DIMSEMs — Diagonally Implicit Single-Eigenvalue Methods for the Numerical Solution of Stiff ODEs on Parallel Computers †

by Robert F. Enenkel ‡ and Kenneth R. Jackson ‡

September 1996

Accepted for publication by Advances in Computational Mathematics

Abstract

This paper derives a new class of general linear methods (GLMs) intended for the solution of stiff ordinary differential equations (ODEs) on parallel computers. Although GLMs were introduced by Butcher in the 1960s, the task of deriving formulas from the class with properties suitable for specific applications is far from complete. This paper is a contribution to that work.

Our new methods have several properties suited for the solution of stiff ODEs on parallel computers. They are strictly diagonally implicit, allowing parallelism in the Newton iteration used to solve the nonlinear equations arising from the implicitness of the formula. The stability matrix has no spurious eigenvalues (that is, only one eigenvalue of the stability matrix is non-zero), resulting in a solution free from contamination from spurious solutions corresponding to non-dominant, non-zero eigenvalues. From these two properties arises the name DIMSEM, for Diagonally IMplicit Single-Eigenvalue Method. The methods have high stage order, avoiding the phenomenon of order reduction that occurs, for example, with some Runge-Kutta methods. The methods are L-stable, with the result that the chosen stepsize is dictated by convergence requirements rather than stability considerations imposed by the stiffness of the problem.

An introduction to GLMs is given and some order barriers for DIMSEMs are presented. DIMSEMs of orders 2–6 are derived, as well as an L-stable class of diagonal methods of all orders which do not, however, possess the single-eigenvalue property. A fixed-order, variable-stepsize implementation of the DIMSEMs is described, including the derivation of local error estimators, and the results of testing on both sequential and parallel computers is presented. The testing shows the DIMSEMs to be competitive with fixed-order versions of the popular solver LSODE on a practical test problem.

Key words: Parallel, Numerical Solution, Initial Value Problem, IVP, Ordinary Differential Equation, ODE, General Linear Method, GLM, Order, Stability.

AMS(MOS) subject classification: 65L05.

---

† This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Center of Ontario.

‡ Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, M5S 3G4
E-mail: {enenkel,krj}@cs.toronto.edu

1 Introduction and Background

In this paper we derive numerical methods for the parallel solution of stiff initial-value problems (IVPs) for ordinary differential equations (ODEs) of the form

$$\begin{aligned} u'(x) &= f(u(x)) \quad \forall x \in [x_0, x_{end}], \\ u(x_0) &= u_0, \end{aligned}$$

where $u : \mathbb{R} \to \mathbb{R}^m$. For notational convenience we assume a scalar equation ($m = 1$) unless otherwise noted, but the formulas generalise easily to vectors with the use of tensor product notation. Methods for solving IVPs typically compute approximations to $u(x_n)$, $n = 1, \ldots, N$, on some mesh $x_0 < x_1 < \cdots < x_N = x_{end}$, where the approximation at each successive step depends on one or more of the approximations at previous steps. We seek methods which permit parallel computation within each step of the solver, rather than attempting to replace the entire sequential step-by-step integration with a parallel process. Our methods will therefore be suitable for machines with a few processors, such as shared-memory machines, rather than the hundreds or thousands of processors available with local-memory, message-passing architectures.

1.1 General Linear Methods

The methods developed here are members of the class of general linear methods (GLMs) introduced by Butcher. (See [9] for a survey.) They are also known as multivalue methods (MVMs) [5]. Given a vector y_n of external stages on the current step, they first compute a vector of internal stages,

$$Y_n = A_1 y_n + hB_1 F(Y_n) \in \mathbb{R}^s,$$

and then compute a new vector y_{n+1} of external stages from it by

$$y_{n+1} = A_2 y_n + hB_2 F(Y_n) \in \mathbb{R}^r.$$

Here h is a fixed stepsize (extension to a variable stepsize will be considered later), $F(Y_n)$ is the vector formed by applying f to each component of Y_n, and A_1, B_1, A_2 and B_2 are matrices of constant coefficients which determine the method. The components of the external stages y_n may in different methods represent approximations to the solution at various points, derivatives of the solution, or other quantities. The internal stages are approximations to the solution at points determined by the internal abscissae $c_1, \ldots, c_s$,

$$Y_{n,i} \approx u(x_n + c_i h), \quad i = 1, \ldots, s.$$

A starting method is required to provide the initial approximation y_0, and, if the solution itself is not a component of y_n, a finishing method is needed to recover it [4, 9].

Two common types of GLM are Nordsieck methods, in which the external stages approximate scaled derivatives of the solution of the ODE, and multistep-type GLMs, in which they approximate the solution itself at various points. That is, for a Nordsieck GLM,

$$y_{n,i} \approx \frac{h^{i-1}}{(i-1)!} u^{(i-1)}(x_n), \quad i = 1, \ldots, r, \tag{1}$$

while for a multistep-type GLM,

$$y_{n,i} \approx u(x_n + d_i h), \quad i = 1, \ldots, r. \tag{2}$$

The d_i's are called external abscissae. A special type of multistep-type GLM is the Multistep Runge-Kutta (MRK) method, in which $d_i = i - r$, $i = 1, \ldots, r$,

$$A_2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 & 0 \\ 0 & \cdots & \cdots & 0 & 1 \\ \alpha_1 & & \cdots & & \alpha_r \end{bmatrix}, \tag{3}$$

and

$$B_2 = \begin{bmatrix} 0 & \cdots & 0 \\ & \vdots & \\ 0 & \cdots & 0 \\ \gamma_1 & \cdots & \gamma_s \end{bmatrix}, \tag{4}$$

for some α_i's and γ_i's.

The representation of GLMs by their coefficient matrices A_1, B_1, A_2, and B_2, is in some sense not unique. Two different methods may be equivalent in that their solution vectors y_n are related by a linear transformation. In particular, any Nordsieck GLM has an equivalent multistep-type GLM in this sense, and vice-versa.

A GLM is called explicit if B_1 is strictly lower triangular, that is, lower triangular with zeroes on the diagonal. Otherwise it is implicit. Although the computation of the internal stages Y_n is considerably less expensive in an explicit method, only implicit methods are suitable for the solution of stiff problems [16, Theorem 8]. In an implicit method, computing Y_n at each step requires the solution of a system of generally nonlinear equations,

$$Y_n = A_1 y_n + hB_1 F(Y_n).$$

This is typically performed using a Newton iteration,

$$\begin{aligned} (I_s - hB_1 F'(Y_n^{(k)}))\Delta_n^{(k)} &= -R_n(Y_n^{(k)}) \\ Y_n^{(k+1)} &= Y_n^{(k)} + \Delta_n^{(k)}, \end{aligned} \tag{5}$$

(or some variant in which $F'(Y_n^{(k)})$ is replaced by an approximation) where

$$R_n(Y) = Y - A_1 y_n - hB_1 F(Y)$$

is the function whose zero is the internal stage vector Y_n, and $Y_n^{(k)}$ is the k-th iterate approximating Y_n.

1.2 Parallelism in Implicit GLMs

The parallelism inherent in an implicit GLM depends on the structure of B_1. (Note that, in the following, the names given to GLMs of various structures vary with different authors.)

A *diagonally implicit* GLM has

$$B_1 = \begin{bmatrix} \lambda_1 & & & & \\ b_{21} & \lambda_2 & & & \\ b_{31} & b_{32} & \lambda_3 & & \\ \vdots & \vdots & & \ddots & \\ b_{s1} & b_{s2} & \cdots & b_{s,s-1} & \lambda_s \end{bmatrix}.$$

With this structure, each internal stage value $Y_{n,i}$ is coupled only to those before it, so that the $Y_{n,i}$'s may be solved for in sequence. If $\lambda_1 = \cdots = \lambda_s$, the method is called *singly diagonally implicit*. Such methods have been investigated, for instance, by Burrage and Chipman [6], for use on sequential machines.

In a *strictly diagonally implicit* GLM, B_1 is a diagonal matrix,

$$B_1 = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_s \end{bmatrix}.$$

This decouples the component equations of the Newton iteration (5) into

$$(1 - h\lambda_i f'(Y_{n,i}^{(k)}))\Delta_{n,i}^{(k)} = -R_n(Y_n^{(k)})_i, \quad i = 1, \ldots, s, \tag{6}$$

with the effect that $\Delta_{n,i}^{(k)}$, $i = 1, \ldots, s$, can be solved for independently. For an m-dimensional IVP, (6) becomes

$$(I_m - h\lambda_i f'(Y_{n,i}^{(k)}))\Delta_{n,i}^{(k)} = -R_n(Y_n^{(k)})_i, \quad i = 1, \ldots, s, \tag{7}$$

with $\Delta_{n,i}^{(k)} \in \mathbb{R}^m$, and a separate factorisation of

$$(I_m - h\lambda_i f'(Y_{n,i}^{(k)}))$$

is required for each distinct λ_i. (Or, some other scheme must be used to accommodate the distinct λ_i's, such as Enright's factorisation [17] or an iterative linear solver.) The s factorisations however can be done in parallel, as can the s solves for $\Delta_{n,i}^{(k)}$, $i = 1, \ldots, s$.

If we restrict B_1 to be a multiple of the identity, $B_1 = \lambda I$, the result is a *singly strictly diagonally implicit* GLM. For an m-dimensional problem, (7) becomes

$$(I_m - h\lambda f'(Y_{n,i}^{(k)}))\Delta_{n,i}^{(k)} = -R_n(Y_n^{(k)})_i, \quad i = 1, \ldots, s,$$

where $\Delta_{n,i}^{(k)} \in \mathbb{R}^m$. On a sequential machine, it is common to approximate each $(I_m - h\lambda f'(Y_{n,i}^{(k)}))$, $i = 1, \ldots, s$, by a single matrix $N_n^{(k)}$, to avoid having to perform s factorisations per iteration. The constant diagonal λ in a singly implicit method is therefore an advantage on a sequential machine. On a parallel machine, however, each factorisation, as well as each of the subsequent solves, can be assigned to a different processor, so there is less incentive to limit the structure of B_1 in this way.

Butcher recently introduced a class of methods called DIMSIMs [11, 12, 13, 14] which contains methods with each of the above structures. Because of their particular relevance to our work, these methods will be discussed separately later.

1.3 Error Coefficients

An important property of a method is the accuracy of the generated solution, and the dependence of this accuracy on the stepsize. This leads to the concepts of local and global discretisation error, error coefficients and order of accuracy. Local discretisation error is the error in the solution produced at the end of a single step started with exact information, while global error is the error after a sequence of steps, only the first of which is started with exact information. In our formulation, error coefficients are a problem-independent characterisation of the local discretisation error.

A valuable tool for the analysis of numerical methods for ODEs is the series expansion known as a Butcher series (or B-series),

$$B(\varphi, u) = \sum_{t \in T} \varphi(t) \frac{h^{\rho(t)}}{\rho(t)!} D(t)(u). \tag{8}$$

Here T is the set of monotonically labeled rooted trees, $\varphi : T \to \mathbb{R}^l$ for some l (in our application l is either r or s), $\rho(t)$ is the order of tree t (that is, the number of nodes in t), and $D(t)$ is the elementary differential associated with the ODE for the tree t. Let ϕ be the empty tree, which is the tree with no nodes, and let $\bullet$ be the tree with only a root node. Let $t = [t_1, \ldots, t_k]$ represent the tree formed by joining each of the trees $t_1, \ldots, t_k$ with a new branch from each of their respective root nodes to a new node, which becomes the root of t. Then for the ODE $u'(x) = f(u(x))$, $D(t)$ is defined recursively by

$$D(\phi)(u) = u,$$

$$D(\bullet)(u) = f(u),$$

and

$$D(t)(u) = f^{(k)}(u)\,(D(t_1)(u) \cdots D(t_k)(u)) \quad \text{for} \quad t = [t_1, \ldots, t_k].$$

If $m > 1$, the multiplication of vectors is done componentwise. More on Butcher series may be found in [4, 5, 10, 19, 20, 21, 22]. As an example, if $\varphi(t) = c^{\rho(t)}$, then $B(\varphi, u(x_n))$ is the Taylor series for $u(x_n + ch)$ expanded about the point x_n.

