{n-2}^{\beta_2} \cdot \omega_{n-2}^{-\alpha}, \tag{4.2}$$

where the scaled control error, accounting for method order, is given by

$$c_{n-1} = \left(\frac{\text{TOL}}{e_{n-1}} \right)^{1/q}. \tag{4.3}$$

Controllers can be designed for special needs. The parameter triple $(\beta_1, \beta_2, \alpha)$ offers a wide range of control structures and digital filters, and Table 4.1 shows the types we have chosen for the variable step parametric multistep methods.

Table 4.1: Basic step size controllers.

Controller	Type	β_1	β_2	α
Classic	I (deadbeat integral control)	1	0	0
PI3040	PI (proportional-integral)	7/10	−4/10	0
PI3333	PI (proportional-integral)	2/3	−1/3	0
PI4020	PI (proportional-integral)	3/5	−1/5	0
H211PI	PI digital filter	1/6	1/6	0
H211b	Noise shaping digital filter	$1/b$	$1/b$	$1/b$

The three PI (proportional-integral) controllers are primarily intended for solvers with bounded stability regions (methods of type E_k and I_k^+ for nonstiff problems), while the two digital filters, H211PI and H211b, are primarily intended for stiff solvers of type I_k. The parameter b in H211b can be varied in the range $b \in [3, 6]$ to tune the noise-shaping filter; a higher value of b leads to smoother step size sequences, but at the cost of a more sluggish response. The value $b = 4$ is recommended as a default, [16]. With the exception of the parameter b, controller parameters should never be varied arbitrarily, as this typically destroys the control design. For example, a first order low-pass filter, as in H211PI and H211b, must necessarily satisfy the order condition $\beta_1 = \beta_2$.

The controllers are used with as little interference as possible. With the exception of the classical deadbeat I controller, they are designed to produce smooth step size sequences, using minute changes without lower bound. Step size changes mostly remain within a 5% bracket. Bounds on maximum increase and decrease can be selected, as large step size changes may cause stability problems. Rejection is based on whether a drastic step size reduction is proposed, cutting the step size by more than 20% in a single step due to a large estimated error. This is typically a rare event, which allows errors to be up to $\sim 2 \cdot \text{TOL}$. While this may seem overly lenient, one notes that estimated as well as actual errors continually deviate from TOL, being both too large and too small. However, as the controllers are expectation value correct, small deviations over time tend to cancel out in the global error accumulation process.

Because the controller (4.2) is a two-step controller, it needs to be initialized. On the first step (or after a restart) when c_0 has been computed and c_{-1} and ω_{-1} are missing, the controller is initialized by simply setting $c_{-1} = 1$ and $\omega_{-1} = 1$. After a rejected step, anti-windup is employed; this means that c_{n-2} is rescaled to match the recomputed step size that replaces the failed step size. The algorithmic features associated with rejection require some careful attention to work reliably.

4.5. Error control modes and accuracy criteria

In a method that constructs a polynomial $P_n(t)$ at the nth point, it is simple to estimate the error. The new point is predicted by $P_{n-1}(t_n)$ and the computed result is $P_n(t_n)$. It follows that the local error can be estimated by the difference, $\ell_n = P_n(t_n) - P_{n-1}(t_n)$. The accuracy criterion, however, is typically a mix of relative and absolute errors. Hence there is both an absolute and a relative tolerance. Let the error's ith component be $l_{n,i}$ and define a vector r component-wise by

$$r_i = \frac{\ell_{n,i}}{w_i|x_{n,i}| + \sigma_i},$$

where $w = \{w_i\}$ is a vector of *weights* for relative error control and $\sigma = \{\sigma_i\}$ is a vector of *scales* for absolute error control. Having constructed r, we compute $e = \|r\|$, where the default norm is the Euclidean norm. If error per unit step is controlled, we must also divide by the current step size, i.e., $e = \|r\|/h$.

