CONFERENTIE VAN NUMERIEK WISKUNDIGEN

7 oktober - 9 oktober 1991

CONFERENTIEOORD WOUDSCHOTEN
ZEIST

Werkgemeenschap Numerieke Wiskunde

CONFERENTIE VAN NUMERIEK WISKUNDIGEN

Werkgemeenschap Numerieke Wiskunde

ZESTIENDE CONFERENTIE NUMERIEKE WISKUNDE

Doel van de conferentie

De Conferentie Numerieke Wiskunde wordt eenmaal per jaar gehouden onder auspiciën van de Werkgemeenschap Numerieke Wiskunde. Het doel van de conferentie is kennis te nemen van recente ontwikkelingen binnen de numerieke wiskunde. Hiertoe worden jaarlijks twee thema's vastgesteld. Enige buitenlandse deskundigen worden uitgenodigd over deze thema's lezingen te houden.

Thema's

A. Het oplossen van evolutieproblemen met behulp van waveform relaxatie en andere (parallelle) methoden.

B. Eigenwaardenbepaling bij grote matrices, in het bijzonder niet-symmetrische matrices.

Organisatie

De organisatie is in handen van de voorbereidingscommissie bestaande uit M.H.C. Paardekooper (KUB)(voorzitter), Th.J. Dekker (UvA), P.J. van der Houwen (CWI) en B.P. Sommeijer (CWI)(secretaris). Medewerking is verleend door Sasha A.I. Dees (CWI) en het Centrum voor Wiskunde en Informatica. Financiële ondersteuning is gegeven door NWO via de Vertrouwenscommissie van het Wiskundig Genootschap.

Sprekers

Thema A.	A. Bellen, University of Trieste (Italy)
	J.C. Butcher, University of Auckland (New Zealand)
	C.W. Gear, NEC Research Inst. Inc. (U.S.A.)
	A.E. Ruehli, IBM Watson Research Centre (U.S.A.)
Thema B.	F. Chatelin, IBM France (France)
	L. Elsner, University of Bielefeld (Germany)
	B. Kagström, University of Umea (Sweden)

Een korte voordracht zal worden gegeven door

S. Vandewalle, Katholieke Universiteit Leuven

Programma (de laatste 10 minuten van elke voordracht zijn gereserveerd voor discussie)

Maandag 7 oktober

10.00 - 11.10	aankomst, koffie	14.15 - 15.15	F. Chatelin
11.10 - 11.15	opening	15.15 - 15.45	thee
11.15 - 12.15	C.W. Gear	15.45 - 16.45	A. Bellen
12.45	lunch	16.50 - 17.50	B. Kagström
		18.15	diner

Dinsdag 8 oktober

8.00	ontbijt	14.15 - 15.15	C.W. Gear
9.00 - 10.00	J.C. Butcher	15.15 - 15.45	thee
10.00 - 10.30	koffie	15.45 - 16.45	F. Chatelin
10.30 - 11.30	L. Elsner	16.50 - 17.20	S. Vandewalle
11.35 - 12.35	A.E. Ruehli	17.25 - 17.35	VergaderingWerkgemeenschap Numerieke Wiskunde
12.45	lunch		
		18.00	diner

Woensdag 9 oktober

8.00	ontbijt	13.45 - 14.45	L. Elsner
9.00 - 10.00	A. Bellen	14.50 - 15.50	A.E. Ruehli
10.00 - 10.30	koffie	15.50	sluiting, thee, vertrek
10.30 - 11.30	B. Kagström		
11.35 - 12.35	J.C. Butcher		
12.45	lunch		

De bar is geopend van 17.00-18.00 uur en van 21.00-24.00 uur.

Titels en Samenvattingen Voordrachten

Maandag 7 oktober

11.15	C.W. Gear:	Massive parallelism across space in ODEs.
14.15	F. Chatelin:	Large nonsymmetric eigenproblems, Part I: the departure from normality.
15.45	A. Bellen:	Numerical waveform relaxation methods, I.
16.50	B. Kagström:	The generalized Schur decomposition of an arbitrary pencil A - λB, Part I: Theory and Algorithms.

