ON THE SOLUTION OF PARABOLIC AND HYPERBOLIC PDE'S BY THE METHOD-OF-LINES APPROACH

François E. Cellier

Institute for Automatic Control, The Swiss
Federal Institute of Technology Zurich,
Physikstr. 3, CH-8006 Zurich, Switzerland

I) ABSTRACT

This tutorial paper surveys the problems arising in the coding and utilization of general purpose packages for the solution of PDE (partial differential equation) problems. It is shown that such packages can never be as general as a package for ODE (ordinary differential equation) problems. There exist, however, enough solvable problems to justify the coding of such general purpose packages using robust methods as the method of lines. Some hints are given on how to select the following parameters:

a) the optimal step size for the integration over time

b) the optimal order of the integration algorithm

c) the optimal class of integration algorithms

d) the optimal grid width for the discretization in space

e) the optimal order of the approximation formulae for the computation of the spatial derivatives.

In this formulation the problem encompasses the ODE systems integration (parameters (a) to (c)) for which reason this simpler problem will be discussed first in a separate chapter.

II) INTRODUCTION

More and more often simulation techniques are applied by scientists having a far better knowledge of the process they are simulating than of the numerical methods behind the tool they use. Especially for this kind of users there exist CSSL-type languages [1] for many years now. These allow the simulation of processes being modeled by a set of first order ODE's freeing the simulation user from deeper knowledge of the simulation techniques behind. They normally have in their "box of tricks" about one dozen of different algorithms for numerical integration and the user is advised by the user's manual to pick out that method best fitting the equations modeling his specific process. In time-uncritical situations (GEAR [2] calls them "trivial" problems) he will, therefore, use the default method of the package, normally being a Runge-Kutta algorithm of 4th order with step size control. He will not bother much about computing time. In time-critical cases he may check through once the few algorithms available and decide according to the results obtained. He will in most cases prefer to use a variable step size algorithm, since it is much more meaningful to him to input an error tolerance than a step size. It would nevertheless be precious for him to have a guidance, which of the offered algorithms is most likely to fit his problem best. Although this question cannot be answered in a conclusive manner, several authors tried already to classify the different *integration methods* and the problems to be solved, and to state which problem is best attacked by which class of algorithms; furthermore to give information on the optimal *order of the algorithm* versus the error tolerance and on the optimal *step size* to be selected in case of fixed step size algorithms. The most comprehensive work on this topic is [2]. It is far beyond the aim of this short paper to summarize the work described in [2]. In a first chapter the author wishes, however, to add some of his own experience in this field. The author wishes to state clearly that his contribution by no means devaluates the work carried out by GEAR [2], which he highly appreciates, but gives just some additional thoughts to a topic for which there does not exist any conclusive theory.

Only recently attempts have been made to tackle in the same manner problems described by PDE's. In [3,4,5,6,7] there are described general purpose packages for the solution of PDE problems. Although the idea is quite apparent, the situation is different from several points of view.

a) The numerical behaviour of PDE's is much more delicate than in the ODE case. It is, therefore, much more important to choose an optimal algorithm. (There do not exist any "trivial" PDE problems.)

b) The algorithms are so differently structured from each other that it seems impossible to offer in one package a conclusive selection of algorithms. For this reason the universal applicability of any such "general purpose" package must be doubted.

c) Since the problems to be solved are always time-critical, application of a robust method is often not practical due to high computational costs, the appropriate selection of the algorithm being most important.

d) If a suitable algorithm has been found, there is still a high degree of freedom in its application (several parameters have to be determined by the user). Therefore, a deep knowledge of the numerical behaviour of the method is an absolute must, this again making the usefulness of such an all-round package more than doubtful. (The aim was primarily to have a tool for the unskilled user, so that he would not have to care about the numerical methods at all.)

For these reasons the author has serious doubts whether such packages will ever be as successful as comparative packages for ODE problems. On the other hand -- when trying to characterize the problems to be solved -- one finds that the same (few) types of equations appear again and again in quite different fields of applications. For this reason it is nevertheless worthwhile to discuss, whether it is possible to define packages suitable at least for one or the other "standard" problem. There exist for example several good programs for stress analysis problems (e.g. [8,9]). These base upon finite elements being the best method so far known for the solution of elliptic PDE problems with complex geometry. For the solution of parabolic and hyperbolic PDE's in one, two and three dimensions with not too exotic boundary conditions there have been coded programs using either specific finite difference schemes (e.g. [6]) or the method-of-lines approach (e.g. [3,4]). The advantage of the former is that for simple examples the convergence range of the algorithm can analytically be computed, the advantages of the latter are:

a) The method of lines is generally more robust (the same algorithm can be applied to a larger group of PDE's, e.g. to parabolic *and* hyperbolic problems).

b) It encompasses ODE problems, thus being more flexible (easy formulation of coupled PDE problems and combined PDE/ODE problems).

c) It allows selection between a large variety of different algorithms by simply leaving the choice between several algorithms for the integration over time and for the differentiation in space.