Because the control system is single-input/single-output, we only use a single scalar tolerance TOL to adapt the step size so that $e = \text{TOL}$. This construction still conforms to the common RTOL – ATOL approach used in current software. Thus, by defining

$$\text{RTOL}_i = w_i \cdot \text{TOL}; \qquad \text{ATOL}_i = \sigma_i \cdot \text{TOL},$$

our scalar TOL, together with weights w and scales σ, implements the standard approach. Absolute error control is achieved by taking $w_i = 0$ and $\sigma_i = 1$, and pure relative control (not recommended as it breaks down if $x_{n,i}$ has a zero passage) is obtained for $w_i = 1$ and $\sigma_i = 0$. Usually, w_i is chosen to be one or zero, controlling whether $x_{n,i}$ is included or excluded from relative error control. When included (i.e., $w_i = 1$), the corresponding scale σ_i becomes the breakpoint between absolute error and relative error control. Thus, relative error is used whenever $|x_{n,i}| \gg \sigma_i$ and absolute whenever $|x_{n,1}| \ll \sigma_i$. Our codes are run with a default setting of scalar tolerances, $\text{RTOL} = \text{TOL} = 10^{-6}$ and $\text{ATOL} = 10^{-3}\text{TOL}$. This corresponds to the default settings in MATLAB, and makes our numerical test results directly comparable to those obtained with MATLAB's solvers in default mode.

5. Numerical Tests and Validation

We have used several classical nonlinear test problems, as well as an artificial, nonstiff, nonlinear problem with a known solution, to test the methods, implementation details and supplementary control algorithms. The nonstiff artificial problem and a stiff van der Pol equation, are used below for the purpose of validating the new variable step size multistep methods. Throughout, the step size is controlled automatically as specified above. Method order is not varied, but remains constant in all tests; although order control is also of interest in adaptive methods, it is beyond the scope of this particular study to simultaneously vary step size and order. The results of the tests are shown in Figures 5.1 - 5.4.

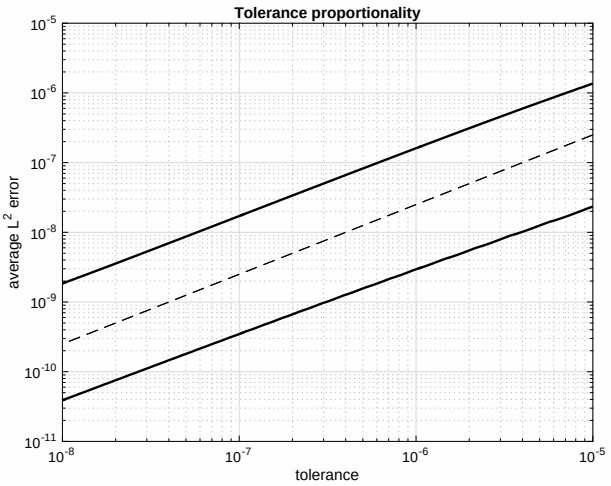

Fig. 5.1. *Tolerance proportionality.* Nonstiff, nonlinear problem P1 was solved with a three-step method of order $p = 3$, as well as by a six-step method of order $p = 6$ (both explicit E_k methods). Absolute local error per unit step was controlled, governed by PI3333. Global L^2 error for $p = 3$ (solid, top) and $p = 6$ (solid, bottom) shows excellent tolerance proportionality when compared to the unit slope reference line (dashed). As is normally the case, the problem- and method-dependent errors deviate from requested accuracy. The graphs further demonstrate exceptional computational stability – when 150 intermediate tolerances were used, there is no discernible irregularity at all in the achieved global L^2 error, which "depends continuously" on TOL. Thus, reducing TOL by any given factor will result in a corresponding, predictable error reduction

- Problem 1 [**P1**] The nonstiff, nonlinear system

$$\dot{y}_1 = y_1 + y_2^2,$$
$$\dot{y}_2 = -y_1,$$

  with initial value $y(0) = (1 \quad 3)^{\mathrm{T}}$ is used to test the new methods with respect to order and accuracy. It has the exact solution $y_1(t) = e^t - 3e^{-2t}$; $y_2(t) = 3e^{-t}$.