Dinsdag 8 oktober

9.00	J.C. Butcher:	Multistage methods for parallel computation, I.
10.30	L. Elsner:	New results on the QR-algorithm and related algorithms, I.
11.35	A.E. Ruehli:	Waveform relaxation for circuit simulation.
14.15	C.W. Gear:	Parallelism across time in ODEs.
15.45	F. Chatelin:	Large nonsymmetric eigenproblems, Part II: the Arnoldi-Tchébycheff method.
16.50	S. Vandewalle:	On waveform relaxation methods for solving parabolic partial differential equations.

Woensdag 9 oktober

9.00	A. Bellen:	Numerical waveform relaxation methods, II.
10.30	B. Kagström:	The generalized Schur decomposition of an arbitrary pencil A - λB, Part II: Software and Applications.
11.35	J.C. Butcher:	Multistage methods for parallel computation, II.
13.45	L. Elsner:	New results on the QR-algorithm and related algorithms, II.
14.50	A.E. Ruehli:	Recent progress in waveform relaxation.

Numerical Waveform Relaxation Methods

by

A. BELLEN
Dipartimento di Scienze Matematiche
Universita' di Trieste

The waveform relaxation methods (WR) for the solution of the initial value problem for system of ordinary differential equations

$$(1)\qquad \begin{aligned} &dy(t)/dt=f(t,y(t)), \qquad t_0\le t\le t_f,\\ &y(t_0)=y_0, \end{aligned}$$

where $f:[t_0,t_f]\times R^m\rightarrow R^m$, consist in various modifications of the Picard Lindelof iteration

$$\frac{dy^{k+1}(t)}{dt} = f(t,y^k(t)), \qquad y^k(t_0)=y_0;$$

$y^0(t)=\text{const}=y_0$.
Modifications are essentially based on some splitting of y in the right hand side of f in (1), such as WR-Jacobi

$$\frac{d\,y_i^{k+1}}{dt} = f(t,y_1^k,\dots,y_{i-1}^k, y_i^{k+1},y_{i+1}^k\dots,y_m^k), \qquad y_i^{k+1}(t_0)=y_{0,i}$$

or WR-Gauss Seidel

$$\frac{d\,y_i^{k+1}}{dt} = f(t,y_1^{k+1},\dots,y_i^{k+1},y_{i+1}^k\dots,y_m^k), \qquad y_i^{k+1}(t_0)=y_{0,i}$$

Such strategies, also called "dinamical iterations" were first introduced for practical purposes by Lalerasmee [10] ,Lalerasmee-Ruheli-Vincentelli [11] (see also [19]) in the simulation of large dynamical systems and electrical circuits. In order to increase the convergence rate of $\|y^k-y\|_\infty$ to 0, further modifications were given,

such as WR-SOR, WR-Newton and more recently, WR with overlapping in Jeltsch-Pohl [9].
The common goal of all WR methods is decoupling the system so as to isolate possible *stiff* components (or subsystems) which will be treated by appropriate (implicit) numerical methods.

In the analysis and implementation of any numerical method based on WR one is faced with the following theoretical and practical problems:

1-the convergence of the continuous-time iteration (1) or its modifications;

2-the choice of the optimal lenght of the integration "window" on which to iterate untill the process has converged, before moving to the next window;

3-the choice of some *continuous numerical* method for the actual integration of each component (or subsystem) across the window;

4-the convergence of the resulting numerical WR and the analysis (order and stability) of the limit method;

5-parallelism.