The author believes that these advantages make the utilization of the method of lines superior for so called general purpose packages, although one certainly can find specific examples where utilization of a specific finite difference scheme results in lower computing time.

The user, however, faces the problem to set several parameters and needs a guidance on how to do this. These parameters have already been mentioned in the abstract of this paper. Parameters (a) to (c) are the same as in the ODE case, whereas parameters (d) and (e) are additional parameters for PDE problems only. In the ODE case it is still possible to check all available algorithms through; in the PDE case there are too many parameters to be selected which makes the suggestion to check through all possibilities no longer feasible. The aim of this paper is to give some guidelines to this problem.

The author believes that there exist enough problems which can be solved by these robust methods to justify the effort for coding a "general purpose" package, knowing that no such package will ever be capable of coping with all imaginable situations. Many problems are so time-critical that they can only be solved by algorithms developed specially for that specific purpose, e.g. by semi-analytical methods such as singular perturbations or conformal mapping techniques. These methods, however, depend so heavily upon the problem equations and upon the problem geometry that they cannot be implemented within a general purpose package.

III) ON THE SELECTION OF STEP SIZE, ORDER AND METHOD FOR THE INTEGRATION OF ODE'S

1) Step size selection:

To predict the required step size, is a difficult problem. The solution to it may, however, be automized. For higher order algorithms the additional costs for error estimation and step size control are neglectable. A Runge-Kutta algorithm of 5th order, for example, needs at least 6 function evaluations per integration step. By using the Fehlberg-coefficients [10], error estimation is possible without need for even one single additional function evaluation to be carried out by comparing the 5th order algorithm to an embedded 4th order algorithm. For this reason the author suggests to implement all algorithms of at least third order as variable step size algorithms for use in a general purpose simulation package. For lower order algorithms the additional costs for the error estimation may not pay out.

2) Selection of the order:

Higher order algorithms are generally favorable if higher accuracy is required. Comparing for instance two different Runge-Kutta-Fehlberg algorithms one of 5th and the other of 8th order, the former will normally be faster for accuracy requirements of 10^{-5} or less, whereas the latter will execute faster otherwise. This number is, of course, somewhat dependent of the problem to be solved and also of the length of mantissa of the computer in use.

There exist algorithms with variable order as well as variable step size. Multistep methods of this kind have been reported in [2]. Recently there have also been published variable order Runge-Kutta-Fehlberg algorithms by BETTIS [11] using the Fehlberg technique of embedding different Runge-Kutta algorithms into each other using the same points for function evaluations. These algorithms in the future are expected to dominate more and more in general purpose simulation packages, since they free the user from deciding on quantities of which he has no knowledge.

3) The integration method:

GEAR [2] shows for some examples that one-step methods are in favor over multistep methods for low to medium accuracy requirements. According to the authors experience this comparison can, however, not be generalized. Runge-Kutta algorithms can give better results in many applications even for higher accuracy requirements, if the order of the algorithm is raised and if some sophistication is put into the step size control mechanism being the strong point about one-step methods. We normally use Runge-Kutta algorithms of 5th and 8th order using the Fehlberg coefficients [10] for sophisticated error estimation and then *compute* the step size according to the following formula:

$$dt_{new} = dt_{old} \cdot \left(\frac{tol}{20 \cdot \varepsilon}\right)^{1/order} \qquad (1)$$

where: dt_{new} = step size suggestion for next step

dt_{old} = step size of current step

tol = required accuracy

ε = local maximum norm integration error computed by Fehlberg formulae

$order$ = order of integration algorithm

This (unpublished) formula has been found by several researchers from the Dept. of Aerospace Engineering of the University of Texas at Austin in a heuristic way. It gives a high degree of step size adaptation. The advantage of this method is that the step size is directly computed and not only halved or doubled as in earlier described algorithms. By using this technique Runge-Kutta comes far better off than described in [2]. Many problems require frequent modification of the step size. For these problems the multistep methods show much poorer results than for those examples given in [2]. For each step size modification these methods need to be restarted which is a rather time consuming procedure. For this reason one will try to integrate with a step size which is much smaller than that necessary for the required accuracy so that one is not forced to update the step size when the behaviour of the system changes within certain bounds. For such problems the Runge-Kutta algorithms will even be superior for high accuracy requirements.