We now summarise the result of using Butcher series to analyse the local discretisation error in a GLM stepping from y_n to y_{n+1}. Let $\hat{y}_k$ be the vector of exact values to which the external stage vector y_k is an approximation, for every step k. By appropriate choice of what is known as the true-value function, $q : T \to \mathbb{R}^r$, we can expand $\hat{y}_k$ in a Butcher series around x_k,

$$\hat{y}_k = B(q, u(x_k)) \quad \forall k, \tag{9}$$

where $u : \mathbb{R} \to \mathbb{R}$ is the true solution of the IVP. Let $y_n = \hat{y}_n$, so that the method starts the n-th step with exact information. The local discretisation error l_n is then defined by

$$\begin{aligned} l_n &= y_{n+1} - \hat{y}_{n+1} && (10)\\ &= B(e, u(x_n)) && (11) \end{aligned}$$

for an appropriate choice of $e : T \to \mathbb{R}^r$, where $e(t)$ is the vector of (local discretisation) error coefficients of the method for the tree t. Using the theorem of Hairer and Wanner [20] on the composition of Butcher series, we can express $e(t)$ recursively in terms of coefficients of trees of lower order, the true-value function q, and the coefficient matrices A_1, B_1, A_2, B_2 of the method. The resulting expressions are the basis for the computer programs we wrote to find error coefficients for our methods.

1.4 Order of Accuracy and Simplifying Assumptions

The order of a method is a characterisation of the sensitivity of the error to changes in stepsize. The global error is defined as

$$g_n = y_{n+1} - \hat{y}_{n+1},$$

where

$$y_0 = \hat{y}_0.$$

A method is said to have global order p (or, simply, order p) if p is the largest value such that $g_n = O(h^p)$ for all n and all sufficiently differentiable problems. Similarly, a method is said to have local order $p + 1$ if p is the largest value such that $l_n = O(h^{p+1})$ for all n and all sufficiently differentiable problems. For a stable method, a global error of order p usually corresponds to a local discretisation error of order $p + 1$, although it is possible in certain situations to have local and global discretisation errors of the same order [15, 25].

The equations which must be satisfied for a method to have a certain local order are called the order conditions. For a method of local order $p+1$, we require that $e(t) = \mathbf{0}$ for all trees t of order $\rho(t) \le p$, where $e(t)$ are the error coefficients. This results in global order at least p, provided the method is stable. Solving these order conditions directly for methods of order greater than four is extremely difficult, however, as they consist of large systems of nonlinear equations. For this reason it is customary to employ simplifying assumptions, which are sufficient, but not necessary, conditions for the order conditions to hold.

The methods developed in this paper make use of the A, B and C simplifying assumptions of Burrage and Moss [7]. These are defined as

$$A(w) \quad : \quad q(t) = q(\tau^{\rho(t)}), \quad \forall t \ni \rho(t) \le w, \tag{12}$$

$$B(w) \quad : \quad A_2 q(\tau^k) + kB_2 c^{k-1} = \sum_{i=0}^{k} \binom{k}{i} q(\tau^i), \quad k = 0, \ldots, w, \tag{13}$$

$$C(w) \quad : \quad A_1 q(\tau^k) + kB_1 c^{k-1} = c^k, \quad k = 0, \ldots, w, \tag{14}$$

where kB_1c^{k-1} and kB_2c^{k-1} are taken as $\mathbf{0}$ when $k = 0$, and $\tau^k = [\bullet, \ldots, \bullet]$ is what is known as the *bushy* tree of order k. The importance of these simplifying assumptions is given by a theorem of Burrage and Moss [7], which states that a GLM satisfying $A(p)$, $B(p)$, and $C(p-1)$ has order at least p, provided the method is stable. Furthermore, the stage order (the order of the internal stages) is at least $p-1$, and if $C(p)$ is satisfied, it is at least p. The solution of these simplified order conditions, while by no means trivial, is much easier than that of the raw order conditions described earlier. More on simplifying assumptions for GLMs may be found in [3, 4, 7].

1.5 Stability Regions of GLMs

When a GLM is used with a sufficiently large fixed stepsize, the computed solution may become unstable, that is, may grow without bound, even though the true solution remains bounded or decays. For certain methods applied to so-called *stiff* problems, the stepsize necessary for stability may be much smaller than that necessary to produce a sufficiently accurate solution with a more appropriate method. An important indication of the suitability of a method is the size of its stability region, which we define for GLMs as follows:

Definition 1 *The stability region R of a GLM is given by*

$$\begin{aligned} R \quad = \quad & \{h\mu : \exists c \in \mathbb{R} \ni \textit{ the GLM applied with stepsize } h \textit{ to the test problem} \\ & u' = \mu u, \mu \in \mathbb{C}, \textit{ produces a solution } y_n \textit{ satisfying } \|y_n\| \le c\|y_0\| \; \forall n\}. \end{aligned}$$

This definition is somewhat stronger than an alternative formulation in which the external stage vector y_n is replaced by u_n, the approximation of $u(x_n)$ to which y_n corresponds. However, a method with bounded $\|u_n\|$'s and unbounded $\|y_n\|$'s is in some sense degenerate, and can likely be re-written without the use of any unbounded $y_{n,i}$'s.

Definition 1 can be shown to be equivalent to

$$R = \{z \in \mathbb{C} : \|M(z)^n\| \text{ is bounded } \forall n \ge 1\},$$

where

Definition 2 *The stability matrix $M(z)$ of a GLM is given by*

$$M(z) = A_2 + zB_2(I - zB_1)^{-1}A_1.$$

A third equivalent definition of R is

$$R = \{z \in \mathbb{C} : |\gamma| \leq 1 \text{ for all non-defective eigenvalues } \gamma \text{ of } M(z) \text{ and } |\gamma| < 1 \text{ for all defective eigenvalues } \gamma \text{ of } M(z)\}.$$

It is generally believed that explicit GLMs can have only finite stability regions: to achieve infinite stability regions an implicit method is required. (We prove this for a subclass of GLMs in [16, Theorem 8].) This means explicit methods are suitable only for non-stiff problems, and stiff problems require the use of an implicit method to avoid the penalty of stability imposed restrictions on the stepsize.

A desirable property for a method intended for the solution of stiff problems is that its stability region R include the left complex half-plane. A method with this property is called A-stable, and has the favourable characteristic that the stepsize is never restricted by stability on linear constant-coefficient problems, regardless of stiffness. If additionally $\|M(z)\| \to 0$ as $z \to \infty$, the method is called L-stable.

The previous discussion has considered stability for the scalar test problem

$$u' = \mu u, \quad \mu \in \mathbb{C}.$$

However, stability for the vector problem

$$u' = Cu, \quad C \in \mathbb{R}^{m \times m},$$

where C is diagonalisable, can be shown to follow from stability on each of the scalar problems

$$u' = \mu_i u, \quad i = 1, \ldots, m,$$

where $\mu_1, \ldots, \mu_m$ are the eigenvalues of C. This, as well as a weaker result which holds when C is not diagonalisable, is given by [16, Theorem 9].

1.6 Overview of the Paper

We now give an overview of the remainder of the paper, and identify the main results. The next section, Section 2, describes the development of our diagonal methods. In it, Section 2.1 defines the class of methods we refer to as DIMSEMs (Diagonally IMplicit Single-Eigenvalue Methods) and gives some motivation for this subclass of GLMs. Section 2.2 gives results on the order properties of single-eigenvalue methods, including the DIMSEMs, and shows that the single-eigenvalue property enables some Runge-Kutta order bounds to be extended to this class of methods. Section 2.3 develops an approach for the derivation of DIMSEMs, and Section 2.4 describes an offshoot class of methods, LDMs (for L-stable diagonal methods), which contains L-stable methods of all orders.

The main contribution of this paper is in two theorems, Theorems 4 and 5, (stated in Section 2.5) which describe how to construct DIMSEMs of any order greater than or equal to two. The remainder of the paper is concerned with selecting from this class specific methods of orders 2 to 6 and testing them.

More specifically, Section 2.6 deals with finding L-stable DIMSEMs and optimising the error coefficients through the selection of free parameters, while Section 2.7 considers optimisation of the LDMs. The variable-stepsize implementation and testing of the DIMSEMs are covered in Section 3, beginning with the investigation of variable-stepsize stability, error estimators and stepsize control schemes, and concluding with performance testing of the DIMSEMs on sequential and parallel machines. The testing demonstrates the viability of the methods by showing them to be competitive with a fixed-order version of the popular integrator LSODE on a practical test problem.

Conclusions and future work are discussed in Section 4.

2 Development of Diagonal Methods

2.1 How We Came to DIMSEMs

The goal of our research is to produce A-stable GLMs with diagonal B_1. Such methods are suitable for solving stiff problems on a parallel machine. A second goal is to find a whole class of such methods, together with a methodology for generating methods of any desired order. The first subclass of GLMs which we considered was Diagonally Implicit Multistep Runge Kutta methods (DIMRKs), which are MRKs with diagonal B_1. Two classes of these methods, arising from modifications to the methods of Burrage and Chipman [6], were investigated initially. The first satisfies $B(r+s-1)$ and $C(r-1)$, and hence has order at least r, with $c_1, \ldots, c_s$ and $\lambda_1, \ldots, \lambda_s$ as free parameters. The second satisfies $B(r)$ and $C(r)$, also has order at least r, and has $\gamma_1, \ldots, \gamma_{s-1}$ (defined in (4)) as free parameters. The simplified order conditions were solved algebraically using Maple [26], and the free parameter space was numerically searched for methods that appeared to be A-stable. The A-stability of these candidate methods was then verified theoretically using a technique based on the Schur criterion and the maximum principle of complex analysis [6]. Although methods of orders 2 to 5 were found this way, this simple approach does not appear to be extensible to higher orders due to the rapidly increasing cost, as order increases, of solving the order conditions by brute force. Another disadvantage of this approach is that it requires trial-and-error numerical searching to find A-stable methods. Little insight is available into the nature of A-stable choices of the parameters, as the generation of algebraic expressions for objects such as the characteristic polynomial of the stability matrix, or the eigenvalues, is computationally intractable.

Our eventual focus on a new class of methods for which A-stability is much easier to analyse grew out of a seminar [8] given by Butcher, in which he introduced his work on DIMSIM methods. Although Butcher's subsequent development of these methods takes a different direction from ours, the two classes are sufficiently related to warrant discussing DIMSIMs in the next section.

2.1.1 DIMSIMs — Diagonally Implicit Multi-Stage Integration Methods

Although our introduction to the GLMs known as DIMSIMs was through an early seminar by Butcher, they have now been described in published work [11, 12, 13, 14]. The methods are classified into four types, depending on the structure of B_1. These are defined in Fig. 1, which also indicates the type of problem and kind of architecture for which each type is suitable. Types 1 and 3 are explicit methods, intended for non-stiff problems, while type 2 is singly diagonally implicit, and type 4 is singly strictly diagonally implicit.

The DIMSIMs described in [11, 12, 13] are constructed with $r = s$, $A_1 = I$, $A_2 e = e$, where e is an r-vector of all 1's, and B_2 is determined from the simplified order conditions $A(r)$, $B(r)$, and $C(r)$ in terms of the c_i's, B_1, and A_2. This gives an order and a stage order of r. Although Butcher gives his order conditions in terms of the exponential function and integrals of interpolating polynomials instead of the expressions (12), (13), and (14), the two representations can be shown to be equivalent. Additional DIMSIMs in which various combinations of r, s, the order, and the stage order differ by one from the values above, are developed by Butcher and Jackiewicz in [14].