Points 1 and 2 have been investigated, mainly for linear systems, by Nevalinna in[16] [17] and Nevanlinna-Miekkala in [13] [14], and by Skeel [18], Gear-Juang [6], Lie-Skalin [12] and others, while points 3 and 4 have been recently developped by Bellen-Jackiewicz-Zennaro in [2],[3],[4] and by Bellen-Zennaro in [5]. Finally, parallel aspects of WR methods has been considered by many authors, in particular by Gear in the survey paper [7] and in [8], by Mitra in [15] and by Bellen-Tagliaferro in [1]

References.

1. A. Bellen & F. Tagliaferro: *A combined WR-parallel steps method for ordinary differential equations*, Proc. "Parallel Computing: Achievements, Problems and Prospects", Capri, June 3-7, 1990. To appear
2. A. Bellen, Z. Jackiewicz & M. Zennaro: *Stability analysis of time-point relaxation Heun method*, Report "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo" n. 1/30, 1990
3. A. Bellen, Z. Jackiewicz & M. Zennaro: *Contractivity of waveform relaxation Runge-Kutta methods for dissipative systems in the maximum norm*, Report Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo" n. 1/61, 1991

4. A. Bellen, Z. Jackiewicz & M. Zennaro: *Time point relaxation Runge-Kutta methods for ordinary differential equations*, preprint
5. A. Bellen, M. Zennaro: *The use of Runge-Kutta formulae in waveform relaxation methods;* preprint;
6 C.W.Gear, F.L.Juang: *The speed of waveform relaxation for ODEs.* In Applied and Industrial Mathematics, Kluwer Academic Pub. Netherlands, 1991
7 C.W.Gear *Parallel methods for ordinary differential equations.* Calcolo 25,(1988) 1-20.
8 C.W.Gear *Massive parallellism across space in ODEs;* preprint
9 R. Jeltsch & B. Pohl: *Waveform relaxation with overlapping splittings*, Report n. 91-02, ETH Zurich, 1991
10 E. Lelarasmee: *The waveform relaxation method for the time domain analysis of large scale nonlinear dynamical systems*, Ph. D. Thesis, University of California, Berkeley, 1982
11 E. Lelarasmee, A. Ruehli & A. Sangiovanni-Vincentelli: *The waveform relaxation method for time domain analysis of large scaled integrated circuits*, IEEE Trans. on CAD of IC and Syst. 1, 131-145 (1982)
12 I. Lie & R. Skalin: *Relaxation based integration by Runge-Kutta methods and its application to the moving finite element method*, Preprint
13 U. Miekkala & O. Nevanlinna: *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Stat. Comp. 8, 459-482 (1987)
14 U. Miekkala & O. Nevanlinna: *Sets of convergence and stability regions*, BIT 27, 557-584 (1987)
15 D. Mitra: *SeAsynchronous relaxations for the numerical solution of differential equations by parallel processors* SIAM J.SCI.STAT.COMP. 8 n; 1(1987) 543-558.
16 O. Nevanlinna: *Remarks on Picard-Lindelof iteration*, Part I, BIT 29, 328-346 (1988); Part II, BIT 29, 535-562 (1989)
17 O. Nevanlinna: *Linear acceleration of Picard-Lindelof iteration*, Numer. Math. 57, 147-156 (1990)
18 R.D.Skeel: *Waveform iteration and shifted Picard splitting.* SIAM J.SCI.STAT.COMP. 10 n; 4 (1989) 756-776.
19 J. White, A. Sangiovanni-Vincentelli, F. Odeh & A. Ruehli: *Waveform relaxation: theory and practice*, Trans. of the Soc. for Computer Simulation 2, 95-133 (1985)

Multistage methods for parallel computation

J. C. Butcher

The serving of *dimsim* in parallel is an efficient solution to the problem of catering to the needs of a large number of diners in a Cantonese restaurant. For the solution of initial value problems, the use of DIMSIMs (diagonally implicit multistage integration methods) is proposed as a component in the overall parallelisation of the computation. While wave-form relaxation achieves parallelism across the system, this should be combined with parallelism across time and this is where it is hoped DIMSIM methods can play a part.