On the other hand we found that in many applications the Runge-Kutta algorithms become instable prior to becoming unaccurate. In such cases multistep methods of low order execute faster than any Runge-Kutta algorithms for low accuracy requirements. For these reasons the author believes that it is not possible to state in a general manner that for certain accuracy requirements one-step algorithms are preferred over multistep methods and vice versa. What can certainly be stated, however, is that higher order algorithms should be taken for higher accuracy requirements, whereas low order algorithms execute faster for low accuracy requirements. It is furthermore possible to generalize, that *multistep methods* will be favorable for *smooth problems* which do not require frequent updating of the step size and also for *stiff systems* where the Runge-Kutta algorithms tend to be costly due to stability constraints, whereas *one-step methods* will turn out to be cheaper in all cases where *frequent updating of the step size* is required and which are *not too stiff*. The determination of the best suited algorithm for *stiff and non-smooth systems* is an unsolved problem which, however, turns out to be an important case, as will be seen later.

IV) TREATMENT OF DISCONTINUITIES

In engineering applications most system definitions include discontinuities. These may be functions of time (e.g. pulse generator) or functions of state variables (e.g. hysteresis function). PRITSKER [12] calls the former *time events* and the latter *state events*. Such discontinuities should be treated in a mathematically proper way and not -- as in most of the continuous simulation packages -- by the integration step size control algorithm. This has been shown in [3,11,12,13,14]. This means that integration

should be restricted to continuous subregions and that in the case of state events an iteration procedure is involved for proper location of the event. Good general purpose packages should thus provide facilities for proper coding of events. For such cases one-step methods are preferable, since:

a) during the iteration procedure the step size has to be modified for each successive step

b) after each event being accomplished the algorithm has entirely to be restarted.

CARVER [13] shows that the iteration procedure requires fewer number of steps when using Nordsieck methods (since higher derivatives are available there). The author, however, believes that -- except for stiff systems -- this trade off will not compensate the higher costs for step size updating of these algorithms. For such problems a one-step algorithm would be needed with good numerical behaviour for integrating stiff systems. New results on this topic have been published by FRIEDLI [15] who developed generalized Runge-Kutta algorithms. His methods are extremely well suited for stiff problems and at the same time are one-step algorithms. They, however, require on-line computation of the state transition matrix which makes the costs per step so high, that they hardly can be justified except for very stiff situations. Other publications present low order Runge-Kutta methods with extended stability range. The author, however, believes that the best algorithms for this class of problems so far known are the IMPEX algorithms described in [16,17]. He is, however, sure that substantial improvements still are possible.

V) FORMULATION OF THE METHOD OF LINES APPROACH

Let a PDE of the following form be given:

$$\frac{\partial \underline{u}(x,t)}{\partial t} = \underline{f}\left\{\underline{u}(x,t), \frac{\partial \underline{u}(x,t)}{\partial x}, \frac{\partial^2 \underline{u}(x,t)}{\partial x^2}, x, t\right\} \qquad (2)$$

$$t \in [0,\infty) \quad x \in [0,L] \quad \underline{u} \in \mathbb{R}^n$$

and:

$$B_1(t) \cdot \frac{\partial \underline{u}(x,t)}{\partial x} + B_2(t) \cdot \underline{u}(x,t) = B_3(t) \qquad (3)$$

$$t \in [0,\infty) \quad x = \{0,L\}$$

and:

$$\underline{u}(x,t=0) = \underline{u}_o(x) \qquad (4)$$

Eq.(2) describes the system, eq.(3) its boundary conditions and eq.(4) its initial conditions. This PDE can now be discretized in space:

$$x \rightarrow \underline{\xi} = \{1,2,\ldots,m\} \text{ (indices)} \qquad (5)$$

resulting in m ODE's of first order:

$$U_t(t) = F\{U(t), U_x(t), U_{xx}(t), \underline{\xi}, t\} \qquad (6)$$

$$t \in [0,\infty) \quad \underline{\xi} = \{1,2,\ldots,m\} \quad U \in \mathbb{R}^{n \times m}$$

It is, of course, possible to extend the above given transformation to more complex PDE problems (higher order derivatives in time, mixed derivatives, higher order derivatives in space, several space dimensions, multi-region problems etc.). In this new formulation the problem can be split up into three simpler problems:

a) computation of the space derivatives $U_x(t)$ and $U_{xx}(t)$ by difference schemes out of U(t) for given values of time t

b) computation of the state derivatives $U_t(t)$ from eq.(6)

c) computation of U(t+dt) by carrying out one step of integration over time.