In his seminar, Butcher exhibited a type-2, order-2 DIMSIM which had been constructed to have the same stability region as a certain diagonally implicit Runge-Kutta (DIRK) method. This was achieved by making one eigenvalue of the stability matrix $M(z)$ zero for all z, and making the other eigenvalue equal to the stability function of the DIRK method. A generalisation of this example leads to the idea of *Single-Eigenvalue Methods* (SEMs), in which all but one eigenvalue of $M(z)$ is zero for all z. We use the term DIMSEM — *Diagonally IMplicit Single-Eigenvalue Method* — to describe a SEM having a diagonal B_1 matrix. Butcher [11] mentions it as a useful stability property, also possessed by Runge-Kutta methods. One benefit of a SEM is the absence of spurious solutions corresponding to non-zero, non-principal eigenvalues, although other factors tend to be

DIMSIM	B_1 Structure	Problem	Machine
Type 1	$\begin{bmatrix} 0 & & & & \\ b_{21} & 0 & & & \\ b_{31} & b_{32} & \ddots & & \\ \vdots & \vdots & \ddots & 0 & \\ b_{s1} & b_{s2} & \cdots & b_{ss} & 0 \end{bmatrix}$	non stiff	serial
Type 2	$\begin{bmatrix} \lambda & & & & \\ b_{21} & \lambda & & & \\ b_{31} & b_{32} & \ddots & & \\ \vdots & \vdots & \ddots & \lambda & \\ b_{s1} & b_{s2} & \cdots & b_{ss} & \lambda \end{bmatrix}$	stiff	serial
Type 3	$\begin{bmatrix} 0 & & & & \\ 0 & 0 & & & \\ 0 & 0 & \ddots & & \\ \vdots & \vdots & \ddots & 0 & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$	non stiff	parallel
Type 4	$\begin{bmatrix} \lambda & & & & \\ & \lambda & & & \\ & & \ddots & & \\ & & & \lambda & \\ & & & & \lambda \end{bmatrix}$	stiff	parallel

Figure 1: Butcher's four DIMSIM types [11]

more significant in determining the suitability of a particular method in practice. Nevertheless, we have found the single-eigenvalue property helpful for the simplification it provides when searching for A-stable methods. Although their DIMSIMs are not SEMs, Butcher and Jackiewicz [12, 13, 14] have since been able to find A-stable methods by deriving tractable algebraic expressions for the eigenvalues of their stability matrices.

2.1.2 Application of the Single-Eigenvalue Property to Proving A- and L-Stability

The problem of determining A- and L-stability of a GLM simplifies considerably if the stability matrix has only one nonzero eigenvalue. If we let

$$\text{eig}(M(z)) = \{e_1(z), \ldots, e_r(z)\},$$

and suppose that

$$e_2(z) = \cdots = e_r(z) = 0 \quad \forall z \in \mathbb{C},$$

then

$$\text{tr}(M(z)) = \sum_{i=1}^{r} e_i(z) = e_1(z).$$

Now recall that

$$M(z) = A_2 + zB_2(I - zB_1)^{-1}A_1,$$

so that the trace is a rational function of z, say

$$e_1(z) = \frac{P(z)}{Q(z)}. \tag{15}$$

For a strictly diagonal method, where

$$B_1 = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_s \end{bmatrix},$$

it follows easily that

$$Q(z) = \prod_{i=1}^{s}(1 - \lambda_i z), \tag{16}$$

so that $e_1(z)$ is analytic and has no poles in $\mathbb{C}^-$ provided all the λ_i's have positive real parts. (For general B_1, (16) holds if $\lambda_1, \dots, \lambda_s$ are the eigenvalues of B_1 [16].) By the maximum principle, it suffices to check that $|e_1(iy)| \leq 1 \; \forall y \in \mathbb{R}$ to ensure A-stability. Now, if we define the *E-polynomial*

$$E(y) = |P(iy)|^2 - |Q(iy)|^2, \tag{17}$$

then

$$|e_1(iy)| \leq 1 \Leftrightarrow E(y) \leq 0,$$

and, assuming the polynomials P and Q have real coefficients, a sufficient condition for A-stability is that $\lambda_i \in \mathbb{C}^+$ for $i = 1, \dots, s$ and $E(y)$ has all non-positive coefficients. (This follows from the observation that, if P and Q are real polynomials, then $E(y)$ is an even polynomial, that is, the coefficient of each odd power of y in $E(y)$ is zero.) An A-stable SEM is also L-stable iff $|e_1(z)| \to 0$ as $|z| \to \infty$. From (15), this is equivalent to $\deg(P) < \deg(Q)$. Given the coefficient matrices of a candidate method, not only can these conditions be easily checked, but they are often simple enough to provide insight into what choices of the free parameters are required to ensure A- and L-stability.

2.2 Maximal Order of Single-Eigenvalue GLMs

Before beginning the construction of DIMSEMs, we develop an upper bound on the order they might attain. The following three theorems and two corollaries apply to a larger class of methods than DIMSEMs, in that B_1 need not be diagonal. They extend to single-eigenvalue GLMs the corresponding result of Jackson and Nørsett for Runge-Kutta methods [24, Theorem 2.1].

Theorem 1 *Let $e_1(z), \dots, e_r(z)$ be the eigenvalues of the stability matrix $M(z)$ of a GLM. Assume $e_1(z) = e^z + O(z^{p+1})$, and $e_2(z) = \cdots = e_r(z) = 0$ for all z. Then*

$$p \leq d + \delta + \sum_{i=1}^{k} r_i,$$

where B_1 has eigenvalues $\gamma_1, \dots, \gamma_k$ with no repetitions, r_i is the size of the largest Jordan block associated with γ_i, d is the number of distinct complex non-real eigenvalues of B_1, and δ is 1 if B_1 has at least one nonzero real eigenvalue and 0 otherwise.

Remark. We have not assumed B_1 is diagonal in the theorem above. Also, d is even if B_1 is real.

Proof. See [16, Theorem 1].

In several of the following theorems, we need to guarantee that the exact values $\hat{y}_n$ approximated by the external stages for the linear test problem $u' = \mu u$ are nonzero. One way to ensure this is to require that the true-value function satisfies $q(\phi) \neq 0$. This property is implied by the preconsistency condition of [5, 7] and is satisfied by any useful method, including both multistep- and Nordsieck-type GLMs.

Lemma 1 *Let $e_1(z), \ldots, e_r(z)$ be the eigenvalues of the stability matrix $M(z)$ of a GLM for which $q(\phi) \neq 0$. Assume the method has local discretisation error of order $q+1$ and $e_2(z) = \cdots = e_r(z) = 0$ for all z. Then $e_1(z) = e^z + O(z^{q+1})$.*

Proof. See [16, Lemma 1].

The previous lemma can now be used to bound the local order of the method.

Theorem 2 *Let $e_1(z), \ldots, e_r(z)$ be the eigenvalues of the stability matrix $M(z)$ of a GLM for which $q(\phi) \neq 0$. Assume $e_2(z) = \cdots = e_r(z) = 0$ for all z and $p \geq 0$ is the largest value such that $e_1(z) = e^z + O(z^{p+1})$. Then the order of the local discretisation error of the method is at most $p+1$.*

Proof. Follows immediately from Lemma 1. (See [16, Theorem 2].)

The next result follows immediately from Theorems 1 and 2.

Corollary 1 *Let $e_1(z), \ldots, e_r(z)$ be the eigenvalues of the stability matrix $M(z)$ of a GLM for which $q(\phi) \neq 0$ and assume $e_2(z) = \cdots = e_r(z) = 0$ for all z. Then the order of the local discretisation error of the GLM is at most $p+1$, where*

$$p = d + \delta + \sum_{i=1}^{k} r_i,$$

B_1 has eigenvalues $\gamma_1, \ldots, \gamma_k$ with no repetitions, r_i is the size of the largest Jordan block associated with γ_i, d is the number of distinct complex non-real eigenvalues of B_1, and δ is 1 if B_1 has at least one nonzero real eigenvalue and 0 otherwise.

We now state a theorem that bounds the order of the global error of a SEM.

Theorem 3 *Let $e_1(z), \ldots, e_r(z)$ be the eigenvalues of the stability matrix $M(z)$ of a GLM for which $q(\phi) \neq 0$. Assume $e_2(z) = \cdots = e_r(z) = 0$ for all z and $p \geq 0$ is the largest value such that $e_1(z) = e^z + O(z^{p+1})$. Then the order of the global error of the method is at most p.*

Proof. See [16, Theorem 3].

The next result follows immediately from Theorems 1 and 3.

Corollary 2 *Let $e_1(z), \ldots, e_r(z)$ be the eigenvalues of the stability matrix $M(z)$ of a GLM for which $q(\phi) \neq 0$ and assume $e_2(z) = \cdots = e_r(z) = 0$ for all z. Then the order of the global error of the GLM is at most p, where*

$$p = d + \delta + \sum_{i=1}^{k} r_i,$$

B_1 has eigenvalues $\gamma_1, \ldots, \gamma_k$ with no repetitions, r_i is the size of the largest Jordan block associated with γ_i, d is the number of distinct complex non-real eigenvalues of B_1, and δ is 1 if B_1 has at least one nonzero real eigenvalue and 0 otherwise.

Applying Corollaries 1 and 2 to DIMSEMs, for which $B_1 = \text{diag}(\lambda_i)$ with $\lambda_i \in \mathbb{R}$ for $i = 1, \dots, s$, we see that the orders of the local and global discretisation errors are at most $s+2$ and $s+1$, respectively, since $d = 0$, $\delta = 1$, each $r_i = 1$ and $k \le s$ imply $p \le s+1$. Moreover, the maximal values $s+2$ and $s+1$ occur only if all the λ_i's are distinct. It remains to be seen whether this order is attainable for general s.

These corollaries also show that high order, singly implicit DIMSEMs do not exist. In particular, the orders of the local and global discretisation errors of a DIMSEM with $B_1 = \lambda I$, where $\lambda \in \mathbb{R}$, are at most 3 and 2, respectively, since $d = 0$, $\delta = 1$, $k = 1$ and $r_1 = 1$ imply $p = 2$.

A consequence of these results is that we need to allow a general diagonal matrix B_1 in DIMSEMs. Our search for such methods began with the class of MRKs with $r = s$, and satisfying the simplifying assumptions $B(r)$ and $C(r-1)$. These methods have order at least r. It turns out, however, that such a DIMSEM with $r = 3$ cannot be A-stable, and in fact has a finite stability region [16, Section 2.3]. We look next at general DIMSEM-GLMs in search of a class of DIMSEMs with more promise for solving stiff problems.

2.3 Linking Stability and Order Conditions

To produce a method with $r-1$ zero eigenvalues, it is necessary to zero the coefficient functions $\beta_i(z)$ of w^i, $i = 0, \dots, r-2$, in the characteristic polynomial

$$\begin{aligned}\varphi(w,z) &= \det(wI - M(z)) \\ &= w^r - e_1(z)w^{r-1} - \beta_{r-2}(z)w^{r-2} - \cdots - \beta_1(z)w - \beta_0(z).\end{aligned}$$

As each $\beta_i(z)$ is a rational function of z, this amounts to zeroing the numerator polynomial, which involves setting all the coefficients of each polynomial to zero. These coefficients are complicated nonlinear functions of the coefficients of the method. We found solving these equations, together with the simplified order conditions, to be an intractable problem when done algebraically to facilitate later choice of the remaining free parameters for A-stability. The computation is eased somewhat by transforming the stability matrix in such a way that the simplified order conditions can be incorporated into it. Then the $r-1$ eigenvalues can be set to zero, with the assurance that a method of the desired order can be recovered later. To do this, we choose the simplifying assumptions $A(r-1)$, $B(r-1)$ and $C(r-1)$, because, from (13) and (14), $B(r-1)$ and $C(r-1)$ can be conveniently written in matrix form as follows.

