Riv. Mat. Univ. Parma (6) **2*** (1999), 69-87

F. MAZZIA and D. TRIGIANTE (*)

Computing codes for ordinary differential equations: state of art and perspectives (**)

1 - Introduction

The fundamental importance of mathematics in the current technological development is by now recognized also in official documents (see for example [**20**]). The fields of mathematics that more directly influence every phase of the technological cycle are the Modeling and the Scientific Calculus. While everyone has a more or less precise opinion of the complex of ideas and techniques which are the basis of Modeling, at least for having learned them in the courses of Physics and Mathematical Physics, this is not equally true for Scientific Calculus, even if, paradoxically, we may say that are very few the mathematicians who have never used some products in commerce (Mathematica, Matlab, etc). This for two reasons:

(a) the mathematics curricula of our Universities seldom go beyond the first elements of the Numerical Analysis;

(b) Scientific Computing is a very young discipline developed in the last thirty years and therefore, in wide measure, it is unknown to unacquainted people.

It is also a varied and complex discipline, ranging by now in every field of the Applied mathematics, from the approximation to the simulation of dynamical systems of every type. Besides, it is a discipline in fast evolution, due both to the fast

---

(*) Dipartimento Interuniversitario di Matematica, Via Orabona 4, 70125 Bari, Italy, Fax: + 39 080 5460612, E-mail: mazzia@dm.uniba.it; Dipartimento di Energetica «S. Stecco», Via Lombroso 6/17, 50134 Firenze, Italy.

(**) Received September 2, 1999. AMS classification 65 L 05, 65 L 06, 65 L 10.
Work supported by GNIM and MURST.

development of the computers, and to the enormous pressure exercised from the always-increasing necessity of the applications. To start with the description of some specific topics of this discipline, we consider first the dimension of the problems. Large value of this parameter is the rule rather than the exception. In fact, even relatively simple continuous problems, once discretized, can give rise to large dimensional problems whose solution requires, often, the execution of a number of operations growing polynomially and, sometimes, also exponentially with the dimension. These operations are not done with real numbers but with a finite representation of them. This implies that there is always a source of errors that disturbs the computing model. Such perturbation operates in a way very similar to that of a permanent perturbation of the equations in the continuous case. How to control the effect of such perturbations? Which are the most opportune algorithms to use? How to structure the calculations in a way that the memories of the computers are sufficient? (think to the three-dimensional models of the fluidodynamics, where the number of the unknowns grows like the cube of the dimensions). Above all, how to structure the calculations in order to have the result in reasonable times?

All these questions and many others, that for brevity we omit, require a systematic study and are the basis of Scientific Calculus.

Naturally, as is common praxis in mathematics, the problem is divided into many subproblems. This process may be repeated many times, until a simple homogeneous problem is derived. The objective is to obtain algorithms with increasing efficiency and reliability. Such algorithms, often, are modules to be combined in a way that more and more complex problems can be solved in more and more reliable way.

The process generates an extreme division of the discipline, very similar to what happens in other fields of mathematics. This may be considered either a positive or negative phenomenon, according to the points of view and we will not be here to give a judgment. Rather we want to emphasize that, in this particular field, the main objective of the study is to render the algorithms more efficient and easy to use. In fact it often happens that the end product, even if obtained after decades of intense study, can be used in extremely simple way by everyone, as the example of the commercial programs demonstrates.

In this review article we will describe some of the problems connected to the construction of codes for the numerical solution of ordinary differential equations. Our aim is to show, on a concrete example, the complexity of the process which, starting from the problem, arrives at its numerical solution handling a more or less large number of intermediate numerical problems. The whole process needs

to be at a very high level of reliability and ready to be encapsulated «in the plastic» (or, if it is preferred) in the «chip» of a computer.

2 - Continuous and discrete problems

In order to fix the ideas, the considered continuous problem will be in the normal form:

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

where the function $f:[t_0, T] \times \mathbb{R}^m \to \mathbb{R}^m$ is so smooth as to satisfy the conditions of existence and uniqueness of the problems considered later on. When studying the order of convergence of the methods, more restrictive conditions are usually required.

The study of the qualitative behavior of the solutions of (1) has engaged the mathematicians in the last century and is still a field of active researches, which often surprises with the variety of its results. Moreover such engagement has been culturally and scientifically very profitable since it has generated new techniques and new ideas, all of which have useful applications in the treatment of other problems.

It turns out, however, that the qualitative information, although of extreme importance, often is not enough in the applications, where it is necessary a more specific knowledge of the solutions, even if on a discrete subset of points. The first attempts to obtain such information go back to the beginnings of the Mathematical Analysis (Newton, Euler, etc.), even though with insufficient results, if particular simple cases are excluded. To the beginning it seemed that the only difficulty was the large amount of time required to execute the needed calculations. The necessity of having machines, or, at least, techniques, to shortening the time of calculation, was perceived very soon (Leibniz, Pascal, ...). Kepler asserted that the invention of the logarithms had doubled the life of an astronomer.