In this formulation it is evident that there is no difference in simulating only one PDE or systems of several coupled PDE's combined with ODE's. It is furthermore clear that different algorithms may be obtained by simply changing either the integration algorithm or the difference scheme or both and that a variety of different algorithms can be offered in a general purpose package in the same manner as it is done in ODE packages.

VI) SELECTION OF INTEGRATION ALGORITHM, GRID WIDTH AND DIFFERENCE SCHEME FOR THE INTEGRATION OF PARABOLIC AND HYPERBOLIC PDE'S

1) Integration algorithm:

Principally this problem has already been discussed in section III. There can, however, something be added.

Let us consider a PDE of form eq.(6). For the k-th u(t) called $u_k(t)$ we can write:

$$u_k(t) = 0.5 \cdot \{u_{k-1}(t) + u_{k+1}(t)\} + o(\Delta x) \qquad (7)$$

where Δx means the grid width of the discretization in the space dimension. For Δx being sufficiently small the above equation describes an almost linear dependency of neighbouring functions, which implies that the eigenvalues of the system will be wide spread. A system of ODE's resulting from the discretization of a PDE will, therefore, almost by definition form a *stiff system*.

For this reason the author highly recommends the Hindmarsh implementation of the Gear algorithm [18] for smooth PDE problems. For non-smooth systems or systems with discontinuities we face the problem described earlier for which no satisfactory solution has been found yet.

2) Grid width of the discretization in space:

For high accuracy requirements and for non-smooth problems Δx should be made sufficiently small to obtain reasonable results.

Let us consider a parabolic PDE of the form:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \cdot \sigma \qquad (8)$$

(diffusion problem). When using the Euler method for integration and a three-point-central difference scheme for the computation of the second spatial derivative one can show analytically, that for Δx being halved, the stability domain of dt will be divided by a factor of 4. In this case the amount of work (CPU-time) will be multiplied by 8, since there are in the system now the double number of integrators proceding with one fourth of the former step size. Although it is not possible to give an explicitely computed formula as above for more complex algorithms, experiments have shown that the situation is similar. For this reason it is advisable to select a smaller number of divisions where possible and instead use a higher order of difference scheme. This has been stated also by CARVER [4]. He found that for most applications it is useful to set the grid width to such a value that one obtains between 11 and 33 discretization points per space dimension. This agrees well with the results found by us.

From the above one can further conclude that for higher accuracy requirements, for which a rather narrow grid is needed, it becomes more and more important to select the integration algorithm carefully, since the problem gets more and more stiff.

Furthermore, since there is no grid width control, the user has no guarantee for obtaining meaningful results. He is -- as in the case of fixed step size integration algorithms -- bothered with the request to decide upon a quantity for which he has essentially no feeling. It would, of course, be possible to define a grid width control algorithm in the same way as it is done in

integration by comparing difference schemes of different order to each other -- one would then be bothered by a variable number of ODE's and for each grid width modification one would require an interpolation procedure to obtain estimates for the new $u_i(t)$ values required. To the author's knowledge this has never been realized so far, probably because:

a) the algorithm would be complicated and time consuming

b) one would have to control a quantity for which there exists much less freedom than in the integration step size control (since from above the grid width may not become too small).

For these reasons it would be advisable to develop (if at all) a *combined grid width and order control* where primarily the order is modified and if required the grid width as well. As in the case of integration this would free the user from deciding upon quantities on which he has essentially no knowledge. Such an algorithm would require a similar decision logic as the variable order, variable step size Adams method for integrating ODE's described in [2], which has partly been developed for the same reasons (the step size should not be modified too often). The author is sure, that there can be done significant improvements in this field. Such an algorithm would be specially valuable for general purpose packages, since the unskilled user with the available packages always bears the risk to trust in results which have nothing in common with reality.