$$B(r-1) \quad : \quad A_2Q + B_2W = QP, \tag{18}$$
$$C(r-1) \quad : \quad A_1Q + B_1W = V, \tag{19}$$

where

$$\begin{aligned}Q &= [q_0, \dots, q_{r-1}] \in \mathbb{R}^{r\times r}, \\ q_k &= q(\tau^k) = q(t) \quad \forall t \ni \rho(t) = k, \quad k = 0, \dots, r-1, \\ V_{ij} &= c_i^{j-1}, \quad V \in \mathbb{R}^{s\times r}, \\ W_{ij} &= (j-1)c_i^{j-2}, \quad W \in \mathbb{R}^{s\times r}, \\ P_{ij} &= \left\{ \begin{array}{cc} \begin{pmatrix} j-1 \\ i-1 \end{pmatrix}, & j \ge i \\ 0, & \text{otherwise} \end{array} \right\}, \quad P \in \mathbb{R}^{r\times r},\end{aligned}$$

and 0^0 is taken to be 1. In the following, we assume that Q is nonsingular. Let $\bar{B}_2 = Q^{-1}B_2$, and consider the transformed stability matrix

$$\bar{M}(z) \quad = \quad Q^{-1}M(z)Q$$

$$
\begin{aligned}
&= Q^{-1}\left(A_2 + zB_2(I - zB_1)^{-1}A_1\right)Q \\
&= Q^{-1}(QP - B_2W) + zQ^{-1}B_2(I - zB_1)^{-1}(V - B_1W) \\
&= P - \bar{B}_2W + z\bar{B}_2(I - zB_1)^{-1}(V - B_1W) \\
&= P + \bar{B}_2\left[z(I - zB_1)^{-1}(V - B_1W) - W\right] \\
&= P + \bar{B}_2\left[(I - zB_1)^{-1}\left(zV - zB_1W - (I - zB_1)W\right)\right] \\
&= P + \bar{B}_2(I - zB_1)^{-1}(zV - W).
\end{aligned} \tag{20}
$$

If we define

$$
U = \begin{bmatrix} 0 & 1 & & & \\ & \ddots & \ddots & & \\ & & & \ddots & r-1 \\ & & & & 0 \end{bmatrix},
$$

then $VU = W$ and

$$
\bar{M}(z) = P + \bar{B}_2(I - zB_1)^{-1}V(zI - U). \tag{21}
$$

The matrix $\bar{M}(z)$ has the same eigenvalues as $M(z)$, but, by incorporating the order conditions (18) and (19), the A_1 and A_2 matrices have been eliminated. The problem of finding a DIMSEM has now been reduced to

- picking values for the free parameters λ_i, c_i, and $\bar{B}_2$, so that $\bar{M}(z)$ has $r-1$ zero eigenvalues, and
- recovering the remaining method coefficients using $B_2 = Q\bar{B}_2$ and the order conditions

$$
A_1 = (V - B_1W)Q^{-1} \tag{22}
$$
$$
A_2 = (QP - B_2W)Q^{-1}. \tag{23}
$$

The resulting method (if it exists and is stable) has order at least $r-1$ and stage order at least $r-1$.

2.4 A Class of L-stable Diagonal Methods (LDMs)

We now describe an interesting class of methods which, although not DIMSEMs, are singly strictly diagonally implicit and L-stable. (In Butcher's classification, Fig. 2.1.1, they are type-4 DIMSIMs.)

If $\lambda_i = \lambda$ for $i = 1, \dots, s$, then the transformed stability matrix (21) simplifies to

$$
\bar{M}(z) = P + \frac{1}{1 - \lambda z}T(zI - U),
$$

where $T = \bar{B}_2V$. We choose T below and then recover $\bar{B}_2 = TV^{-1}$. This requires that $r = s$ and the c_i's are distinct so that V is nonsingular. If we let T be upper triangular, $\bar{M}(z)$ is also upper triangular, and so the eigenvalues are simply the diagonal elements

$$
e_i(z) = 1 + \frac{T_{ii}z}{1 - \lambda z}, \quad i = 1, \dots, r. \tag{24}
$$

For $\lambda \in \mathbb{R}^+$ and any choice of

$$
T_{ii} \in (0, 2\lambda), \quad i = 1, \dots, r, \tag{25}
$$

it follows that

$$
|e_i(z)| < 1 \quad \forall z \in \mathbb{C}^- \setminus \{0\}, \quad i = 1, \dots, r.
$$

However, $e_i(0) = 1$ for $i = 1, \ldots, r$, so the method will be A-stable if and only if $\bar{M}(0)$ is diagonalisable. If we choose

$$T_{ij} = \frac{1}{j}\left(\begin{array}{c} j \\ i-1 \end{array}\right), \quad i = 1, \ldots, r-1, \quad j = i, \ldots, r-1, \tag{26}$$

then $\bar{M}(0) = P - TU = I$ is diagonal. Moreover, since $T_{ii} = 1$ for $i = 1, \ldots, r-1$, (25) is satisfied if

$$\lambda > \max(1/2, T_{rr}/2).$$

Thus, we have found a class of A-stable, singly strictly diagonally implicit GLMs of order and stage order $r-1$ for any $r \geq 1$. The c_i's, λ, T_{rr}, and q are still free. If we choose $\lambda = T_{rr} = 1$, then (24) reduces to

$$e_i(z) = \frac{1}{1-z}, \quad i = 1, \ldots, r,$$

from which it follows immediately that $|e_i(z)| \to 0$ as $|z| \to \infty$. This L-stable class has the c_i's and q free.

Interestingly, recovering the original coefficient matrix A_2 using (23) gives

$$\begin{aligned} A_2 &= (QP - Q\bar{B}_2 W)Q^{-1} \\ &= (QP - QTV^{-1}VU)Q^{-1} \\ &= Q(P - TU)Q^{-1} \\ &= QIQ^{-1} \\ &= I. \end{aligned}$$

This is also the design choice made in the DIMSIMs of [11, 12, 13, 14], of which the type-4 variants are also singly strictly diagonally implicit. The DIMSIMs however have higher orders and the remaining coefficients are chosen differently.

2.5 A Class of DIMSEMs

This section contains the main contribution of this paper. We introduce two theorems — Theorems 4 and 5 — which define a class of DIMSEMs of order $r-1$, for any $r \geq 2$. The choice of method coefficients made in Theorem 4 arose as a generalisation of patterns observed in methods of orders 2–6, which were derived by solving the zero-eigenvalue equations in Maple. The computational feasibility of solving these equations depends on the simplification achieved by taking all the internal abscissae c_i's to be equal. Although this results in the Vandermonde matrix V being singular, this choice is acceptable, since our technique described in Section 2.3 for determining the coefficients of a GLM satisfying the zero-eigenvalue equations does not require V to be nonsingular.

We initially applied this somewhat unusual choice of parameters with all the c_i's equal to one, resulting in the creation of A-stable DIMSEMs up to order four, but greater success was obtained by setting all the c_i's to zero. In this case, the Vandermonde matrix V simplifies to

$$V = \left[\begin{array}{cccc} 1 & 0 & \cdots & 0 \\ \vdots & & \vdots & \\ 1 & 0 & \cdots & 0 \end{array}\right],$$

so that the quantity $V(zI - U)$ appearing in the transformed stability matrix

$$\bar{M}(z) = P + \bar{B}_2(I - zB_1)^{-1}V(zI - U)$$

becomes

$$V(zI-U)=\begin{bmatrix} z & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \\ z & -1 & 0 & \cdots & 0 \end{bmatrix}.$$

The transformed stability matrix itself becomes

$$\bar{M}(z)=P+\begin{bmatrix} za_1(z) & -a_1(z) & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \\ za_r(z) & -a_r(z) & 0 & \cdots & 0 \end{bmatrix}, \tag{27}$$

where

$$a_i(z)=\sum_{k=1}^{s}\frac{b_{ik}}{1-\lambda_k z}, \quad i=1,\ldots,r, \tag{28}$$

$b_{ik}=(\bar{B}_2)_{ik}$ and $B_1=\text{diag}(\lambda_1,\ldots,\lambda_s)$. This form of $\bar{M}(z)$ is simple enough to enable direct algebraic computation of the characteristic polynomial up to $r=5$ (corresponding to order $r-1=4$) using Maple. The advantage of this technique, however, is that relationships observed between elements of the $\bar{B}_2$ matrix of the resulting DIMSEMs can be generalised to permit the derivation of methods of higher order.

When solving the zero-eigenvalue equations using Maple, certain patterns were observed. When $s=r-1$, it appeared that there was always a unique solution which leaves the λ_i's free. There also appeared to be a direct recurrence expressing the rest of the $\bar{B}_2$ matrix in terms of its last row. These observations gave rise to Theorem 4, the proof of which relies on an expression for the characteristic polynomial of the transformed stability matrix given by Theorem 5.

Theorem 4 *Let* $b_{ij}=(\bar{B}_2)_{ij}$. *Let*

$$s=r-1\geq 1,$$

$$c_i=0, \quad i=1,\ldots,s,$$

$$\begin{aligned} b_{rj} &= \frac{1}{s!}\frac{\sum_{k=0}^{s-1}\frac{\theta_{kj}^{(s)}}{(s-k)!}}{\prod_{\substack{l=1\\l\neq j}}^{s}(\lambda_l-\lambda_j)} \\ &= \frac{1}{s!\lambda_j^{s-1}p_j(\frac{1}{\lambda_j})}\sum_{k=0}^{s-1}\frac{\theta_{kj}^{(s)}}{(s-k)!}, \end{aligned}$$

where

$$\theta_{kj}^{(s)} \textit{ is the coefficient of } x^k \textit{ in } p_j(x)$$

and

$$p_j(x)=\prod_{\substack{l=1\\l\neq j}}^{s}(\lambda_l x-1).$$

Let

$$b_{ij}=T_{ij}(\lambda)b_{rj}, \quad i=1,\ldots,r-1,$$

where $\lambda=(\lambda_1,\ldots,\lambda_s)$ *with* λ_i, $i=1,\ldots,s$, *distinct and nonzero,*

$$T_{ij}(\lambda)=\frac{1}{(i-1)!}\sum_{l=0}^{r-i}\left[(i-1+l)!\gamma_{s-i-l+1}^{(s)}\lambda_j^l\right],$$

and

$$\gamma_k^{(s)} = \begin{cases} \sum_{1\le i_1<\cdots<i_k\le s} i_1\cdots i_k, & k\ge 1,\\ 1, & k=0.\end{cases}$$

Then the transformed stability matrix $\bar{M}(z)$ has $r-1$ zero eigenvalues for all $z\neq 1/\lambda_i$, $i=1,\ldots,s$.

Remark. The requirement that $z\neq 1/\lambda_i$ for $i=1,\ldots,s$ is a necessary restriction, since the factor $(I-zB_1)$ that occurs in both $M(z)$ and $\bar{M}(z)$ is singular if and only if $z=1/\lambda_i$ for some $i=1,\ldots,s$. Note that, if we choose $\lambda_i>0$ for $i=1,\ldots,s$, then $M(z)$ and $\bar{M}(z)$ are well defined and the single eigenvalue property holds for all $z\in\mathbb{C}^-$.

Proof. See [16, Theorem 4].

Theorem 5 *Assume $z\neq 1/\lambda_i$ for $i=1,\ldots,s$ and let*

$$\phi_r(w,z)=\mathit{det}(wI-\bar{M}(z)),$$

where $\bar{M}(z)$ is defined in (21). Let

$$s=r-1\ge 1$$

and

$$c_i=0,\quad i=1,\ldots,s.$$

Then

$$\phi_r(w,z)=\sum_{k=0}^{r}\phi_k^{(r)}(z)w^k,$$

where

$$\phi_k^{(r)}(z)=\begin{cases}\delta_k^{(s)}+\sum_{i=1}^{r}\left[(\alpha_{k,i}^{(s)}+\beta_{k,i}^{(s)}z)a_i(z)\right], & k=0,\ldots,r-1,\\ 1, & k=r,\end{cases}$$

$$\delta_k^{(s)}=(-1)^{s+1-k}\begin{pmatrix} s+1\\ k\end{pmatrix},$$

$$\alpha_{k,i}^{(s)}=\begin{cases}0, & i=1,\\ -(i-1)\beta_{k,i-1}^{(s)}, & i\ge 2,\end{cases}$$

$$\beta_{k,i}^{(s)}=\begin{cases}(-1)^{s+1-k}\begin{pmatrix} s\\ k\end{pmatrix}, & i=1,\\ (-1)^{s-k}\left[\begin{pmatrix} s-1\\ k\end{pmatrix}+\sum_{l=1}^{s-k-1}(-1)^l\begin{pmatrix} s-l-1\\ k\end{pmatrix}S_l^{i-1}\right], & i\ge 2,\end{cases}$$

$$a_i(z)=\sum_{k=1}^{s}\frac{b_{ik}}{1-\lambda_k z},$$

and

$$S_l^n=\begin{cases}\sum_{1\le i_1<\cdots<i_l\le n-1}\begin{pmatrix} i_2\\ i_1\end{pmatrix}\begin{pmatrix} i_3\\ i_2\end{pmatrix}\cdots\begin{pmatrix} i_l\\ i_{l-1}\end{pmatrix}\begin{pmatrix} n\\ i_l\end{pmatrix}, & l\ge 1,\\ 1, & l=0.\end{cases}$$

Remark. Again note that the requirement that $z \neq 1/\lambda_i$ for $i = 1, \ldots, s$ is a necessary restriction, since the factor $(I - zB_1)$ that occurs in both $M(z)$ and $\bar{M}(z)$ is singular if and only if $z = 1/\lambda_i$ for some $i = 1, \ldots, s$. If we choose $\lambda_i > 0$ for $i = 1, \ldots, s$, then $M(z)$ and $\bar{M}(z)$ are well defined and the single-eigenvalue property obtained by Theorem 4 holds for all $z \in \mathbb{C}^-$.