The sideboard that the only difficulty was the quantity of calculations remained for more than two centuries. Also Richardson, at the beginning of this century, thought in the same way when he proposed its «Mathematical factory», in which 64000 mathematicians, trained to execute particular types of calculations, could have been able to solve, in «real time», the equations for the weather forecast.

With the advent of modern computers, and therefore with the overcoming of the mentioned long lasting difficulty, the existence of a more subtle difficulty, i.e. the possible catastrophic propagation of unavoidable errors, showed up. As a mat-

ter of fact, even if the plan of the «Mathematical factory» could have been realized, the executed calculations would have been unusable since affected from high relative errors. Von Neumann was among the firsts to be aware of this. He obtained a condition on the discretization steps to have the computations executed in regime of stability. This last term, already famous in the qualitative theory, enters for the first time in the study of the propagation of the errors. In the following years it has become the hinge around which the current theory of the approximate solutions has been built. In the same period many of the results obtained in the qualitative theory of differential equations were extended to the solutions of the difference equations. This gave a remarkable impulse to the study of the numerical methods. To approximate numerically the solution of (1) is equivalent in fact to replace it with a difference equation.

With the passing by of time it became evident that the concepts which in the past the construction of the numerical methods was based on, i.e. the local error and the order of convergence, were inadequate. Examples of methods with high orders of convergence and behaving in disastrous way were very soon found. Also the 0-stability concept, recently introduced, resulted widely inadequate.

In years 60 and 70, people started to realize that the right condition to impose was to demand that the solutions of both continuous and discrete problems should share the same qualitative behavior. Since, as we have said, the qualitative behavior of the solutions of (1) may be extremely varied, depending on the shape of the limit set, this condition is easy to formulate, but extremely difficult to realize.

One started from the simpler case, fortunately the most common in the applications, i.e. the case, in which an initial condition is associated to (1) and the resulting problem has an asymptotically stable equilibrium point. One supposes moreover that the solutions are to be approximated in a region around such point. In this case, the qualitative theory, both in the continuous and in the discrete case, had already developed mathematical instruments, like, for example, the fundamental theorem of stability in first approximation, which permits to study the linearized problem. The introduction of the famous equation test

$$y' = \lambda y\,, \qquad \mathrm{Re}\,\lambda < 0\,,$$

done by Dahlquist is justified by the cited theorem. For each method it is then defined a region in the complex plane $h\lambda$ (said region of Absolute stability) within which the product $h\lambda$ has to stay in order to have the equilibrium point asymptotically stable for the discrete equation.

The choices of the discrete equations are unbounded. A long job of classifica-

tion and study of methods essentially, but not exclusively, belonging to the two great classic families, i.e. the Multistep and the Runge-Kutta methods, started in the sixties and still continues.

The absolute stability regions can vary very much from method to method: they may range from empty regions to regions containing all the negative complex plane (A-stable methods). Often there is a conflict between the order of accuracy of the methods and the amplitude of the absolute stability region. A typical result is due to Dahlquist:

Theorem 2.1 (Dahlquist Barrier). *If initial conditions are associated to equations in the multistep class, the resulting methods cannot be A-stable with order of precision greater then two.*

3 - Stiff problems

As from the sixties some applications coming from Chemistry and Mechanics had evidenced that, with the methods known at that time, the solution of some problems demanded the use of very small integration steps. Such problems were said *stiff* and are characterized by two constants of time very large apart. It is stiff, for example, the equation test if the integration interval is $[0, T]$ with $T = 10$ and $\lambda = -10^6$. The ratio $\frac{T}{|\lambda|}$ is said stiffness ratio. To recognize stiffness is not always easy as in the example and the discussion on its exact definition still continues. It happens sometimes that some entities are more recognizable from their consequences than from their definitions. This is the case of stiff problems. The consequence is that, if a method has a bounded region of absolute stability, to have $h\lambda$ inside it, one must take $h \approx |\lambda|^{-1}$. This value of h is very small in many applications. Explicit methods, having bounded regions of absolute stability, are not suitable to solve this type of problems. A-stable methods, necessarily implicit, are needed instead. To fix the ideas, we will say that the stiff problems have, in general, solutions with fast variations on small intervals. We have said in general, since it is obvious that for special initial conditions this may be not true, but the fast variations come in as soon as the initial conditions are perturbed. From the point of view of the Numerical Analysis this does not make any difference. The use of implicit methods, however, requires, the solution of a nonlinear system at every step. Moreover, for the class of multistep methods, at least in their classic formulation, the limitation due to the barrier of Dahlquist, which limits the attainable precision, has to be considered. An analogous limitation does not exists for class of the Runge-Kutta methods and this explains the preference for this class

of methods in the last years. Recently the class of the multistep methods has been generalized. In such generalization the Dahlquist barrier has been eliminated and therefore A-stable methods of all order can be used. The new methods have been called Boundary Value Methods (BVMs).