- Problem 2 [**P2**] The nonlinear van der Pol problem

$$\dot{y}_1 = y_2,$$
$$\dot{y}_2 = \mu(1 - y_1^2)y_2 - y_1,$$

Fig. 5.2. *Work/precision diagram.* A five-step, 5th order explicit E_k method with parameters $[7\pi/12\ 7\pi/16\ 17\pi/32\ 31\pi/64]$ was used to solve problem P1 and is compared to the solution obtained with MATLAB's 5th order Dormand–Prince `ode45` solver. Top curve (dashed) shows total number of steps for `ode45` vs achieved accuracy. Bottom curve (solid) shows the corresponding data for the E_k method, which uses approximately half the number of steps at any given accuracy. Absolute local error per unit step was control was controlled, using the PI3333 controller in the multistep method. A total of eleven different tolerance settings were used in each method to generate the graphs

with initial value $y(0) = (2 \quad 0)^{\mathrm{T}}$ is used to test stiff solvers, as the parameter μ controls stiffness. No analytic solution is known, but the problem is frequently used as a test problem with its challenging dynamic behavior. The problem is solved on the interval $[0, T] = [0, \mu]$ to cover a little more than half a period of the limit cycle. (As the other half leads to a similar behavior, it is sufficient to cover one stiff branch, as well as the sharp transition to the other stiff branch.) This problem is also used to specifically assess stiff performance, by using a shorter interval that excludes the nonstiff transition. Stiffness scales in proportion to μ^2, [20], and we use $\mu = 500$ and $\mu = 1200$ in the tests. This is large enough to exclude the use of explicit methods, which would require in excess of 10^5 – 10^6 steps to solve the problem. By contrast, our stiff solvers manage with a few hundred.

6. Conclusions

This paper has developed a new, comprehensive methodology for constructing high order variable step-size multistep methods. Contrary to the classical approach of extending fixed step-size formulas case by case, the new technique constructs multistep methods in terms of interpolation and collocation conditions defining a polynomial, which generates the next solution point, as well as a "continuous extension" of the multistep method. For equidistant time steps the classical methods are obtained. A main advantage of this approach is that whether the step size varies or not, each method is defined in terms of a single, fixed set of parameters. Thus time step adaptivity is supported without additional difficulties.

Hairer et al. [9] give variable step-size formulas for the 2-step Adams-Moulton and BDF methods. Our technique renders exactly the same variable step-size formulation in these two

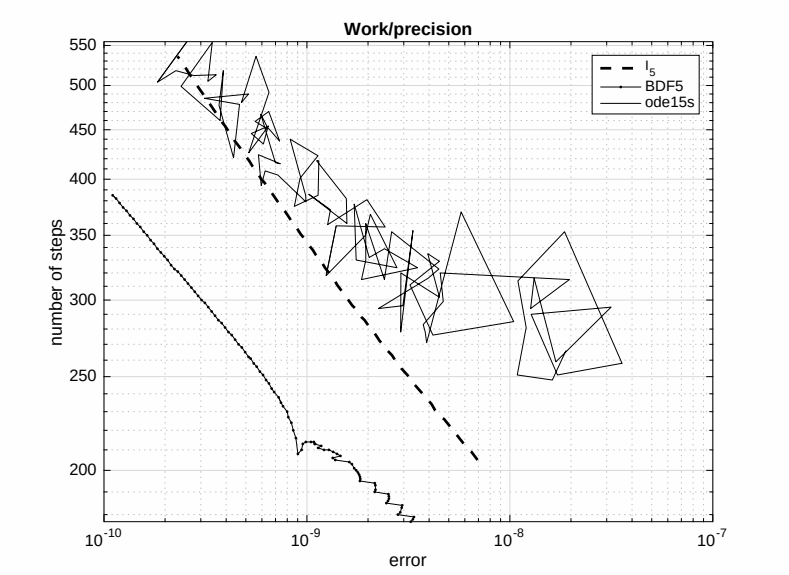

Fig. 5.3. *Work/precision diagram.* The stiff van der Pol problem P2 with $\mu = 500$ was solved by a five-step 5th order implicit I_k method with parameters $[\pi/24\ \pi/512\ -\pi/512\ \pi/128]$ as well as by BDF5 in its new I_k implementation. Absolute local error per step was controlled, governed by the digital filter H211PI. A total of one hundred different values of ATOL = TOL were used. For comparison, the problem was also solved using MATLAB's `ode15s`, running the BDF option, with the same absolute error criteria and tolerance settings, but using its own step size controller. Work in terms of steps is plotted vs achieved global accuracy, measured in the L^2 norm. The I_k BDF5 (bottom) uses approximately half as many steps as `ode15s` (top) for a given global accuracy, and for a given number of steps achieved accuracy is approximately one order of magnitude better. More importantly, due to strongly improved computational stability, both the I_k BDF5 (bottom) as well as the new 5th order I_k method (center, dashed) have smooth work-precision graphs, while the `ode15s` implementation (top) shows an erratic response to changes in TOL. Still, in terms of wall-clock time, `ode15s` is somewhat faster

cases. Furthermore, we have constructed a continuum of multistep methods that depend on its defining parameters in $(-\pi/2 \quad \pi/2]$. As the parameters change, method properties such as zero-stability, error coefficient and stability region typically change continuously.