Proof. See [16, Theorem 5].

As mentioned in Section 2.3, the characteristic polynomial associated with a DIMSEM has the form

$$\begin{aligned} \varphi(w, z) &= \det(wI - M(z)) \\ &= w^r - e_1(z)w^{r-1} - \beta_{r-2}(z)w^{r-2} - \cdots - \beta_1(z)w - \beta_0(z), \end{aligned} \tag{29}$$

where

$$\beta_i(z) = \frac{P_i(z)}{Q_i(z)}, \quad i = 0, \ldots, r-2,$$

P_i and Q_i are polynomials and $e_1(z)$ is a rational function of z. The requirement that $M(z)$ have $r-1$ zero eigenvalues is equivalent to $\beta_0(z) = \cdots = \beta_{r-2}(z) = 0$ for all z, which in turn is equivalent to $P_0(z) = \cdots = P_{r-2}(z) = 0$ for all z, which in turn is equivalent to having all the coefficients of $P_i(z)$ equal to zero for $i = 0, \ldots, r-2$. This last condition is what we call the zero-eigenvalue equations. Then $e_1(z)$ is the remaining nonzero eigenvalue of $M(z)$, so it governs the stability of the method.

A-stability is easy to check, even without computing $e_1(z)$ from the characteristic polynomial. This is important, because algebraic computation of the characteristic polynomial is too complicated to be feasible for high orders. From (27), (28), and the zero-eigenvalue property, it follows that

$$\begin{aligned} e_1(z) &= \operatorname{tr}(\bar{M}(z)) \\ &= r + za_1(z) - a_2(z) \\ &= r + \sum_{k=1}^{s} \frac{zb_{1k} - b_{2k}}{1 - \lambda_k z}. \end{aligned}$$

This can be expressed in the form

$$e_1(z) = \frac{P(z)}{Q(z)}, \tag{30}$$

and the associated E-polynomial (17) constructed. Recall that a sufficient condition for A-stability is that the coefficients of the E-polynomial are negative and the λ_i's are positive. It is sometimes possible to solve these inequalities directly for the remaining free parameters in the low-order methods. The procedure followed to construct L-stable DIMSEMs up to order 6 for numerical testing is described in the next section.

2.6 A Class of L-Stable DIMSEMs

Until now, it has not been necessary to specify the true-value function q. The only restrictions on q have been the simplifying assumption $A(r-1)$, which states that $q(t)$ depends only on the order of the tree t, up to order $r-1$, and the requirement that the matrix Q be nonsingular. Both of the common choices of true-value function, based on Nordsieck and multistep implementations, satisfy these requirements. Because our methods satisfy $A(r-1)$, we will write q_k for $q(t)$, where t is any tree of order $k \leq r-1$. Also, let e_k be the vector in $\mathbb{R}^r$ with 1 as the k-th component and 0 elsewhere. Then in Nordsieck-type methods (1), q is given by

$$q_k = e_{k+1}, \quad k = 0, \ldots, r-1,$$

so that

$$Q = I_r \in \mathbb{R}^{r\times r}.$$

On the other hand, for multistep GLMs,

$$q_k = d^k, \quad k = 0, \ldots, r-1, \tag{31}$$

where $d \in \mathbb{R}^r$ is the vector of external abscissae, and d^k is taken componentwise with $0^0 = 1$. The corresponding matrix Q is

$$Q = [d^0, d^1, \ldots, d^{r-1}] \in \mathbb{R}^{r\times r}.$$

This corresponds to the external stages approximating the solution at various points,

$$y_{n,i} \approx u(x_n + d_i h), \quad i = 1, \ldots, r.$$

In deriving specific methods from our classes for testing purposes, we chose the multistep-type true-value function, and used the same external abscissae

$$d_i = i - r, \quad i = 1, \ldots, r,$$

as used in MRKs. Unlike MRKs, however, our A_2 and B_2 are general matrices, and not the restricted forms (3) and (4).

The first step in the creation of the methods was the algebraic solution of the zero-eigenvalue equations using the techniques of the previous section. This produced classes of DIMSEMs of order $p = 2, \ldots, 6$, $p = r - 1 = s$, with the λ_i's as free parameters. Then the E-polynomial was formed and A- and L-stability investigated.

2.6.1 Choice of Free Parameters for the DIMSEMs

The DIMSEMs of order $s = r - 1$ defined by Theorem 4 have free parameters $\lambda_1, \ldots, \lambda_s$, where $B_1 = \text{diag}(\lambda_1, \ldots, \lambda_s)$. The values of $\lambda_1, \ldots, \lambda_s$ were chosen to make the methods L-stable and to achieve a good compromise between small error coefficients and reasonably sized entries in the coefficient matrices A_1, A_2, B_1 and B_2.

The error coefficient function $e(t)$ defined in (11) is a vector-valued function, the i-th component of which gives the local discretisation error associated with the tree t for the i-th external stage. For optimisation purposes, we only used the quantities T_1, T_2 and T_3, where

$$T_k = \frac{1}{(p+k)!} \max \left\{ \| \, e(t) \, \|_\infty \colon \rho(t) = p + k \right\}, \quad k = 1, 2, 3, \ldots$$

and p is the order of the method. These are the principal, secondary, tertiary etc. normalised, maximum-norm error coefficients associated with the method. We wrote a Maple program, incorporating a data structure for rooted trees, to compute algebraic expressions for the error coefficients. All computation involved in the choice of the free parameters was done algebraically using Maple.

The first step in the procedure was to determine λ_s in terms of $\lambda_1, \ldots, \lambda_{s-1}$ so that $\deg(P) < \deg(Q)$, where the one nonzero eigenvalue of $M(z)$ is $e_1(z) = P(z)/Q(z)$. This ensures that, if the method is A-stable, it will also be L-stable. The next step was to heuristically choose values for $\lambda_3, \ldots, \lambda_{s-1}$, leaving at most λ_1 and λ_2 still to be selected. To achieve A-stability, the region where all the coefficients of the E-polynomial (17) are negative, and the region where $\lambda_i > 0$, $i = 1, \ldots, s$, were plotted. The intersection of these regions indicates choices of the free parameters which result in an A-stable method, and hence an

order	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	T_1	T_2	T_3	T_4	T_5	T_6	maxcf
2	.49	$\frac{1}{51}$					.33	.33	.26	.12	.045	.020	2
3	.25	.43	$\frac{79}{87}$				.44	.51	.44	.27	.14	.13	5
4	1.3	.27	1.0	$\frac{1337}{2710}$			.46	.87	.95	.80	.92		8
5	.37	.29	.25	.50	$\frac{654}{4981}$		.50	1.1	1.4	1.3			85
6	.71	.095	.25	.50	.75	$\frac{5086}{15879}$	.53	1.4	2.1				33

Table 1: Choice of Free Parameters and Error Coefficients for the DIMSEM Class.

L-stable one, because of the earlier choice of λ_s. Then the error coefficients T_1, T_2 and T_3, and the maxcf function were plotted, where

$$\begin{aligned} \text{maxcf} &= \max\{\langle A_1\rangle, \langle B_1\rangle, \langle A_2\rangle, \langle B_2\rangle\}, \\ \langle A\rangle &= \{\max|A_{ij}| : i, j\}. \end{aligned} \tag{32}$$

Finally, values for the free parameters resulting in L-stable methods while keeping the T_i's and maxcf small were found heuristically, with the aid of 3-dimensional plots and numerical searching.

The chosen values for the free parameters, along with the resulting error coefficients and maxcf (32) are given in Table 1. The λ_i's shown are exact, while the T_i's and maxcf are rounded. (Some of the T_i's are omitted because they were too expensive to compute.)

The error coefficients of the L-stable DIMSEMs were then compared with those of two other classes of A-stable methods, the MRKs of Burrage and Chipman [6] and the Gauss and Radau-IA IRKs (see, for example, [10]). The MRKs are implicit with a lower-triangular B_1 matrix, while the Gauss/Radau formulas are fully implicit Runge-Kutta methods. The Gauss formulas have order $2s$ and are used for orders 2, 4 and 6, while the Radau-IA formulas have order $2s-1$ and are used for orders 3 and 5. In Burrage and Chipman's methods, the choices of the free parameters used were those given on page 410 of their paper, which were used in their numerical testing. The results are shown in Table 2. Most of the error coefficients of the DIMSEMs are an improvement over those of the MRKs, and the ones which are not are only slightly worse. The improvements in T_2 and T_3 are quite significant. Both the DIMSEMs and MRKs have much larger error coefficients than the corresponding Gauss or Radau IRKs, however.

Table 3 compares the error coefficients of the DIMSEMs and the Backward Differentiation Formulas (BDFs) [19, p. 311]. (These formulas are used by the integrator LSODE, to which we compare the performance of the DIMSEMs in Section 3.) The BDF error coefficients are generally smaller, though comparable, to those of the DIMSEMs. The BDFs of order 3 and up are not A-stable however.

2.7 Choice of Free Parameters for the LDMs

LDMs of order $r-1$ with $r=s$ are described in Section 2.4. Like the DIMSEMs derived in the previous section, the specific LDMs given in this section are based on a multivalue-type true-value function. The free parameters λ and T_{rr} are both set to 1, giving methods which are L-stable for all values of $c_1, \ldots, c_s$. It turns out that the principal error coefficients, and hence T_1, are fixed for each method, independent of the free parameters. The selection of the c_i's was performed in a manner similar to that of the λ_i's for the DIMSEMs (Section 2.6.1), except that only T_2, T_3, and the maxcf function (32) had to be considered due to the fixed T_1 and inherent L-stability.

The results of the optimisation are shown in Table 4. The values given for the c_i's are exact, while those for the T_i's are rounded. (Again, some of the T_i's are omitted because they were too expensive to compute.)

	DIMSEMs					Burrage's MRKs					Gauss/Radau-IA IRKs				
order	r	s	T_1	T_2	T_3	r	s	T_1	T_2	T_3	r	s	T_1	T_2	T_3
2	3	2	.33	.33	.26	2	1	.33	.33	.23	1	1	.083	.17	.18
3	4	3	.44	.51	.44	2	2	.37	1.1	2.1	1	2	.014	.031	.037
4	5	4	.46	.87	.95	3	2	.45	1.4	2.8	1	2	1.3E-3	2.8E-3	2.9E-3
5	6	5	.50	1.1	1.4	4	3	1.4	9.7	39	1	3	1.4E-4	3.0E-4	3.3E-4
6	7	6	.53	1.4	2.1	5	4	.49	5.3	30	1	3	9.9E-6	2.0E-5	7.9E-5

Table 2: Comparison of Error Coefficients of DIMSEMs, Burrage and Chipman's MRKs, and Gauss/Radau-IA IRKs.

	DIMSEMs					BDFs				
order	r	s	T_1	T_2	T_3	r	s	T_1	T_2	T_3
2	3	2	.33	.33	.26	2	1	.22	.43	.47
3	4	3	.44	.51	.44	3	1	.14	.32	.41
4	5	4	.46	.87	.95	4	1	.096	.27	.41
5	6	5	.50	1.1	1.4	5	1	.073	.24	.42
6	7	6	.53	1.4	2.1	6	1	.058	.22	.44

Table 3: Comparison of Error Coefficients of DIMSEMs and BDFs.