4 - Boundary value methods

Multistep methods are difference equation of order k while equation (1) has order one. This means that the discrete problem needs $k-1$ additional conditions: in the classical use of such methods, all the additional conditions are imposed at the beginning of the interval. In such case the above mentioned conflict between the stability conditions and the precision order appears. If one takes the freedom of using such additional conditions where it appears to be more convenient, the conflict disappears. The question has been discussed in the recent monograph [**12**]. Obviously, except for the case $k=1$, methods which were considered stable in the previous formulation, are no longer stable in the new formulation and vice versa. Typical is the case of the Top Order Methods (TOMs), i.e. those having the maximal precision order in the class ($p=2k$). They were already known in the fifties and never used because unstable. As BVM, they are perfectly stable (perfect stability means that their absolute stability region coincide with $\mathbb{C}^-$).

Multistep methods contain well-known historical families of methods. For future references we report such families along with their generalizations as BVMs.

4.1 - *Generalized backwards methods*

The methods in this family are defined by

$$\sum_{i=0}^{k} \alpha_i y_{n+i} = h f_{n+j}. \tag{2}$$

For $j=k$, one obtains the classical *backward differentiation formulae*. They are important because they are used in many of the most important codes. Their region of absolute stability is very large and contain $\mathbb{C}^-$ for $k=1$, while it become

smaller and smaller as k increases. For $j = \nu$ given by

$$\nu = \begin{cases} \dfrac{k+2}{2} & \text{for even } k\,, \\ \dfrac{k+1}{2} & \text{for odd } k\,. \end{cases} \tag{3}$$

the methods of order k, used as BVM with ν additional conditions at the beginning of the interval and $k - \nu$ at the end, are A-stable for all k.

4.2 - *Generalized Adams methods*

Such family is defined by

$$y_{n+j} - y_{n+j-1} = h \sum_{i=0}^{k} \beta_i f_{n+i}. \tag{4}$$

The classical Adams-Moulton methods are obtained for $j = k$. Such methods have very small absolute stability regions and were used only for non stiff problems.

As BVMs, by taking

$$j = \begin{cases} \dfrac{k+1}{2} & \text{for odd } k\,, \\ \dfrac{k}{2} & \text{for even } k\,, \end{cases} \tag{5}$$

the methods of order $k + 1$, are A-stable for all k. On such methods is based the code GAM [**31**].

4.3 - *Symmetric schemes*

This family has not a classical counterpart since they were all unstable. As BVMs, those of higher order are perfectly A-stable. Their main characterization is

$$\alpha_i = -\alpha_{k-i}, \qquad \beta_i = \beta_{k-i}, \qquad i = 0, \ldots, k\,. \tag{6}$$

In practice such characterization is too wide. It is necessary to introduce some more strict classification. We have then

- Extended Trapezoidal Rules (ETRs)

$$y_{n+\nu} - y_{n+\nu-1} = h \sum_{i=0}^{2\nu-1} \beta_i f_{n+i}. \tag{7}$$

- Extended Trapezoidal Rules of second type (ETR_2s)

$$\sum_{i=-\nu}^{\nu-1} \alpha_{\nu+i} y_{n+i} = h(\beta f_n + (1-\beta) f_{n-1}),$$

and the already mentioned TOMs. All such methods are suitable to be used in special problems where the perfect stability is important (Hamiltonian problems, boundary value problems, etc.).

5 - Mesh selection

As evident in the case of stiffness, many difficult problems are multiscale. This implies that, if one wishes to recover from the numerical approximation enough information about the solutions in a reasonable amount of time, the use of variable stepsizes is needed. This leads to the problem of defining the most suitable choice of points (mesh points), on which the value of the solution will be computed, in order to bound the global error within a prefixed tolerance.

Mesh selection is in a sense the numerical counterpart of the scaling theory of Mathematical Physics. It is a non trivial task, which has been studied in the last thirty years. There is not, so far, a complete satisfactory theory of mesh selection. Recently, (see [**11**], [**12**]), it has been proposed the introduction of two parameters, essentially the l_∞ and the l_1 norms of the solutions, as characterizing the presence of multiscale for both continuous and discrete problems. The fulfilment that the discrete problem has both parameters as near as possible to the corresponding continuous ones, permits to select an optimal mesh. Such criterion is costly and, in our opinion, can be justified only for boundary value problems. It can be proved that for well conditioned initial value problems, it may be simplified. The obtained simplification reduces to the classical *deferred correction* (see [**12**]). Anyway this problem needs more studies. In practice one uses more empirical methods usually based on the control of local error, which essentially consists in the computation of a measure of the local error, scaled with the user specified accuracy require-

ment, and in choosing, at the current time of integration, the optimal stepsize as