General linear methods were proposed in 1965 [3] as a single comprehensive theory for describing and analysing many known numerical methods as well as some new hybrid methods proposed at that time by C. W. Gear [10] and others. Although theoretical considerations of general linear methods have been studied by many authors (see the bibliography in the present author's survey paper [5] and his monograph [6]), they have not yet made a significant impact on practical computation.

Within the large class of general linear methods, DIMSIMs are identified as a particularly interesting subclass. Within DIMSIMs themselves is a smaller class for which parallel computation can be exploited and the aim of this lecture is to consider to what extent methods of this type can provide potentially useful algorithms for parallel implementation.

The first part of the lecture will deal with stability, accuracy and implementation questions for general linear methods. From these considerations, DIMSIM methods will arise as natural candidates for practical use and the second part of the lecture will centre on these methods, particularly on the subclass suitable for parallel implementation.

The list of references below contains material suitable for background reading as well as items dealing specifically with general linear methods.

References

[1] R. Alexander, Diagonally implicit Runge-Kutta methods for stiff ODEs, *SIAM J. Numer. Anal.* **14** (1977), 1006 - 1021.

[2] K. Burrage and J. C. Butcher, Non-linear stability of a general class of differential equation methods, *BIT* **20** (1980), 185 - 203.

[3] J. C. Butcher, On the convergence of numerical solutions to ordinary differential equations, *Math. Comp.* **20** (1966), 1 - 10.

[4] J. C. Butcher, The order of numerical methods for ordinary differential equations, *Math. Comp.* **27** (1973), 793 - 806.

[5] J. C. Butcher, General linear methods: a survey, *Appl. Numer. Math.* **1** (1985), 273 - 284.

[6] J. C. Butcher, *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*, (1987), John Wiley and Sons, Chichester and New York.

[7] J. C. Butcher, Linear and non-linear stability for general linear methods, *BIT* **27** (1987), 182 - 189.

[8] J. C. Butcher, The equivalence of algebraic stability and AN-stability, *BIT* **27** (1987), 510 - 533.

[9] R. Frank, J. Schneid and C. W. Ueberhuber, The concept of B-convergence, *SIAM J. Numer. Anal.* **18** (1981), 753 - 780.

[10] C. W. Gear, Hybrid methods for initial value problems in ordinary differential equations, *SIAM J. Numer. Anal.* **2** (1965), 69 - 86.

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

[12] A. Iserles and S. P. Nørsett, On the theory of parallel Runge-Kutta methods, *IMA J. Numer. Anal.* **10** (1990), 463 - 488.

Nonsymmetric eigenvalues problems

Françoise Chatelin[*]

1. **Algorithms**
   - subspace iterations, QR, Arnoldi-Tchébycheff
2. **Error bounds and condition numbers**
   - simple and multiple eigenvalues
   - spectra and pseudospectra
   - simulation by random perturbations
3. **Influence of the departure from normality**
   - iterative methods for $Ax = b$
   - Arnoldi and the computation of spectra of highly nonnormal matrices

References

F. Chatelin *Valeurs propres de matrices*, Masson 1988.

F. Chatelin *Spectral approximation of linear operators*, Academic Press 1983.

[*]IBM-FRANCE, CEMAP, 68-76 Quai de la Rapée, 75592 Paris cedex 12, France. (chatelin@fribm11.bitnet) and University of Paris IX Dauphine.

The QR–algorithm and newer developments

Ludwig Elsner
Fakultät für Mathematik
Universität Bielefeld
Postfach 8640
D-4800 Bielefeld 1, FRG
(0521) 106-4800 Bielefeld 1
umatf105@dbiuni11.bitnet

September 9, 1991

In these lectures we shall treat iterative methods for the eigenvalue problem for general matrices.

In the first part we discuss the QR–algorithm. Let a matrix $A = A_0$ be given whose eigenvalue we want to find.