Observe that the error coefficients of these LDMs are large in comparison to the other methods considered. The size of the coefficient matrix entries, maxcf, also seems to increase more quickly with the order of the method for LDMs than for DIMSEMs. This might limit the maximum order of practical methods of this class, unless a way is discovered to reduce this coefficient growth. Nevertheless, the LDM methods found perform well in numerical testing [16].

In the interest of reducing the error coefficients, the c_i's were allowed to range outside the usual interval $[0, 1]$. This is a disadvantage in a practical implementation, since problems with singularities outside the interval of integration may be encountered.

order	c_1	c_2	c_3	c_4	c_5	c_6	c_7	T_1	T_2	T_3	T_4	T_5	T_6	maxcf
2	0	1	-1.6					.83	.63	.49	.43	.37	.39	2
3	0	1	-2.26	-1.2				1.8	.92	1.36	.96	.65	.57	5
4	0	1	-1	-2.85	-3.81			4.2	5.6	5.3	3.7	2.6		10
5	0	1	-1	2	-5.67	-5.7		10	15	19	15			36
6	0	1	-1	2	-2	-6.66	-3	25	50	68				224

Table 4: Choice of Free Parameters and Error Coefficients for the LDM Class.

3 Variable-Stepsize Implementation and Testing

A variable-stepsize implementation of our methods is desirable to permit performance testing under realistic conditions, and to allow comparison with existing methods. This section develops the components of the variable-stepsize implementation, and gives test results on sequential and parallel machines.

3.1 Stepsize Change Scheme

We have until now assumed a fixed-stepsize formulation for our GLMs, that is,

$$\begin{aligned} Y_n &= A_1 y_n + h_n B_1 F(Y_n), \\ y_{n+1} &= A_2 y_n + h_n B_2 F(Y_n), \end{aligned} \tag{33}$$

with $h_n = h$ for all n. If we desire h_n to vary, the formula (33) no longer directly applies, as the order conditions were derived under the assumption of a fixed stepsize. Rather than re-deriving the coefficient matrices to depend on a variable stepsize, we will use an interpolant to adjust the external stage vector prior to taking a step with a new stepsize. Assuming a multistep-type true value function, y_n, produced using stepsize h_{n-1}, satisfies

$$y_{n,i} \approx y(x_n + d_i h_{n-1}), \quad i = 1, \ldots, r. \tag{34}$$

However, to compute y_{n+1} using (33), we require that

$$y_{n,i} \approx y(x_n + d_i h_n), \quad i = 1, \ldots, r.$$

We therefore interpolate the data values associated with y_n, evaluate this interpolant at $x_n + d_i h_n$, $i = 1, \ldots, r$, producing a new external stage vector $\bar{y}_n$ satisfying

$$\bar{y}_{n,i} \approx y(x_n + d_i h_n), \quad i = 1, \ldots, r, \tag{35}$$

and then replace y_n with $\bar{y}_n$ in (33). (A corresponding transformation exists for Nordsieck-type GLMs.) It remains to specify the interpolant.

Since the DIMSEMs are locally order r, we derive a polynomial interpolant of degree $r-1$, hence of order r. Given y_n satisfying (34), we define a polynomial p of degree $r-1$ such that

$$p(x_n + d_i h_{n-1}) = y_{n,i}, \quad i = 1, \ldots, r.$$

A useful representation of p is

$$p(x_n + \tau h_{n-1}) = \sum_{i=1}^{r} l_i(x_n + \tau h_{n-1}) y_{n,i},$$

where l_i, $i = 1, \ldots, r$, are the Lagrange basis functions

$$\begin{aligned} l_i(x_n + \tau h_{n-1}) &= \frac{\prod_{j \neq i}(x_n + \tau h_{n-1} - (x_n + d_j h_{n-1}))}{\prod_{j \neq i}(x_n + d_i h_{n-1} - (x_n + d_j h_{n-1}))} \\ &= \frac{\prod_{j \neq i}(\tau - d_j)}{\prod_{j \neq i}(d_i - d_j)}. \end{aligned}$$

Then

$$\bar{y}_{n,i} = p(x_n + \tau_i h_{n-1})$$

satisfies (35) for

$$\tau_i = \omega_n d_i, \quad d_i h_n = \tau_i h_{n-1}, \text{ and } \omega_n = \frac{h_n}{h_{n-1}}.$$

Therefore we can write our variable-stepsize GLM as

$$\bar{y}_n = T(\omega_n) y_n, \tag{36}$$
$$Y_n = A_1 \bar{y}_n + h_n B_1 F(Y_n), \tag{37}$$
$$y_{n+1} = A_2 \bar{y}_n + h_n B_2 F(Y_n), \tag{38}$$

where

$$T_{ij}(\omega_n) = \frac{\prod_{k \neq j}(\omega_n d_i - d_k)}{\prod_{k \neq j}(d_j - d_k)}. \tag{39}$$

3.2 Stability for Variable-Stepsize Methods

It is possible for arbitrary, frequent stepsize changes to make a method asymptotically unstable, even though the underlying formula is stable in fixed-stepsize mode. It is possible to prove a result guaranteeing stability, however, provided the stepsize is not changed too often.

Theorem 6 *Suppose the problem satisfies $|f(u)| \leq L|u|$ for all u, the variable-stepsize DIMSEM (36-38) has order at least one and the stepsize-change ratio $\omega_n = h_n/h_{n-1}$ is bounded above. Then the method is asymptotically stable provided at least $r-1$ steps are taken with the same stepsize before each stepsize change.*

Remark. The uniform bound on f is imposed to simplify the proof, but it is quite restrictive, excluding for example the stable problem $u' = -u^3$. A similar assumption (uniform Lipschitz continuity) is made by Skeel [25] when proving a corresponding result (Lemma 3.5) for fixed-stepsize methods. However, as pointed out in [19, p. 395], it is only necessary to assume the condition in a neighbourhood of the solution, in which case the assumption becomes reasonable. The cited references, as well as our proof, do not deal with this rigorously though.

Remark. It is not necessary to bound ω_n away from zero.

Proof. See [16, Theorem 6].

3.3 Local Error Estimators

So far we have developed a means to allow the method to vary its stepsize, assuming it knows what stepsize to use. This choice is made by a stepsize controller, based on feedback from a local error estimator, the details of which we consider next.

The construction of error estimators for DIMSEMs was more difficult than we expected. Although we found estimators that enabled us to conduct variable-stepsize testing, their robustness is not ideal and more work remains to be done in this area.

3.3.1 DIMSEM Principal Error Coefficients

It turns out that, for a DIMSEM, the principal term of the B-series expansion of the local discretisation error is independent of the particular tree. This result, which is formalised in the following theorem, reduces the task of estimating the local discretisation error to estimating the r-th derivative of the solution.

Theorem 7 *Consider a multistep-type GLM with order and stage order p, satisfying simplifying assumption $A(p+1)$. Let $e(t)$ be the vector of local discretisation error coefficients (11) for the tree t. Then $e(t)$ is the same for all trees t of order $p+1$.*

Remark. In particular, our DIMSEMs of order p satisfy the conditions of the theorem.

Proof. See [16, Theorem 7].

3.3.2 Difference-Type Local Error Estimate

Theorem 7 provides the basis for one type of local error estimate. The local discretisation error vector l_n satisfies

$$\begin{aligned} l_n &= B(e, u(x_n)) \\ &= \sum_{t \in T} e(t) \frac{h^{\rho(t)}}{\rho(t)!} D(t)(u(x_n)). \end{aligned}$$

Consider a DIMSEM with local discretisation error of order $p+1$. Then $e(t) = 0$ for all trees with $\rho(t) \le p$. However, by Theorem 7, there is a vector e_{p+1} satisfying $e(t) = e_{p+1}$ for all trees t of order $p+1$. Thus

$$\begin{aligned} l_n &= \sum_{\rho(t) \ge p+1} e(t) \frac{h^{\rho(t)}}{\rho(t)!} D(t)(u(x_n)) \\ &= \sum_{\rho(t) = p+1} e(t) \frac{h^{\rho(t)}}{\rho(t)!} D(t)(u(x_n)) + O(h^{p+2}) \\ &= e_{p+1} \frac{h^{p+1}}{(p+1)!} \sum_{\rho(t)=p+1} D(t)(u(x_n)) + O(h^{p+2}) \\ &= e_{p+1} \frac{h^{p+1}}{(p+1)!} u^{(p+1)}(x_n) + O(h^{p+2}). \end{aligned} \tag{40}$$

Since e_{p+1} can be computed for any particular method using the results mentioned in Section 1.3, (40) gives an estimate of the local discretisation error, provided that $u^{(p+1)}(x_n)$ can be estimated. This may be done using a difference approximation, which we consider next.

The k-th order difference $\Delta^k u_n$ is defined for the function u by

$$\Delta^k u_n = \sum_{i=0}^{k} (-1)^i \left(\begin{array}{c} k \\ i \end{array} \right) u(x_{n+k-i}), \tag{41}$$

where $x_j = x_0 + jh$, $j = n, \ldots, n+k$, and h is some fixed spacing. If $u^{(k)}(x)$ exists and is continuous for $x \in [x_n, x_{n+k}]$, then

$$\Delta^k u_n = h^k u^{(k)}(\xi), \quad \xi \in [x_n, x_{n+k}],$$

and so $\Delta^k u_n$ can be used to obtain an approximation to $u^{(k)}(x_n)$. Suppose we have just completed a step, generating y_{n+1} from $\bar{y}_n$ using (38). For our local error estimate (40) we require an estimate of $u^{(r)}(x_{n+1})$. In order to apply (41), $r+1$ data points are needed, but y_{n+1} contains only r values. We therefore compute two difference estimates,

$$h_n^{r-1} u^{(r-1)}(x_{n+1}) \approx D_n^{(r-1)} = \sum_{i=1}^{r} (-1)^i \left(\begin{array}{c} r-1 \\ i \end{array} \right) y_{n+1, r+1-i} \tag{42}$$

and

$$h_n^{r-1}u^{(r-1)}(x_n) \approx D_{n-1}^{(r-1)} = \sum_{i=1}^{r}(-1)^i \left(\begin{array}{c} r-1 \\ i \end{array} \right) \bar{y}_{n,r+1-i}. \tag{43}$$

Then we subtract them to give

$$h_n^r u^{(r)}(x_{n+1}) \approx D_n^{(r)} = \sum_{i=1}^{r}(-1)^i \left(\begin{array}{c} r-1 \\ i \end{array} \right) (y_{n+1,r+1-i} - \bar{y}_{n,r+1-i}), \tag{44}$$

and therefore the local error estimate

$$l_n \approx \frac{e_r}{r!} \sum_{i=1}^{r}(-1)^i \left(\begin{array}{c} r-1 \\ i \end{array} \right) (y_{n+1,r+1-i} - \bar{y}_{n,r+1-i}). \tag{45}$$

The local error estimator (45) can be extended to provide estimates for a hypothetical step taken by DIMSEMs one order higher and one order lower than the current method. Although a variable-order implementation based on these estimates appeared viable, preliminary testing [16] exposed reliability problems with the higher-order estimate, especially for the higher-order methods. For this reason it was decided to postpone a variable-order implementation until more attention can be given to local error estimation.