$$h_{new} = h_n \left(\frac{1}{err} \right)^{1/(p+1)}$$

where p is the order of the method and h_n is the old stepsize. Usually a safety factor is introduced in order to have more probability that the error will be accepted the next step. If the control on the local error is satisfied then the new stepsize is used to advance the solution in time. For explicit methods this stepsize variation strategy may generate a sequence of oscillating stepsizes. Such behavior depends on a conflict between accuracy and numerical stability when they are used for solving stiff or mildly stiff problems. It can be explained by studying the corresponding dynamical systems on the border of the stability region. The SC-stability has been introduced to explain this phenomenon [**27**]. The study of the best way of choosing the stepsize is essential for the construction of efficient codes, so researches have been made to find techniques that generate a smoother sequence of stepsize for explicit and implicit methods [**21**], [**22**], [**23**], [**24**], [**37**]. One of this is based on a discrete PI (proportional integral) controller which, for implicit method, is essentially based on the following formula

$$h_{new} = h_n \left(\frac{1}{err_{n+1}} \right)^{1/p_1} \left(\frac{h_n}{h_{n-1}} \right) \left(\frac{err_n}{err_{n+1}} \right)^{1/p_2}$$

where p_1 and p_2 are opportunely chosen [**24**]. This technique is used, for example, in the codes RADAU5 (1996) and RADAU (1998) written by E. Hairer and G. Wanner, based on RADAU IIA methods [**27**], [**28**].

For one-step methods, once the new stepsize has been chosen, one needs only to advance the solution in time using the same numerical method. On the contrary, the use of a variable stepsize for multistep methods needs some more care because such methods use information taken from previous points, which have been computed by using different stepsizes. The most common techniques used are:

(1) the constant coefficient technique, which computes the constant step multistep formula by using the new stepsize and then interpolates to find the approximation of the solution in the previous points;

(2) the variable coefficient technique, which computes the coefficient of the multistep formula of a given order by using variable stepsize.

In order to maintain the stability and the convergence properties of the whole method, restrictions on the ratios of contiguous stepsizes have to be respected, usually the sequence of stepsizes must be quasi-uniform and $h_{n+1}/h_n = 1 + O(\max(h_n))$. Some theoretical and empirical results show that the variable coefficient implementations have better stability properties.

The amount of work that each implementation requires is different. The constant coefficient method requires an interpolation if the stepsize changes. This can be performed efficiently by using the Nordsieck history array, with a cost proportional to m, the dimension of the continuous problem. For the variable coefficient technique one needs to compute the new coefficients and this work is proportional to k, the number of steps of the formula.

An alternative technique consists in using, for implicit methods, a fixed leading coefficient formula. Such formula uses variable coefficients except for the leading one, which is kept independent on the ratios of the stepsizes [**34**]. This has the advantage to minimize the number of factorizations, as we will see in the next section.

For multistep formulae the estimation of the local truncation error is computed by using asymptotic relation, that are easy to find in the case of constant stepsize and can be generalized in the case of variable stepsize. For Runge-Kutta methods the approximation of the local error may be computed by Richardson extrapolation, or by using embedded Runge-Kutta formulas, that compute, without further evaluations of functions, two different approximation of the solution of order p and q, whose difference gives an estimation of the local truncation error.

6 - Solution of nonlinear systems

Together with the mesh selection described in the previous section, the solution of nonlinear systems is the most delicate of the entire process. While the mesh selection reflects the presence of possible multiscales and then it is more related to the properties of the continuous solutions, the solution of the nonlinear systems depends more on the dimension of the discrete problem. In other words, if the previous steps (i.e. choice of the discretization method, mesh selection, etc.) have been done judiciously, the discrete problem should have a similar conditioning of the continuous one. This means that the two problems have a similar sensitivity to perturbations. If this is true, then the difficulties from now on are essentially due to the dimension of the discrete problem and to the ability to get a good approximation of the initial profile of the solution from which the iterative

process for solving the nonlinear problems will start. We shall discuss the latter problem later. Concerning the dimension, in the most general case, this is given by the product Nm where N is the total number of points needed to cover the whole interval and m is the dimension of the continuous problem. Even though the mesh selection has minimized N, still this number may result to be very large. The main objective is now to minimize the number of operations to arrive at the final solution, as well as to design the sequence of operations such that no ill conditioning is added to the problem.

For IVPs, the dimension may be lowered by subdividing the interval in a certain number of parts and then solving sequentially the problem on each subinterval. The value of the solution in the last point of each subinterval is the initial value of the next problem. Of course, if the subintervals are N, the resulting methods turn out to be one-step, such as Runge-Kutta methods. However, since such methods are multistage, the dimension of the nonlinear systems is ms where s is the number of stages.

For linear multistep formulae the nonlinear equation to be solved at each step is:

$$y_n - \alpha_n h_n f(t_n, y_n) = b_n,$$

with α_n a positive scalar. The solution of this equation by the Newton method require at each iteration one jacobian evaluation and the solution of a linear system of size m. This can be expensive if m is large.