The QR–algorithm is of the form:

For $i = 0, 1, \dots$

1. Decompose $p_i(A_i) = Q_i R_i$, where $p_i(x)$ is some simple polynomial (usually of degree 1 or 2), into a product of an orthogonal matrix Q_i and upper triangular R_i.
2. Form $A_{i+1} = Q_i^H A_i Q_i (= R_i A_i R_i^{-1})$.

In particular, we discuss the following features, which are crucial for the "success" of this algorithm:

- invariance of Hessenberg form
- invariance of symmetry
- implicit calculation of A_{i+1}
- shift strategies
- deflation

After this part, which is quite standard and can be found also in books, like

- Golub–Van Loan: *Matrix computations*

- D.S. Watkins: *Fundamentals of Matrix computations*

we turn to related and more recent algorithms.

One reason not to use QR is that certain symmetries are destroyed by using QR.

In optimal control Hamiltonian matrices arise. Here A is Hamiltonian, if JA is symmetric, $J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}$. The so-called SR-algorithm, based on the SR-decomposition (instead of QR-decomposition) leaves Hamiltonians invariant. Also here the "good" properties of QR can be carried over in a modified way.

There are very many algorithms of this form, preserving symmetries and saving thus storage and computing time.

Another reason to look for other algorithms is that the QR-decomposition is not inexpensive. The LR-algorithm, based on the decomposition into a product of lower and upper triangular matrices (and in fact being the historical precedessor of the QR-algorithm), is cheaper, but can be instable. In a recent paper, Watkins-Elsner considered "Algorithms of decomposition type". Here it is only required that there is a well-defined function mapping each nonsingular matrix A into a nonsingular matrix $G(A)$, s.t. $G^{-1}(A)A$ is upper triangular, i.e. $A = GR$. Under not too strong assumptions convergence can be shown. Examples are LR decomposition with partial pivoting and the so-called chasing algorithms.

Massive parallelism across space in ODEs Extended Abstract

C. W. Gear
NEC Research Institute
4 Independence Way
Princeton, NJ 08540

August 12, 1991

1 Introduction

This paper examines a number of previously proposed methods for the parallel integration of differential equations from the perspective of computation graphs. The inherent structure of the computation graph is imposed by the differential equation, but it may not permit adequate parallelism. Many methods can be viewed as modifications of the graph to introduce parallelism at the expense of additional computation and this viewpoint allows us to consider alternate approaches. The various approaches to parallelism can be classified as method parallelism, parallelism across space, or parallelism across time. Method parallelism is suitable for low degree parallelism only. This paper, which is based largely on two earlier papers[5, 6] concentrates on parallelism across space, both by direct and waveform methods. A companion paper considers parallelism across time.

We are concerned with the integration of the initial value ODE

$$y' = f(y), \qquad y(0) = y_0 \tag{1}$$

where y and f are ν-dimensional vectors. Frequently the components of y have been derived by semi-discretization of a PDE in space so that the κ-th component of y, y_κ, is an approximation to $y(x_\kappa)$. Even if the system did not arise via semi-discretization, it is often the description of a "lumped circuit" model such as an electronic or structural network in which a discrete approximation was made to a continuous system before a differential equation model was developed. (In this view, the universe is a large PDE with a lot of internal boundaries!) For this reason, we will refer to the direction of the independent variable as "time" and the "direction" of the components of the system as "space."

The flow of information in the solution of an IVP for an ODE is shown in Figure 1. The circles represent the "computation" necessary to advance the solution from one approximation, represented by a small black disk, to the next. In a one-step method, this computation starts with nothing more than the prior solution value and, by means of arithmetic operations, represented by $+*$, and function evaluations, represented by

f, computes an approximation to the solution at the next mesh point. In a multistep method, the values from several prior mesh points are used to compute the next value. This could be represented in the graph by adding additional edges in the obvious way or we could view the disk nodes as representing the collection of saved information at each mesh point.