3.4 Sequential Variable-Stepsize Testing

The variable-stepsize DIMSEM implementation was initially tested on a sequential machine to verify that it was working correctly and to determine whether there was any potential for performance gains on a parallel machine. LSODE [23] with the BDF option was chosen as a good-quality integrator for comparison purposes. Since LSODE is a variable-order code and our DIMSEM implementation is fixed order, the LSODE code was modified to accept an order parameter, which prevents order change after the initial startup process had reached the desired order. We refer to the order-p fixed-order modification of LSODE as LSODEp. The scaled stiff DETEST [18] test problem set was used. An efficiency comparison was made on the basis of the number of steps used to achieve a given endpoint global error. (Even though monitoring the maximum global error might be more appropriate for stiff problems, the endpoint global error is easier to compute. DETEST provides precomputed true solutions for the endpoint, whereas to compute maximum global error, a high-accuracy integration of the problem would have to be performed with a reliable method. Despite the use of endpoint global error, limiting accuracy being reached by the end of the integration was not a problem in the testing.) The number of steps taken is a reasonable quantity to compare directly between DIMSEMp and LSODEp, because a BDF formula requires one new function evaluation per nonlinear solver iteration on each step, while a DIMSEM requires s, all of which can be done in parallel. For a DIMSEM to have a chance of being competitive with LSODE, it must therefore take fewer steps for the same accuracy, assuming that the same number of nonlinear solver iterations are taken per step by the two methods. (The DIMSEMs might be expected to require fewer iterations, as $y_{n,r}$ provides a good initial guess for each $Y_{n,i}$, $i = 1, \ldots, s$.) Then, in addition, the parallel speedup must be sufficient and the heuristics and number of iterations per step comparable, points which are considered in the next section.

The problems were solved with absolute local error tolerances of 10^{-k}, $k = 4, \ldots, 13$, and the number of steps taken by each method to achieve various endpoint global errors $||y_{N,r} - u(x_N)||$ was compared [16, Figs. 3.2–3.5]. The results are summarised in Table 5, which shows the DIMSEMp's to take fewer steps than the corresponding LSODEp's for most of the problems and accuracies. The next section investigates whether this potential translates into better real-time performance when the DIMSEMs are run in parallel.

order	DIMSEM better	LSODE better	% DIMSEM better
2	65	33	66
3	88	28	76
4	93	32	74
5	116	18	87

Table 5: Number of Scaled Stiff DETEST Problem Runs for which DIMSEMp Took Fewer Steps Than LSODEp and Vice Versa, for the Same Accuracy

3.5 Variable-Stepsize Implementation Details

This section contains details of the final variable-stepsize fixed-order DIMSEM implementation.

The starting values $y_{0,i}$, $i = 1, \ldots, r$, were computed by (variable-order) LSODE to the same tolerance specified for the DIMSEM. $y_{0,1}$ was arranged to coincide with the initial condition, and LSODE integrated successively from $y_{0,i}$ to $y_{0,i+1}$, for $i = 1, \ldots, r-1$. For a sufficiently lengthy integration, the cost of the starting method is small compared to the overall cost, so the effect on performance is small. In a more realistic implementation, a scheme similar to that used by LSODE could be adopted. For example, take $y_{0,1}$ from the initial condition, compute $y_{0,2}$ and $y_{0,3}$ using a BDF or Runge-Kutta method, and then compute y_1 with DIMSEM2, y_2 with DIMSEM3, etc. This would allow the parallel DIMSEMs to be used sooner.

To satisfy the requirement, employed to assure stability, that $r-1$ steps be taken with the same stepsize before a stepsize change is allowed, it is not necessarily possible simply to re-take a failed step. Instead, a failed step causes the method to back up to the most recent point where a stepsize change would be allowed under the stability requirement. Although this strict approach is not necessary for all problems, it was not clear how to determine exactly when it is necessary, and so it was used at all times in the test code. Failed steps therefore incur a higher cost than in methods that can re-take only the step that failed. Although this is not obvious from the test results reported in the next section, reduced performance of the prototype implementation can be expected on integrations with a large proportion of failed steps.

The convergence criterion for the Newton iteration was that the maximum norm of the update vector be less than the local error tolerance. Because this criterion would often cause two iterations to be taken on linear problems (which are solved exactly in one Newton iteration, up to roundoff), the method was told if the problem is linear, in which case it was made to use exactly one iteration. LSODE uses a more sophisticated convergence criterion which almost always terminates after one corrector iteration on linear problems, without a priori knowledge of linearity.

The solution to the problem at the end of the integration was taken as $y_{n,r}$, the most advanced component of the external stage vector.

The heuristics used in the variable-stepsize DIMSEM code were adopted from VODE [2] for convenience. (In some cases, the comments in the VODE code differ slightly from the FORTRAN statements; in such cases we used the FORTRAN statements.) Although it is possible that some performance gain might result from tuning the parameters to specifically suit the DIMSEM code, this was not investigated, and the VODE parameters were used directly. The DIMSEM heuristics are summarised below.

- Minimum stepsize-change ratio = 0.1

- $$\text{Maximum stepsize-change ratio} = \begin{cases} 10^4 & \text{on 1st step} \\ 10 & \text{on 2nd and later accepted steps} \\ 1 & \text{on 1st retry of failed step} \\ 0.2 & \text{on 2nd retry of failed step} \end{cases}$$

- On 2nd-7th retry of a failed step, use $h \leftarrow h/10$. Give up after 7 retries.
- Declare Newton convergence failure after 3 failed iterations.
- On Newton convergence failure, use $h \leftarrow h/4$ and retry step. Give up after 10 retries.
- Increase h only if it increases by a factor of at least 1.5.
- Update and factorise Newton matrix:

  – on 1st step,
  – after Newton failure,
  – at least every 20 steps,
  – if stepsize has changed by $>$ 30% since last update.

- Update Jacobian if

  (Newton failure on this step
  OR Jacobian > 50 steps old)
  AND (No Newton failure with old Jacobian on this step
  OR stepsize has changed < 20 % since last update of Newton matrix)

- If Jacobian evaluation or Newton matrix factorisation is done on a step, it is done only before first Newton iteration, and not after each iteration.

3.6 Parallel Performance Testing

3.6.1 Parallel Machine

The parallel testing was done on a Silicon Graphics Challenge L computer, with eight 150 MHz R4400 processors, 512 MB of memory, and running IRIX 5.3. The programs were written in FORTRAN 77, and used the native SGI parallel directives. This machine was chosen for parallel performance testing because of its simple, shared-memory architecture, and consequent freedom from performance-distorting architectural features such as non-homogeneous communication paths and memory access times. The only consideration needed in this regard was to separate simultaneously-written memory by at least 64 bytes, in order to avoid cache conflicts. Although it was not possible to ensure exclusive access to the processors during performance testing, the tests were run when the system load was near zero, giving sufficiently consistent results. Timing was done with the FORTRAN function dtime, based on a clock with microsecond resolution.

Although the primary source of parallelism is in the simultaneous Newton iterations, the overhead incurred in starting parallel process was found to be sufficiently small that the various smaller matrix-vector operations, for example, the computation of the external stages, could also be effectively parallelised.

3.6.2 Test Problem

We desired a large, non-linear, real-world test problem with expensive function and Jacobian evaluations, for speedup can be expected to suffer if too little time is spent in the major parallel section of the program (see (50)). For this reason, a parallel method such as our DIMSEM implementation is only useful on sufficiently expensive problems. The problem chosen was Problem #8 in [1], an expression of the graph-theoretic travelling salesman problem as an ODE via a neural-network approach. The ODE is

$$\begin{aligned} u'_{ij}(t) &= -\frac{1}{\tau}u_{ij}(t) - A\sum_{\substack{k=1\\k\neq j}}^{n} V_{ik}(t) - B\sum_{\substack{k=1\\k\neq i}}^{n} V_{kj}(t) - C\left(\sum_{k=1}^{n}\sum_{l=1}^{n} V_{kl}(t) - p\right) \\ &\quad -D\sum_{k=1}^{n} d_{ik}(V_{k,j+1}(t) + V_{k,j-1}(t)), \quad i,j = 1,\ldots,n, \end{aligned} \tag{46}$$

where d_{ik} is the distance between cities i and k (placed equidistantly around the unit circle),

$$\begin{aligned} V_{ij}(t) &= \frac{1}{2}\left(1 + \tanh\frac{u_{ij}(t)}{u_0}\right), \\ n &= 10, \\ A &= 500, \\ B &= 500, \\ C &= 200, \\ D &= 500, \\ u_0 &= 0.02, \\ p &= 15, \\ \tau &= 1, \end{aligned}$$

and the initial condition is

$$u_{ij}(0) = a + \delta_{ij}, \tag{47}$$

where the δ_{ij}'s are uniformly randomly chosen in $[-0.1u_0, 0.1u_0]$, and a satisfies

$$\frac{1}{2}\left(1 + \tanh\frac{a}{u_0}\right) = 0.1.$$

(The random values used were identical for all the test runs.)

The dimension of the problem is 100. The problem is stiff, with a full Jacobian having an eigenvalue of maximum magnitude of approximately -3.8×10^5 at the initial point $t = 0$. All the eigenvalues are real. The range of integration was from 0 to 1. A true solution for comparison purposes was computed by LSODE with a maximum absolute error tolerance of 10^{-13}, the most stringent value for which the integration was successful in double precision with the default heuristics (including a maximum of 10 convergence failures per step).

3.6.3 DIMSEM Speedup Testing

Some simple measures of parallel performance are speedup and efficiency. Speedup can be defined in various ways, depending on the purpose of the measurement. To isolate the effect of the quality of the programming and utilisation of the parallel machine, the algorithm can be compared to itself via

$$S = \frac{\text{elapsed time for the parallel algorithm on 1 processor}}{\text{elapsed time for the parallel algorithm on } n \text{ processors.}} \tag{48}$$

The maximum such speedup is (usually) n, and the efficiency $E = S/n$ indicates what proportion of the maximum has been attained. To determine the practical value of the parallel algorithm, speedup can be defined instead as

$$S^* = \frac{\text{elapsed time for a good sequential algorithm}}{\text{elapsed time for the parallel algorithm on } n \text{ processors.}} \tag{49}$$

We use (48) here, and consider (49) in the next section.

The first test explored the speedups attained by each DIMSEMp on s processors compared to DIMSEMp on 1 processor, solving problem (46) with a maximum absolute error tolerance of 10^{-4}. The speedups were compared to the bound provided by Amdahl's law, which states that

$$S \leq A = \frac{P+Q}{P/n+Q}, \tag{50}$$

where P is the time spent in the parallel section of the code when it is run on a single processor, Q is the time spent in the sequential section, and n is the number of processors that can be applied to the parallel section.

The computation of the initial external stages $y_{0,i}$, $i = 2, \ldots, r$, for the DIMSEMs was done using LSODE, simply because it is convenient ($y_{0,1}$ comes directly from the initial condition). A more sophisticated implementation might use LSODE to compute $y_{0,2}$ and $y_{0,3}$, and then use DIMSEMp to compute $y_{0,p+2}$ for $p \geq 2$, allowing parallel computation to take place during the startup phase. In the test implementation, the approach taken however was to keep the timing statistics for the LSODE startup phase separate from that for the rest of the integration. Simply omitting the startup times then provides a reasonable approximation to the situation encountered in a long integration, in which the startup is a negligible proportion of the whole.

The results in Table 6 are from a run on one processor, and show the LSODE startup time, the time spent in all other sequential sections, and the time spent in the parallel sections, in columns 2-4 respectively. The remaining sequential sections are sufficiently small that the Amdahl speedup bounds are practically optimal.

The results in Table 7 are from runs on both one and s processors. The sequential LSODE startup phase is included in the timings. The efficiencies range from 76% to 87%, significantly less than optimal due to the non-parallelisable startup. However, when the startup time is excluded in Table 8, the efficiencies are quite good, ranging from 87% to 94%. This asymptotic level of performance can be expected to be approached as the integration becomes longer.

Besides the startup phase, another possible source of lost speedup is poor load balance resulting when the Newton iterations for each internal stage, each assigned to its own processor, converge after different numbers of iterations. On this particular problem, however, each stage proved to converge in one iteration on almost every step, and only rarely were the stages split between one and two, or two and three, iterations.

It is worth explaining why the times reported in Tables 6-8 increase with the order of the method. There are two reasons for this. First, these integrations were run at a fixed local-error tolerance of 10^{-4}, and the higher-order methods are achieving tighter global errors. Second, 10^{-4} is a sufficiently lax tolerance for this problem that the lower-order methods are more efficient. (This phenomenon could be taken advantage of by a variable-order scheme.) A comparison of the performance of the various DIMSEMs over a range of tolerances is provided in the next section, where the higher-order methods are shown to be more efficient at stringent tolerances.