Methods which do not require the solution of linear systems are, for example, the simple functional iteration or the predictor corrector technique (usually the predictor is an explicit method and the corrector is an implicit method with the same accuracy). In both cases the A-stability of the implicit method is no longer preserved by the resulting scheme, this means that such techniques are useful only for nonstiff or mildly stiff problems.

For stiff problems the nonlinear equation is therefore solved by using the Newton method or some variants of its in order to minimize the computational cost. Usually the jacobian is kept constant, so that only one LU factorization is needed. The initial approximation of the solution is computed by using an interpolation polynomial, or an explicit method.

For example, the solver LSODE (1980) written by Hindmarsh [**29**] uses predictor-corrector Adams methods in the nonstiff case, and a fixed coefficient implementation of Backward Differentiation Formulae in the stiff case. The nonlinear system are solved by a modified Newton scheme.

In the solver CVODE (1994) for stiff and nonstiff initial value problems written by Cohen and Hindmarsh [**18**], the methods used are fixed-leading-coefficient implementation of variable coefficients Adams and BDF methods. The nonlinear systems are solved by using a simple fixed point iteration in the nonstiff case and a modified Newton iteration in the stiff case.

For Runge-Kutta methods the solution of the nonlinear systems is more entangled, since the dimension depends on the internal stages and usually the initial approximation must be known a priori for all the stages.

The nonlinear systems are of the form:

$$Y_i - y_{n-1} - h \sum_{j=1}^{s} a_{ij} f(t_{n-1} + c_j h,\, Y_j) = 0\,, \qquad i = 1,\, s$$

which in matrix form became

$$R(Y) = Y - h(A \otimes I_m)\, F(Y) - e \otimes y_{n-1} = 0$$

where the matrix A, of size s, is the matrix containing the coefficients a_{ij} of the method, $Y = (Y_1, \ldots, Y_s)^T$ is the vector of the internal stages, $F(Y) = (f(t_{n-1} + c_1 h,\, Y_1), \ldots, f(t_{n-1} + c_s h,\, Y_s))^T$ is the vector of the function evaluations, $e = (1 \ldots 1)^T$ and $\otimes$ is the Kronecker tensor product. The solution of this systems with a simplified Newton method leads to the following iteration

$$(I - hA \otimes J)(Y^j - Y^{j-1}) = -R(Y^{j-1})$$

where J is the jacobian of f evaluated at $(t_{n-1},\, y_{n-1})$. When A is a full matrix, all the stages are coupled. If a direct solution method is used, the LU decomposition requires $\frac{2}{3} s^3 m^3$ operations. This becomes expensive if $s > 3$. The research has followed two main directions to try to reduce this computational cost.

The first idea, due to Butcher [**14**], is to transform A to a simpler matrix, for example diagonal or block diagonal, by using a similarity transformation $T^{-1} A T = D$. This transformation requires the solution of some real linear systems of dimension m for real eigenvalues and some complex linear systems for the complex eigenvalues.

This procedure has been used in the code RADAU5, which computes the quantities $z_i = Y_i - y_{n-1}$ by solving real and complex systems of size m obtained using the Jordan canonical form of the matrix A.

If the matrix A has only real eigenvalues then the computational cost for the LU factorization is reduced to $\frac{2}{3} s m^3$. Moreover, if the matrix has a one point spectrum, only one factorization is needed, making the computational cost equiva-

lent to that of multistep methods. This is the reason of developing singly implicit RK methods. The basic idea in the code STRIDE (1979), written by J. C. Butcher, K. Burrage and F. H. Chipman, is the use of the singly implicit Runge-Kutta methods of Burrage [**13**]. Singly-implicit methods have a coefficient matrix with a one-point spectrum, so the operation count is reduced to the level which prevails in linear multistep methods.

If the eigenvalues are more then one, then it is possible to implement in parallel the solution of the s linear systems, thus giving a parallelism across the method implemented for example in the code PARSODES (1996) written by C. Bendtsen in Fortran 90 and MPI that uses a multi implicit Runge-Kutta with parallelization across the method [**5**].