Figure 1 displays no parallelism: each computation, represented by a circle, must be completed before the next can start. However, the circles can themselves be expanded as computation graphs. One approach to parallelism is to explore methods which maximize the parallelism in the graph representing the single step computation. This can be referred to as *method parallelism* and is typified by parallel Runge-Kutta methods[17, 8], although block methods[16] can also be viewed in this class. Method parallelism, while potentially important for a low degree of parallelism computer, does not scale with the problem size (which increases with the number of equations and the number of mesh points in the time dimension).

Unfortunately, Figure 1 does not display parallelism that might be possible due to the structure of the differential system. Figure 2 shows three of a system of ODEs, arising from semi-discretization of a PDE, representing approximations to the values of the dependent variable at different, one-dimensional spatial positions.

This figure indicates communication between the computations to be carried out at each time level. Depending on the "strength" of this communication, it may be possibly to handle it explicitly using prior values of other variables, or we may have to do it implicitly.

Execution of a graph consists of computing the output value of a computation node when all of the input values to that node have been computed. When there are cycles in the graph, computation is not possible until the graph has been modified to break the cycles. The computation nodes shown in the Figure 2 are compound ones and it may be possible to break cycles when the nodes are expanded. Thus, when the computation in Figure 2 is handled by a predictor-corrector method, the cycles are broken. (In fact, any actual computation has to be explicit by the very nature of computers, although the result of an iterative procedure is sometimes best viewed as the solution of an implicit system.)

Parallel execution of a computation graph consists of assigning each node in the graph to a processor. All processors can operatė in parallel only if every processor has nodes which are ready to be evaluated at all times. Clearly, our objective is to structure the computation graph to have a large *width* so that there is a large degree of parallelism.

By *parallelism across space* we refer to the allocation of separate processors to each of the horizontal lines in Figure 2. By *parallelism across time* we refer to the allocation of separate processors to each of several time points, so that, for example, one processor would be computing values for $y_{n-1,\kappa}$ for all κ, another would be doing the same $y_{n,\kappa}$ for all κ, and so on. When parallelism across space is applied to Figure 2 and an explicit method is used, it is clear that all processors can execute simultaneously provided that the processing time for each node is approximately the same. In that case, the only considerations are those of the communication between processors, indicated by the diagonal lines, and synchronization. However, when parallelism across space is applied to Figure 2, the each processor must wait for another to finish and there is an apparent deadlock until the implicitness has been resolved. If parallelism across time is applied to these figures, each processor must wait until its predecessor has computed its left neighbor and there appears to be no parallelism present.

Because the numerical solution of ODEs has little natural parallelism unless explicit spatial communication is possible, it is necessary to restructure the computation graph to introduce parallelism.

Let us first look at the issue of implicit systems. A system of three implicitly coupled equations is shown in Figure 3. If the equations have a simple structure, for example, if they are linear, the computation can be re-arranged to get explicit solution, as in the direct solution of linear equations. However, for general nonlinear equations that is not possible and we have to use iterative methods to get the graph shown in Figure 4. Initial values, P_0, etc., are selected, and then new values are computed in terms of the prior values. We have not indicated what the computation in the new nodes is, so this is a model for any of a number of processes. If we have equations like $q = f(p, q, r)$ then the simple functional iteration $q_{n+1} = f(p_n, q_n, r_n)$ could be used. Alternatively, $q_{n+1} = f(p_n, q_{n+1}, r_n)$ could be solved for q_{n+1} at each processing step. This requires more computation, but is likely to converge more rapidly. Yet another alternative is to linearize with respect to the new value of the variable being computed and to solve $q_{n+1} = f((p_n, q_n, r_n) + f_q[q_{n+1} - q_n]$ directly at each iteration for q. All approaches yield Jacobi-style iterations. Clearly Figure 4 can be executed in parallel on three processors.