3) Order of the difference scheme:

Many applications have shown that an optimal selection is to choose a difference scheme which involves about the same number of neighbouring points as the order of the integration algorithm. This should furthermore be an odd number to obtain a central method (for higher accuracy). Accordingly we normally use a five-point-central method in connection with a Runge-Kutta algorithm of 5th order and in connection with the Gear-Hindmarsh algorithm, whereas we use a seven-point-central method in connection with a Runge-Kutta algorithm of 8th order and with Gear's implementation of the variable order Adams algorithm. A three-point formulae is used for all lower order integration algorithms.

VII) TREATMENT OF DISCONTINUITIES

In the PDE case there exist more different kinds of discontinuities. The equations themselves can show discontinuities as in the ODE case. These can be solved in precisely the same manner as described in section IV. On the other hand there may be discontinuities introduced by the boundary or initial conditions (we call them geometry-discontinuities). These cannot be treated in the same manner as before. They normally tend to "walk" through space, which means, that a discontinuity having entered the system will continuously be present over a specific period of time. It thus cannot be treated as an event. Geometry-discontinuities in the initial conditions can be thought of as discontinuities of the boundary conditions having entered the system sometimes earlier. They can, therefore, be treated both the same way.

In the discretized formulation of eq.(6) the situation is again different from the original PDE formulation, in that it is not evident, what a geometry-discontinuity precisely means. The normal definition of a discontinuity (non-existence of the first derivative) must fail, since the function in this formulation is no longer continuous in space, but consists of a series of discrete points.

It is well known that any difference scheme to obtain the slope of a function must fail when applied over a discontinuity. For higher derivatives the situation gets worse and worse. The difference scheme, however, will give same results whether applied to a continuous function ar to a discrete function being defined only at those points, where function evaluations are required. It is, therefore, obvious that the difference scheme will produce nonsense if applied over a range of points of the discrete function, between which the original PDE solution shows a discontinuity.

A proper solution would be to involve an iteration algorithm to detect where (at each instant of time t) the discontinuities are, then to split up the definition range in the space dimensions into several subregions without discontinuities and to compute the spatial derivatives for these regions separately. This procedure is, however, rather complicated, since we then have a variable number of discretization points for each subregion. Special problems will then arrise for discontinuities being located close to the border of the entire definition range. For lower accuracy requirements it will, therefore, be better to solve the problem in a different way. When using e two-point formulae for computation of the spatial derivatives we obviously avoid all troubles, since the discontinuity can never lie within the range of points in use. The second order spatial derivatives have then to be computed by computing two times the first derivative.

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial x}\left(\frac{\partial u}{\partial x}\right) \qquad (9)$$

This is, of course, only possible for low accuracy requirements,since the approximation of the spatial derivatives will only be of first order.

For the integration over time the situation has not changed, except that no iteration of events can help us out of the troubles, since the discontinuity is continuously present. We thus will have to make sure that discontinuities appear only *between* successfully computed integration steps and then use a one-step method for integration. This can be achieved by computing new values for the spatial derivatives only between integration steps but not for intermediate computations.

For many PDE problems the state derivatives depend only upon spatial derivatives (e.g. eq.(8)).If spatial derivatives are computed only between integration steps, the state derivatives would then be constant over one integration step and we can as well use Euler for integration as any other more sophisticated method. One can see that, if ever possible, such problems should be treated by using the simplest algorithms available. If this method fails being too time consuming (higher accuracy requirements) a substantial effort is required to find a better solution to the problem. It is then doubtful whether one should use the method of lines at all.

Looking at the different classes of PDE problems, parabolic PDE's in most cases behave numerically well. They generally have a high damping factor which tries to wipe out discontinuities as they arise. It is essential to compute the spatial derivatives carefully (not include any discontinuities), whereas the integration is not so critical. Hyperbolic PDE's involving discontinuities are much more difficult to solve. The author, however, has not yet enough experience to give any guidance on how to solve such problems.

VIII) CONCLUSIONS

An attempt has been made to survey the problems arising with the coding and utilization of general purpose simulation packages for the solution of parabolic and hyperbolic PDE problems. The author knows very well that this topic cannot be treated in a conclusive manner. The subject is likely to raise emotions, since for each single statement it is relatively easy to find a counterexample proving the opposite. For these reasons the author wants to state that this survey is more a synopsis of some ideas and certain approaches which have been found effective, rather than a remedy curing all diseases.