The second idea of solving the nonlinear systems is to use a one-step splitting iteration

$$(I - hL \otimes J)(Y^j - Y^{j-1}) = -R(Y^{j-1})$$

where L is a lower triangular matrix. The convergence of this kind of splitting is studied for the test problem $y' = \lambda y$ with λ in the negative part of the complex plane. In this case, by using constant stepsize, we obtain a linear difference system of the form $Y^{k+1} = SY^k + \beta$ with iteration matrix S defined as

$$S(q) = q(I - qL)^{-1}(A - L), \qquad q = h\lambda$$

and the region of convergence is described by

$$\Gamma = \{ q \in \mathbb{C} : \varrho(S(q)) < 1 \}.$$

with $\varrho(S(q))$ the spectral radius of S. If we are dealing with A-stable formulae, L should be chosen in order that Γ contain the left half complex plane (A-convergence) or at least the region $\{q : \pi - \alpha < q < \pi + \alpha\}$ (A(α)-convergence) with $\alpha \simeq \pi/2$. Another constraint for the elements of L is to require $\varrho(S(\infty)) \simeq 0$ because in such a case the iteration is fast in correspondence of stiff components of the solution. In general we need s factorizations that can be performed in parallel. The code PSIDE (1998), written by J. J. B. de Swart, W. M. Lioen, and W. A. van der Veen, that implement the RADAU IIA method of order 7, uses this techniques for solving implicit differential equations on shared memory parallel computers [**38**].

If the matrix L has constant entries on the main diagonal then we need to perform only one real factorization to solve the block lower triangular system. This technique is used in the code GAM (1997) [**31**], [**32**]. This code implements a block version of the Generalized Adams Methods and the novelty with respect to the

other codes based on Runge-Kutta methods is the error estimation which allows an order variation strategy. The one point spectrum of the matrix L considerably reduces the computational cost making the code comparable with the most used ones. This code has also been parallelized using the diagonalization with similarity transformations for solving the nonlinear systems. The resulting matrices are diagonal with complex eigenvalues, so the number of processors that can be used is $s/2$ if s is the number of stages. The order variation strategy is allowed by making some processors idle when the order used is not the maximum one. In the actual implementation the formulae have order 3,5,7 and 9 and the number of active processor changes form 2 to 5 depending on the order used [**33**].

7 - Boundary value problems

For this type of problem, the condition coupled to (1) to get a unique solution is not concentrate at the first point, but it involves more points. We shall refer, for simplicity, to the case where such points are the first and the last of the interval of integration. This makes more difficult both the problems of mesh selection and the solution of linear systems. Concerning the former, for example, the techniques based on local errors are often inadequate because at each point the global error depends not only on the previous points, as in the case of IVP, but also on the subsequent ones. Only when one can assume that the error at each point depends essentially on the points around it, such technique may give good results. Note that this is equivalent to ask that the Green's function $G(t, s)$, for each fixed t, decays fast as $|t - s|$ increases. When this is not true, the general approach of using two measures, already mentioned in section 5 gives much better results [**7**], [**12**].

7.1 - *Choice of the methods*

The choice of the methods needs a deeper discussion. In fact until few years ago, the solution of this kind of problems was brought back to the solution of an *equivalent* IVP problem. This sounds logic. One looks for the initial condition giving rise to the same solution, by solving iteratively a nonlinear problem (*shooting method*). Here we have a clear example of the conflict between what is *equivalent* in Analysis and what is so in Numerical Analysis. The key difference is that the two disciplines operate on two different sets of numbers. Only when the initial condition is obtained with infinite precision and the subsequent computations are made with the same precision, the two problems can be considered equivalent. This is never the case. It turns out that the IVP problem correspondent to a well-

posed BVP is always ill conditioned. In fact a well-posed boundary value problem needs to have the so-called dichotomy. To be more clear, suppose to consider a linear function $f(t, y) \equiv Ay$. The dichotomy amounts to assert that the matrix A has eigenvalues in both sides of the complex plane. An initial value problem with such matrix would have unstable critical points, and then be very sensitive to perturbations. The shooting technique is then an example of how a good continuous problem may be transformed in a bad discrete one by choosing a wrong method. Of course such criticisms are more evident when the interval of integration is large (or when the absolute value of real parts of the eigenvalues are large).

It is then not surprising that the most recent and best codes do not use neither the celebrate shooting method, neither any of its derivations. For example the codes COLSYS and COLNEW [**3**], [**4**] use a truncated powers collocation method at Gaussian points, whereas the code TWTBVP [**17**] uses a deferred correction algorithm with mono-implicit Runge-Kutta methods.

A code based on BVMs is under construction. It will use the symmetric schemes described in Section 4. The reason of such choice is the following. The presence of dichotomy in such problems requires that the method should treat in a symmetric way both the decreasing and increasing modes. Such condition, not necessary for contractive problems, is very important for the present class of problems. The symmetric schemes are able to match such necessity.

8 - Perspective

It is evident, from what said in the previous sections, that the existing codes for IVPs are well suited to solve contractive problems. Often they give good results for more general problems but they are unable to match special requirements such as, for example, the conservation of energy for conservative systems. The main objectives of next generation codes is then to be able not only to approximate the solution of larger class of problems but also, to keep some important properties of the continuous solution. Among the most important problems which need more insight we quote

1) Hamiltonian problems;
2) algebraic differential problems;
3) delay differential problems.