If a Gauss-Seidel-type iteration is used, we get Figure 5 in the case of a system of two equations. Note that it has no apparent parallelism unless there is method parallelism. However, if the "variables" P and Q are collections of values, for example vectors whose components are approximations to the solution of a differential equation at a set of mesh points, if the computation represented by the nodes in Figure 5 computes the components of those vectors serially (e.g., in the increasing time direction for ODEs), and if the *consumer nodes* – the nodes that use the information to compute their own values – also use the information serially, there is some hidden parallelism because one node can be producing elements of the vector one or more steps in advance of their use by the consumer nodes. This is called *wavefront processing*. The modification of predictor-corrector methods suggested by Miranker and Liniger[13] is in this class, and the approach may prove to be important for waveform methods.

The computation graph for the standard predictor-corrector method has the form shown in Figure 6 (this shows a $PE(CE)^2$ method) where $f + *$ represents the predictor calculation and $*f$ represents the corrector. The use of the final corrector as the starting value for the next predictor and subsequent correctors makes this an essentially serial process. Miranker and Liniger suggested using the previous predictor to compute the next predictor, and so on for the correctors as shown in Figure 7. This permits the computation of the predictor at t_{n+1}, the first corrector at t_n, and the second corrector at t_{n-1} to be handled simultaneously since the vectors of predictor and corrector values can be computed serially from left to right.

The amount of parallelism that can be obtained by the predictor-corrector wavefront method or by block methods is small. Direct methods for differential equations do not have much opportunity for massive parallelism unless explicit methods can be used for large systems of equations as typically arise in the semi-discretization across space of hyperbolic PDEs. In all other cases we have to look to modifications of existing methods to introduce parallelism.

For parallelism across space, it is useful to classify systems of ODEs into two types: *homogeneous*, namely those in which all of the ODE subsystems are similar as usually happens in the semi-discretization of PDEs, and *heterogeneous*, as usually happens when the equations are derived from a "lumped circuit" model such as VLSI systems. Since the processing time per step for homogeneous systems will be essentially the same for each subsystem, it is attractive to think of integrating each subsystem on a separate processor *synchronously*. If we have explicit communication, this is almost certainly the best approach unless the time of communication for information from a single time step is large relative to the computational time. If the system must be implicitly integrated, we must find ways to parallelize. If the system is heterogeneous, it may be better to consider waveform methods across space.

2 Explicit Methods for Parabolic Equations

Parallelism across space is a natural form of parallelism, especially on SIMD machines or hypercubes, for time-dependent PDEs. It was used in codes for the Illiac IV, one of the first large-scale parallel computers. It is good for large-scale parallelism when there are many equations (or spatial mesh points); its major limitation seems to be communication.

We consider the underlying physical problem to understand the amount of communication needed. In hyperbolic problems, communication is limited to nearby neighbors—no information can travel faster than the local speed of sound (along characteristics). Consequently, as problem size increases, communication does not increase relative to computation. Although communication may be expensive relative to computation (as, for example, in most hypercube architectures), its volume relative to computation can be reduced by handling a large number of equations in each processor[7].

On the other hand, in parabolic problems some information travels arbitrarily far in any time step, no matter how small. This suggests that there may have to be full interprocessor communication.

If this amount of communication is really required, it seems to impose a serious limit on the amount of parallelism possible, since, as the number of equations, N, grows, the apparent communication increases like N^2. However, the communication needed may not grow that rapidly. If we consider the heat equation in one dimension, we can show that, for any given accuracy ϵ, communication is needed only from mesh points $O(\epsilon^{1/2p-1/q}(-\log\epsilon)^{1/2})$ distant asymptotically, where p and q are the orders of the time and spatial discretizations, respectively. This leads us to consider methods which use explicit communication from that number of points. If $p = q$ and we choose step sizes based on accuracy (rather than stability) we find that as the accuracy, ϵ, is reduced, we can allocate the $O(\epsilon^{-1/p})$ spatial mesh points to $O(\epsilon^{-1/2p})$ processors and use nearest neighbor communication only, and yet achieve stability. This is similar to a scheme is discussed by Kuznetsov[12].