3.6.4 Comparison of Parallel DIMSEMs with Fixed-Order LSODE

In this section we investigate the effectiveness of our DIMSEM implementation by comparing it to the fixed-order version of LSODE. The problem (46) was integrated from $t = 0$ to 1, as in the previous section, but for a range of tolerances 10^{-k}, $k = 1, \ldots, 11$. Since the runs at stringent tolerances took considerable CPU time, which was expensive on the parallel machine, they were done instead on a sequential machine

order	startup time	sequential time	parallel time	Amdahl speedup	Amdahl efficiency
2	1.46	0.07	13.47	1.99	0.99
3	2.17	0.06	21.07	2.98	0.99
4	2.74	0.10	41.67	3.97	0.99
5	3.16	0.12	53.36	4.96	0.99
6	3.73	0.17	86.52	5.94	0.99

Table 6: DIMSEM Amdahl's Law Performance (ignoring LSODE startup phase)

order	time on 1 processor	time on s processors	speedup	efficiency
2	15.00	8.64	1.74	0.87
3	23.30	9.94	2.34	0.78
4	44.50	14.73	3.02	0.76
5	56.64	14.86	3.81	0.76
6	90.43	19.78	4.57	0.76

Table 7: DIMSEM Parallel Speedups (including LSODE startup phase)

order	time on 1 processor	time on s processors	speedup	efficiency
2	13.54	7.17	1.89	0.94
3	21.12	7.76	2.72	0.91
4	41.77	11.98	3.49	0.87
5	53.47	11.66	4.59	0.92
6	86.70	16.02	5.41	0.90

Table 8: DIMSEM Parallel Speedups (ignoring LSODE startup phase)

and the elapsed times were corrected by the speedups in Table 8. To verify the accuracy of this indirect procedure, DIMSEM4 and LSODE4 were used to solve the problem on the parallel machine for a reduced range of tolerances, and good agreement with the corrected sequential timings was obtained. The timing for the DIMSEMs includes the LSODE startup phase. The sequential machine used was a SUN SPARCstation 20/172 with two 75 MHz processors and 256 MB of memory, running SunOS 5.4. As for the parallel testing, timing was done with the FORTRAN function dtime, based on a clock with microsecond resolution.

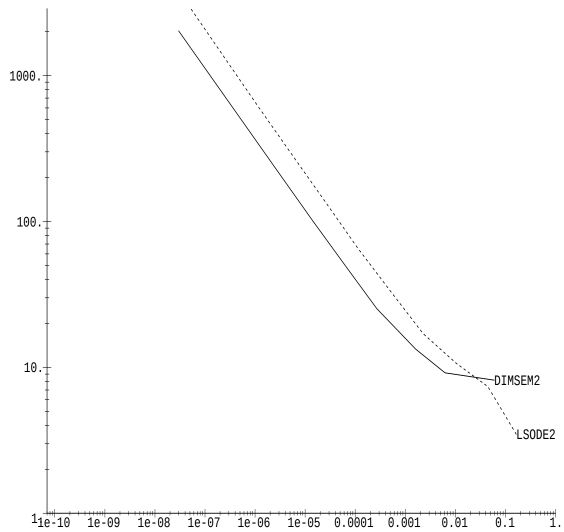
The results are shown in Fig. 2. In each case, the performance of the DIMSEM decreases at more relaxed tolerances, and decreases more with increasing order. This suggests a fixed overhead in the methods, and may have to do with the use of LSODE as a starting method for the DIMSEMs. At the most relaxed tolerances, LSODEp always outperforms DIMSEMp. For sufficiently stringent tolerances, however, DIMSEMp outperforms LSODEp by a factor of 30% to 50%, demonstrating the efficacy of the DIMSEM methods.

4 Conclusions and Future Work

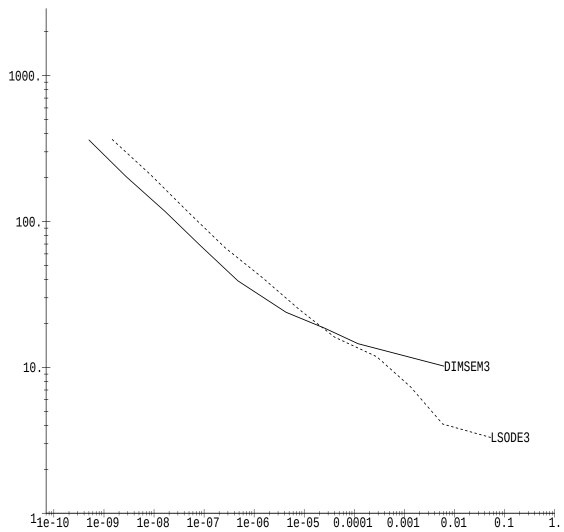
In this paper, we have introduced two new classes of methods which show promise for the solution of stiff ODEs on parallel computers. Both the LDM and DIMSEM classes contain methods of all orders ≥ 2. L-stable methods of orders 2–6 from each of these classes have been derived, and have behaved satisfactorily in fixed-stepsize numerical testing. A fixed-order, variable-stepsize implementation of the DIMSEM class has been developed, and testing performed on sequential and parallel machines compared the performance of the DIMSEMs to a fixed-order version of LSODE. While the DIMSEM implementation is not a production code, it has demonstrated the potential of DIMSEMs by showing them to be competitive with fixed-order LSODE on a practical test problem. In addition, the L-stability of the methods implies that they can be expected to outperform multistep methods on problems with eigenvalues near the imaginary axis.

There is much further research to do on DIMSEMs. It is possible that the DIMSEM class developed in this work has suffered from the restriction, made early on for the sake of simplicity when solving the order and single-eigenvalue conditions, that the internal abscissas $c_1, \ldots, c_s$ are all zero. An interesting possibility for future research would be to attempt to derive DIMSEMs for which at least some of the c_i's are free parameters, and use them to reduce the error coefficients, making the methods more efficient. This would likely require a new approach to proving Theorems 4 and 5 relating to the single-eigenvalue conditions. The external abscissas d_i's are also possible free parameters. Further generalisations might be to use the Nordsieck type, or some other type, of true-value function in place of the multistep-type true-value function employed here. Also, the nonlinear stability of the methods should be investigated. A production code employing both variable stepsize and variable order is a reasonable goal, which would require further study of robust local error estimators.

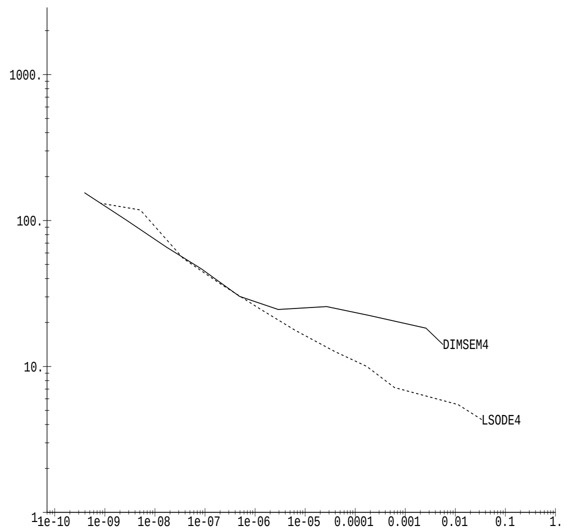
The parallel performance exhibited by the DIMSEMs in this paper suggests that further research on such methods would be worthwhile.



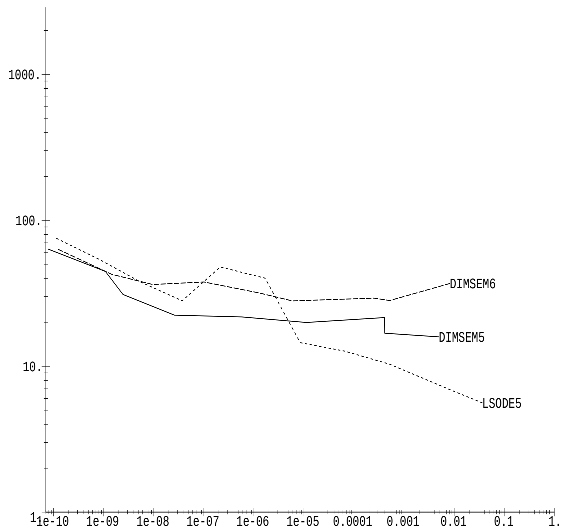
LSODE2 and DIMSEM2



LSODE3 and DIMSEM3



DIMSEM4 and LSODE4



DIMSEM5, DIMSEM6 and LSODE5

Figure 2: Elapsed Time (sec) vs. Endpoint Global Error for Parallel DIMSEM and Fixed-Order LSODE on Problem (46)

References

[1] A. Bellen. PADETEST - a set of real-life test differential equations for parallel computing. Technical Report 103, Università di Trieste, 1992.

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

[3] K. Burrage. Order and stability properties of explicit multivalue methods. *Appl. Numer. Math.*, 1:363–379, 1985.

[4] K. Burrage. Order properties of implicit multivalue methods for ordinary differential equations. *IMA J. Numer. Anal.*, 8:43–69, 1988.

[5] K. Burrage. *Parallel and Sequential Methods for Ordinary Differential Equations*. Oxford University Press, 1995.

[6] K. Burrage and F. H. Chipman. Construction of A-stable diagonally implicit multivalue methods. *SIAM J. Numer. Anal.*, 26(2):397–413, 1989.

[7] K. Burrage and P. Moss. Simplifying assumptions for the order of partitioned multivalue methods. *BIT*, 20:452–465, 1980.

[8] J. C. Butcher. DIMSIMs — Diagonally implicit multi-stage integration methods. Seminar given at the University of Toronto, Nov. 26, 1991.

[9] J. C. Butcher. General linear method: A survey. *Appl. Numer. Math.*, 1:273–284, 1985.

[10] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations*. John Wiley & Sons, 1987.

[11] J. C. Butcher. Diagonally-implicit multi-stage integration methods. *Appl. Numer. Math.*, 11:347–363, 1993.

[12] J. C. Butcher. General linear methods for the parallel solution of ordinary differential equations. *World Series in Applicable Analysis*, 2:99–111, 1993.

[13] J. C. Butcher. A transformation for the analysis of DIMSIMs. *BIT*, 34:25–32, 1994.

[14] J. C. Butcher and Z. Jackiewicz. Diagonally implicit general linear methods for ordinary differential equations. *BIT*, 33:452–472, 1993.

[15] G. J. Cooper. The order of convergence of general linear methods for ordinary differential equations. *SIAM J. Numer. Anal.*, 15(4):643–661, Aug. 1978.

[16] R. F. Enenkel. *DIMSEMs – Diagonally Implicit Single-Eigenvalue Methods for the Solution of Stiff Ordinary Differential Equations on Parallel Computers*. PhD thesis, University of Toronto, 1996.

[17] W. H. Enright. Improving the efficiency of matrix operations in the numerical solution of stiff ordinary differential equations. *ACM Trans. Math. Softw.*, 4(2):127–136, June 1978.

[18] W. H. Enright and J. D. Pryce. Two FORTRAN packages for assessing initial value methods. *ACM Trans. Math. Softw.*, 13:1–27, 1987.

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

[20] E. Hairer and G. Wanner. Multistep-multistage-multiderivative methods for ordinary differential equations. *Computing*, 11:287–303, 1973.

[21] E. Hairer and G. Wanner. On the Butcher group and general multi-value methods. *Computing*, 13:1–15, 1974.

[22] D. Higham. Everything you wanted to know about aSymptotic Error eXpansions but were afraid to ask. Course handout, May 1991.

[23] A. C. Hindmarsh. LSODE and LSODI, two new initial value ordinary differential equations solvers. *ACM SIGNUM Newsletter*, 15(4):10–11, 1980.

[24] K. R. Jackson and S. P. Nørsett. The potential for parallelism in Runge-Kutta methods. Part I: Runge-Kutta formulas in standard form. *SIAM J. Numer. Anal.*, 32:49–82, 1995.

[25] R. Skeel. Analysis of fixed-stepsize methods. *SIAM J. Numer. Anal.*, 13(5):664–685, Oct. 1976.

[26] Waterloo Maple Software. Maple V. A symbolic algebraic manipulation package.