Concerning the Hamiltonian problems, in the last decade there has been a great improvement in the identification of numerical methods, which are able to

define a discrete symplectic map. There is a debate whether symplecticity is enough to get the conservation properties. While this is enough for continuous maps, it may not be true for discrete maps. Anyway, for Runge-Kutta methods, which are one-step methods, many symplectic methods have been identified. For multistep methods, the problem is more difficult. It has been proved (see [**19**]) that when used as to generate discrete initial problems, they are unable to provide conservative methods. On the contrary, when they are used as BVMs many of them, the symmetric ones, are conservative on a subset of the mesh points.

For Differential-Algebraic equations the research has allowed the development of robust and efficient codes, even if some numerical and theoretical difficulties that not arise in the solution of ODEs, need more studies. Codes like DASSL written by L. Petzold and well described in [**6**], RADAU5, RADAU and MBDF-DAE (1998) written by Cash [**15**], [**16**] allow the numerical solution of subclasses of DAEs with a given structural form.

The multistep codes are based on backward differentiation formulae, which were the first class of methods considered for the solution of DAEs. Such formulae can be viewed as a way to approximate the derivative in one point and this simplify the theoretical analysis about convergence and stability [**35**], [**6**]. The code DASSL works on implicit DAEs, the only limitation is that it can fail for DAEs of index higher than two. The code must be suitably modified for what concerns the scaling and the error estimation. Moreover the implicit Euler method itself, used to start the computation, is not suitable for high index problems. Usually it is suggested to reduce the index of the problem before solving it numerically. Results about Runge-Kutta methods applied to DAEs are in [**25**], where the code RADAU5 is described. The Runge-Kutta formulae are well suited for solving higher index problems because they do not need starting procedures and, moreover, they have good stability features. The code RADAU5 is designed for solving DAEs up to order three in Hessenberg form. Another important question concerning DAEs is the computation of a complete set of consistent initial values. The continuous problem needs only a subset of them to have locally a unique solution, whereas the numerical methods usually require all the starting values. Recently a class of BVMs, namely the GBDFs, has been analyzed for the solution of this problem [**1**], [**2**], [**36**]. A numerical scheme requiring only the initial values needed by the continuous problem has been analyzed for DAEs in Hessenberg form up to the order three. Such methods are able to gather most of the positive features of BDFs and the Runge-Kutta methods without having some negative ones, such as the instability of BDFs or the reduction of the order depending on the internal stages of some Runge-Kutta methods.

References

[1] P. AMODIO and F. MAZZIA, *Numerical solution of differential algebraic equations and computation of consistent initial/boundary conditions*, J. Comput. and Appl. Math. **8** (1997), 135-146.

[2] P. AMODIO and F. MAZZIA, *An algorithm for the computation of consistent initial conditions for differential-algebraic equations*, Numer. Algorithms **19** (1998), 13-23.

[3] U. ASCHER, J. CHRISTIANSEN and R. D. RUSSELL, *Collocation software for boundary-value odes*, ACM Trans. Math. Softw (7) (1981), 209-222.

[4] G. BADER and U. ASCHER, *A new basis implementation for a mixed order boundary value ode solver*, SIAM J. Sci. Statist. Comput. **8** (1987), 483-500.

[5] C. BENDTSEN, *Parsodes, a parallel stiff ode solver*, Version 1.0, User's Guide, available from netlib, 1996.

[6] K. E. BRENAN, S. L. CAMPBELL and L. R. PETZOLD, *Numerical solution of initial-value problems in differential-algebraic equations*, North Holland, New York 1989.

[7] L. BRUGNANO and D. TRIGIANTE, *High order multistep methods for boundary value problems*, Appl. Numer. Math. **18** (1995), 79-94.

[8] L. BRUGNANO and D. TRIGIANTE, *Block boundary value methods for linear hamiltonian systems*, Appl. Math. Comput. **81** (1997), 49-68.

[9] L. BRUGNANO and D. TRIGIANTE, *On the potentiality of sequential and parallel codes based on extended trapezoidal rules* (ETRs), Appl. Numer. Math. **25** (1997), 169-184.

[10] L. BRUGNANO and D. TRIGIANTE, *Parallel implementation of block boundary value methods on nonlinear problems: theoretical results*, Appl. Numer. Math. **28** (1998), 127-141.

[11] L. BRUGNANO and D. TRIGIANTE, *A new mesh selection strategy for ODEs*, Appl. Numer. Math. **24** (1997), 1-21.

[12] L. BRUGNANO and D. TRIGIANTE, *Solving ODEs by linear multistep initial and boundary value methods*, Gordon and Breach 1998.

[13] K. BURRAGE, *A special family of Runge-Kutta methods for solving stiff differential equations*, BIT **18** (1978), 22-41.

[14] J. C. BUTCHER, *On the implementation of implicit Runge-Kutta method*, BIT **16** (1976), 237-240.

[15] J. R. CASH, *The integration of stiff initial value problems in O.D.E.s using modified extended backward differentiation formulae*, Comput. Math. Appl. **9** (1983), 645-657.