Using this methodology, we have developed unified implementations of explicit methods of order $p = k$, and implicit methods of orders $p = k$ and $p = k + 1$. The three method categories E_k, I_k and I_k^+ cover all higher order methods, as well as all practical needs for stiff and nonstiff problems. Should a "new" high order multistep method be proposed in the literature, it can be implemented in an instant, following the identification of its θ_j parameters; the latter task requires the use of a symbolic package, such as MAPLE.

The methodology and implementation offers

- a new understanding of variable step size multistep methods
- a platform for computational experiments with any high order multistep method
- a platform for analyzing particular properties of multistep methods

Among the software's specific features, we have included

- a new robust starting step size algorithm

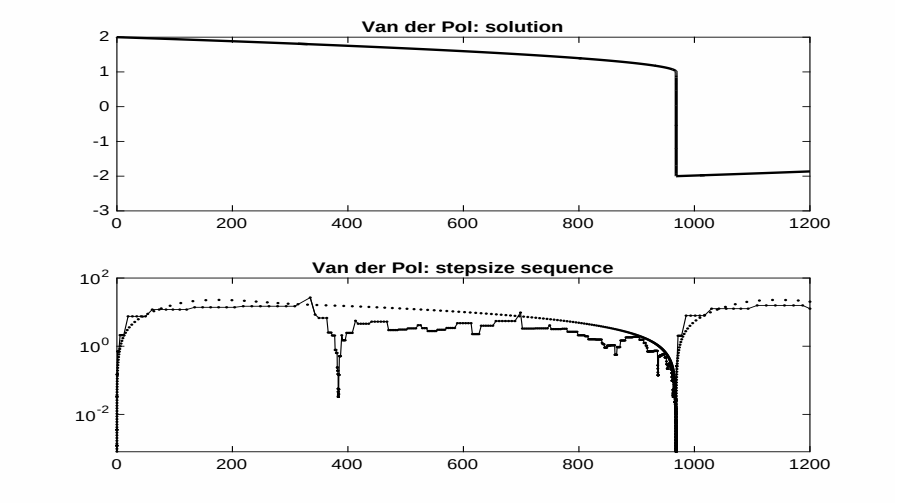

Fig. 5.4. *Step size sequences.* The stiff van der Pol problem P2 with $\mu = 1200$ was solved on the interval [0 1200] using the BDF5 method in its new I_k implementation, and compared to MATLAB's `ode15s` running the BDF option. Top panel shows the solution, while bottom panel shows step size sequences. Both codes control error per step at RTOL $= 10^{-8}$ and ATOL $= 10^{-11}$. Total number of steps were 1100 for I_k and 1634 for `ode15s`. Unlike `ode15s`, which gives preference to piecewise constant step size (solid-dots), in the I_k method the step size sequence is controlled by the digital filter H211PI, generating a smooth step size sequence (discrete dots). As a result, step size ratios in the I_k code are smaller, maintaining better stability. Thus "mistakes" of the type observed in the red sequence at $t = 385$ are eliminated

- a full range of step size controllers based on PI control and digital filters
- smooth step size sequences conforming to variable step size zero stability conditions
- full functionality with respect to error control criteria
- functionality for multiple order method families by increasing the dimension of the method's defining parameter vector

The software, which can be obtained from the authors, is intended for research purposes, such as method and algorithmic development. Tests carried out on well-known benchmark problems indicate that the method construction is robust and performs in full agreement with theory. The implementation is competitive insofar as step sizes and direct measures of efficiency are concerned – thus, work/precision diagrams are improved compared to standard multistep software; accuracy is enhanced; step size sequences are smoother; and fewer steps are typically required. Computational stability is much enhanced, and at large, achieved accuracy "depends continuously" on TOL, see [18].

