DIFFICULTIES IN EVALUATING DIFFERENTIAL EQUATION SOFTWARE

by Robert D. Russell
Simon Fraser University
Burnaby, B.C. V5A 1S6/Canada

I. Introduction

The recent proliferation mathematical software is accompanied by the responsibility for its evaluation. This has of course often been recognized by numerical analysts, and comparisons of many types have been conducted. In areas where the software is complex and multi-faceted, such as the numerical solution of boundary value problems for ordinary differential equations (BVODES), making a relative comparison of the performance of codes in order to recommend which is the most appropriate in a given situation is at best extremely difficult. In this paper, we shall discuss the types of comparisons which have been performed for BVODES, the limitations of these comparisons, and the problems which arise when these limitations are not fully appreciated. The current situation is well summarized in [1]: "Unfortunately the standards expected for mathematical exposition are only rarely applied to the reporting of computational experiments." While the focus here is on global methods for BVODES and their particular characteristics, the concerns and points expressed are equally applicable for most other software for differential equations and indeed many other areas of numerical analysis.

II. History of BVODES

Comparison of BVODE methods, as for most other numerical methods, have traditionally been of 3 types:

(i) comparison of methods in a general way,
(ii) comparison of algorithms, and
(iii) comparison of codes.

In a comparison of type (i), it is assumed that methods can be linearly ordered, i.e. that it is reasonable to call one method better than another or others. For example, a recent numerical analysis book states unequivocally that multiple shooting is superior to finite differences and finite elements for solving difficult BVODES. In support of this statement, only one problem is solved, only equal spacing is used for the global (finite difference and finite element) methods, only low order global methods are used, and only one finite element method is considered. It is hopefully clear that comparisons of type (i) for popular, well-used methods will often border on the ridiculous because the objective is far too vague and unrealistic.

Comparisons of type (ii) have been quite popular, for it is natural to investigate which methods seem the most promising before investing the time to im-
implement them in portable mathematical software. In [2] algorithms for the three common finite element methods (collocation, Rayleigh-Ritz-Galerkin, and least squares) are compared for an $m$th order linear BVODE, and on the basis of operation counts it is concluded that least squares is more efficient than collocation which in turn is more efficient than Rayleigh-Ritz-Galerkin. When a more efficient linear system solution scheme for collocation is considered the roles of least squares and collocation are switched [3]. Recently variants of these finite element methods have been compared (on the basis of CPU times for simple implementations), and collocation is concluded to be most efficient [4]. Obviously conclusions could be reversed if some more efficient versions are developed. Other examples of type (iii) comparisons are [5], and [6], where the first concludes that certain finite difference and initial value methods are more efficient than collocation methods, and the second, considering a more efficient implementation of collocation, concludes that it is competitive with finite differences.

Regardless of the conclusions drawn from a comparison of type (ii), it is still not clear that the type (iii) comparisons (i.e. of codes themselves) will give consistent results. Several components, such as the mesh selection and nonlinear iteration strategies, are difficult to evaluate adequately in terms of operation counts in algorithms, yet they are overriding features in determining the efficiency of the codes. This leads some to suggest that the only realistic definition of an algorithm for a numerical method is a computer code itself. Type (iii) comparisons are ultimately the most realistic ways to compare methods; unfortunately, difficulties still arise in areas such as BVODES from the complexity of the general programs. This is discussed further in future sections.

III. Global BVODE Codes

The first BVODE codes were of the initial value type, incorporating the available robust initial value codes, usually with a shooting or multiple-shooting strategy (e.g., see [7], [8], [9]). The BVODE codes based upon global (non-initial value) methods which have been developed to the generality described below are PASVA3, a finite difference code using the trapezoidal rule with deferred corrections [10] and COLSYS, a spline collocation code [11].

Figure 1 shows how PASVA3 and COLSYS are more-or-less designed. The purpose of both programs is to produce an approximate solution accurate to within the user's requested error tolerance (TOL). Both also provide an estimate of the error (EST) in the global solution. The adaptive mesh selection and nonlinear iteration strategies are two features of the codes which have been extensively tested and enable them to solve fairly difficult problems. The mesh selection strategy operates as follows: after a solution has been computed on a mesh $\pi$, the error estimates are used to select a new mesh $\pi'$ such that hopefully the new approximate solution on $\pi'$ satisfies $|\text{EST}| \leq \text{TOL}$ and the magnitude of the error is the same