[16] J. R. CASH and S. CONSIDINE, *An MEBDF code for stiff initial value problems*, ACM Trans. Math. Softw. **18** (1996), 142-155.

[17] J. R. CASH and M. H. WRIGHT, *A deferred correction method for nonlinear two-point boundary value problems: implementation and numerical evaluation*, SIAM J. Sci. Statist. Comput. **12** (1991), 971-989.

[18] S. D. COHEN and A. C. HINDMARSH, *CVODE User guide*, Available via WWW at URL http://www.netlib.org/ode/index.html.

[19] T. EIROLA and J. M. SANS-SERNA, *Conservation of integrals and symplectic structure of differential equations by multistep methods*, Numer. Math. **15** (1975), 10-48.

[20] *Mathematical Sciences, Technology and Economic Competitiveness. Board of Mathematical Science, Mathematics and Applications*, National Research Council, Editor J. G. Glimm. National Academy Press, Washington D.C. 1991.

[21] K. GUSTAFSSON and G. SODERLIND, *Control strategies for the iterative solution of nonlinear equations in ODE solvers*, SIAM J. Sci. Comput. **18** (1997), 23-40.

[22] K. GUSTAFSSON, M. LUNDHAND and G. SODERLIND, *A PI stepsize control for the numerical solution of ordinary differential equations*, BIT **18** (1988), 270-287.

[23] K. GUSTAFSSON, *Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods*, ACM Trans. Math. Software **17** (1991), 533-554.

[24] K. GUSTAFSSON, *Control theoretic techniques for stepsize selection in implicit Runge-Kutta methods*, ACM Trans. Math. Software **20** (1994), 496-517.

[25] E. HAIRER, CH. LUBICH and M. ROCHE, *The numerical solution of differential-algebraic systems by Runge-Kutta methods*, Lecture Notes in Math. **1409** Springer Verlag, Berlin 1989.

[26] E. HAIRER, S. P. NORSETT and G. WANNER, *Solving ordinary differential equations I: nonstiff problems*, Springer Series in computational mathematics, Springer Verlag 1993. Second Revised Edition.

[27] E. HAIRER and G. WANNER, *Solving ordinary differential equations II: stiff and differential-algebraic problems*, Springer Series in computational mathematics, Springer Verlag, 1996. Second Revised Edition.

[28] E. HAIRER and G. WANNER, *Stiff differential equations solved by RADAU methods*, J. Comput. Appl. Math. to appear.

[29] A. C. HINDMARSH, *Lsode and lsodi, two new initial value ordinary differential equation solvers*, ACM SIGNUM Newsletter **15** (1980), 10-11.

[30] A. C. HINDMARSH, *Odepack, a systematized collection of ode solvers*, In R. S. Stepelman et al., editor, *Scientific Computing*, North-Holland, Amsterdam 1983.

[31] F. IAVERNARO and F. MAZZIA, *Solving ordinary differential equations by generalized adams methods: properties and implementation techniques*, Appl. Numer. Math. **28** (1998), 107-126.

[32] F. IAVERNARO and F. MAZZIA, *Block-boundary value methods for the solution of ordinary differential equations*, SIAM J. Sci. Comput. (1998) to appear.

[33] F. IAVERNARO and F. MAZZIA, *On the extension of the code GAM for parallel computing*, EURO-PAR'99 Parallel Processing, 1136-1143, Lecture Notes in Computer Science **1685**, Springer, Berlin 1999.

[34] K. R. JACKSON and R. SACKS-DAVIS, *An alternative implementation of variable step-size multistep formulas for stiff odes*, ACM Trans. Math. Software **6** (1980), 295-318.

[35] R. MARTZ, *Numerical methods for differential algebraic equations*, Acta Numerica (1991), 141-198.

[36] F. MAZZIA, *Boundary value methods for the numerical solution of boundary value problems in differential-algebraic equations*, Boll. Un. Mat. Ital. **7** (1997), 579-593.

[37] P. K. MOORE and L. R. PETZOLD, *A stepsize control strategy for stiff systems of ordinary differential equations*, Appl. Numer. Math. **15** (1994), 449-463.

[38] J. J. B. DE SWART, W. M. LIOEN and W. A. VAN DER VEEN, *PSIDE*, November 25, 1998. Available via WWW at URL http://www.cwi.nl/cwi/projects/PSIDE/.

[39] D. TRIGIANTE *Multipoint methods for linear Hamiltonian systems*, Advances in Nonlinear dynamics, Stability Control Theory Methods Appl. **5**, Gordon and Breach, Amsterdam 1997.

Abstract

This review article describes some of the problems connected to the construction of codes for the numerical solution of ordinary differential equations. Our aim is to show, on a concrete example, the complexity of the process which, starting from the continuous problem, arrives at its approximate solution handling a large number of intermediate numerical problems.

* * *