Still, the methodology needs further development. Wall-clock time is typically longer than for standard software, especially when using I_k^+ methods, due to the current parametrization requiring the solution of a larger system on each step than what is necessary in standard software. In addition, we have not included automatic order control, which needs to be coordinated with the advanced step size control. Finally, no provisions have been included for "standard" add-ons typically featured in production codes, such as sparse or matrix-free solvers, event handling, discontinuities, and implicit output or stopping criteria.

References

[1] C. Arévalo, J. Díaz López and G. Söderlind, Constant coefficients linear multistep methods with step density control, *J. Comp. and Appl. Math.*, **205** (2007), 891–900.

[2] C. Arévalo, Claus Führer and Mónica Selva, A collocation formulation of multistep methods for variable step-size extensions, *Appl. Numer. Math.*, **42** (2002), 5–16.

[3] C. Arévalo, C. Führer and G. Söderlind, β-blocked multistep methods for Euler-Lagrange DAEs: Linear analysis, *ZAMM*, **77** (1997), 1–9.

[4] C. Arévalo, Claus Führer and Gustaf Söderlind, Regular and singular β-blocking of difference corrested multistep methods for nonstiff index-2 DAEs, *Appl. Numer. Math.*, **35** (2000), 293–305.

[5] P.N. Brown, G.D. Byrne and A.C. Hindmarsh, VODE, A variable-coefficient ODE solver, *SIAM J. Sci. Stat. Comput.*, **10** (1989), 1038–1051.

[6] M.T. Chu, An automatic multistep method for solving stiff initial value problems, *J. Comp. Applp. Math.*, **9** (1983), 229–238.

[7] T.P. Coffee, J.M. Heimerl and M.D. Kregel, A numerical method to integrate stiff systems of ordinary differential equations, Technical report ARBRL-TR-02206, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland, 1980.

[8] Y. Hadjimichael, D. Ketcheson, L. Lóczi and A. Németh, Strong stability preserving explicit linear multistep methods with variable step size, arXiv:1504.04107v1 [math.NA], (2015)

[9] E. Hairer, S.P. Nørsett and G. Wanner, Solving Ordinary Differential Equations I. Nonstiff Problems, Springer Ser. Comput. Math., Vol. 8, Springer, New York, 1993.

[10] E. Hairer and G. Wanner, Solving Ordinary Differential Equations II. Stiff and Differential-Algebaic Problems, Springer Ser. Comput. Math., Vol. 14, Springer, New York, 1996.

[11] A. Nordsieck, On numerical integration of ordinary differential equations, *Math. Comp.*, **16** (1962), 22-49.

[12] K. Radhakrishnan and A.C. Hindmarsh, Description and use of LSODE, the Livermore solver for ordinary differential equations, LLNL report UCRL-ID-113855, December 1993.

[13] G.K. Rockswold, Implementation of α-type multistep methods for stiff differential equations, *J. Comput. and Appl. Math.*, **22** (1988), 63–69.

[14] R.D. Skeel, Construction of variable-stepsize multistep formulas, *Math. Comp.*, **47**(176) (1986), 503–510, S45-S52.,

[15] G. Söderlin, A multi-purpose system for the numerical integration of ODEs, *Appl. Math. Comput.*, **31** (1989), 346–360.

[16] G. Söderlind, Digital filters in adaptive time-stepping, *ACM Trans. on Math. Software*, **20** (2003), 1–26.

[17] G. Söderlind, Logarithmic norms. History and modern theory, *BIT Numerical Mathematics*, **46** (2006), 631–652.

[18] G. Söderlind and L.Wang, Adaptive time-stepping and computational stability, *J. Comput. and Appl. Math.*, **56** (2006), 225–243.

[19] G. Söderlind, Time-step selection algorithms: Adaptivity, control, and signal processing, *Appl. Num. Math.*, **185** (2006), 488–502.

[20] G. Söderlind, L. Jay and M. Calvo, Stiffness 1952–2012. Sixty years in search of a definition, *BIT Num. Math.*, **55** (2015), 531–558.

[21] D. Wang and S.J. Ruuth, Variable step-size implicit-explicit linear multistep methods for time-dependent partial differential equations, *J. Comput. Math.*, **6** (2008), 838–855.