IX) ACKNOWLEDGMENTS

I would like to express my gratitude towards Prof.Dr.M.Mansour who is heading out institute and who gave me the opportunity to carry out the work described herein, I furthermore would like to thank Prof.Dr.h.c.E.Gerecke, the former head of our institute, for his continuous encouraging support with algorithm-neck-breaking PDE problems.

X) REFERENCES

[1] The SCi Continuous System Simulation Language (CSSL) *Simulation Vol. 9 No. 6* 1967 (December)

[2] C.W.Gear: Numerical Initial Value Problems in Ordinary Differential Equations. *Prentice Hall Series in Automatic Computation* 1971.

[3] F.E.Cellier, Blitz A.E.: GASP-V: A Universal Simulation Package. *Proc. of the 8th AICA congress on simulation of systems* North-Holland Publishing Company 1976.

[4] M.B.Carver: FORSIM: A FORTRAN Package for the Automated Solution of Coupled Partial and/or Ordinary Differential Equation Systems. User's Manual. *Atomic Energy of Canada Ltd., Chalk River Nuclear Laboratories, Chalk River, Ontario, Canada* 1974

[5] W.E.Schiesser: LEANS-III, User's Manual. *Dept. of Chemical Engineering, Whitaker Laboratory, Lehigh University, Bethlehem Pennsilvania 18015, USA* 1971

[6] M.G.Zellner: DSS: Distributed System Simulator. User's Manual (PhD-Thesis 1970) *c/o Prof.Dr.W.E.Schiesser, Dept. of Chemical Engineering, Whitaker Laboratory, Lehigh University, Bethlehem Pennsylvania 18015, USA.*

[6] A.F.Cardenas, Karplus W.J.: PDEL - A Language for Partial Differential Equations. *Communications of the ACM. Vol. 13, No. 3* 1970 (March) pp. 184-191.

[7] K.J.Bathe: The SAP-IV User's Manual. *IPSI, 127, Boulevard Rodin, 92130 Issy les Moulineaux, France.*

[8] K.J.Bathe: The NONSAP User's Manual. *IPSI, 127, Boulevard Rodin, 92130 Issy les Moulineaux, France.*

[9] D.G.Bettis: Efficient embedded Runge-Kutta methods. *Proc. der Arbeitstagung über numerische Behandlung von Differentialgleichungen.* Springer Verlag, Lecture Notes in Mathematics 1977. (Tagung: Oberwolfach 1976)

[10] E.Fehlberg: Classical 5th-,6th-,7th-, and 8th-order Runge-Kutta Formulas. *Report: NASA TR R-287*

[11] A.A.B.Pritsker: The GASP-IV Simulation Language. *John Wiley* 1974.

[12] F.E.Cellier, Rufer D.F.: Algorithm Suited for the Solution of Initial Value Problems in Engineering Applications. *Proc.: SIMULATION'75, Zurich, Switzerland, P.O.Box 354, 8053 Zurich, Switzerland* 1975

[13] M.B.Carver: Efficient handling of discontinuities and time delays in ordinary differential equations. *Proc.: SIMULATION'77 Montreux, Switzerland, P.O.Box 354, 8053 Zurich Switzerland* 1977.

[14] R.Mannshardt: Simulation of discontinuous systems by use of Runge-Kutta methods combined with Newton iteration. *Proc.: SIMULATION'77, Montreux, Switzerland, P.O.Box 354, 8053 Zurich, Switzerland* 1977

[15] A.Friedli: Verallgemeinerte Runge-Kutta Verfahren zur Lösung steifer Differentialgleichungssysteme. *Proc. der Arbeitstagung über numerische Behandlung von Differentialgleichungen.* Springer Verlag, Lecture Notes in Mathematics 1977. (Tagung: Oberwolfach 1976)

[16] B.Lindberg: IMPEX - A Program for Solution of Systems of Stiff Differential Equations. *The Royal Institute of Technology Stockholm Sweden. Report: NA 72.50* 1972

[17] B.Lindberg: IMPEX 2 - A Procedure for Solution of Systems of Stiff Differential Equations. *The Royal Institute of Technology Stockholm Sweden. Report: TRITA-NA-7303* 1973.

[18] A.C.Hindmarsh, Gear C.W.: Ordinary Differential Equation System Solver. *Lawrence Livermore Laboratory. Report: UCID 30001, Rev.2* August 1972.