3 Waveform Methods

Heterogeneous problems are not amenable to solution by the type of technique discussed in the previous section for several reasons:

1. each equation or subsystem of equations may take different processing time per time step (load imbalance),

2. some systems may be able to use larger time steps than others (more load imbalance),

3. the communication between subsystems is not regular so also causes an imbalance.

There seem to be two ways around these problems. One is to go to waveform iteration, the other is to allocate tasks to processors from a pool of integration processes awaiting execution as the processors complete prior tasks. In waveform iteration, a processor obtains the current values of the waveforms from other processors prior to the start of the next integration and then integrates its subsystem of equations over the integration interval independently of the progress of other processors. Thus, step sizes appropriate to the subsystem can be used.

The difficulty with waveform methods is that, considered as an iterations in function space on differential equations, they are superlinearly convergent so there is no measure such as rate of convergence. While a numerical solution using implicit, discrete numerical integration methods will be linearly convergent so that a rate of convergence can be defined, that rate of convergence is not meaningful since the numerical approximation will exhibit features similar to a superlinearly convergent scheme for the small numbers of iterations of interest, that is, the amplification matrix from step to step will be strongly asymmetric so that the lower-triangular terms in its powers will dominate for a long time.

There does not seem to be a good solution to the measurement of convergence yet. Nevanlinna[14] has analyzed convergence for linear problems. Another approach, given by Juang in her thesis[10, 9, 11, 6], is to look at the rate of increase of the order of accuracy of the iterates. A waveform iterate is defined by a differential equation, so each iterate is a function of t. The number of terms in the power series expansion of the iterate that match the terms in the true solution can be computed, and the *speed of convergence* is defined to be the average number of additional terms correct in each iteration. It was shown that this is at least one, and, under some conditions can be much higher. There are three important cases, waveform Jacobi (WJ), waveform Gauss-Seidel (WGS), and waveform Newton (WN). Juang has shown that

WJ the accuracy increase is exactly one for general problems,

WGS the accuracy increase exceeds one,

WN the accuracy increases by a factor of two on each iteration.

The accuracy increase in WGS is dependent on the numbering of the equations. The average accuracy increase can be computed by examining the directed dependency graph for the differential equations. This graph contains a vertex corresponding to each subsystem and an edge from vertex A to vertex B is a variable from subsystem A appears in the equation for a variable in subsystem B. If a vertex has no edges entering it, the first integration computes its solution exactly because it depends on no other values. If a directed dependency graph has no cycles, the average accuracy increase is infinite because, after a number of iterations equal to the longest path in the graph, all subsystems have been integrated exactly. However, if there are cycles, errors from one subsystem will be propogated to the next, although the accuracy will increase by one order at each integration.

4 Implementation of Parallel Waveform Methods

While waveform methods offer many potential benefits for parallelism across space, extracting those benefits complicates the code considerably. The problems arise because of the need to communicate the waveforms between processors. If a shared memory architecture is used, all processors can refer to a common data area, whereas in a distributed memory architecture, the waveforms will have to be sent from one processor to another. If wavefront processing is desired, waveforms must be updated "on the fly" as new information is generated so that other processors have access to the most recent results. This is probably not practical on a distributed processor like a hypercube because of the relatively high cost of short messages. If wavefront methods are not used, the results only have to be transmitted after each integration of a subsystem.

The organizational problems that must be addressed in the implementation include:

1. how to schedule tasks if there are more subsystems than processors.
2. how to represent a waveform so that it can be updated easily,
3. how to keep the overhead of synchronization to a minimum,
4. how to determine when the iteration is done.

If we assign each subsystem to a processor, issues of synchronization are somewhat simpler since each processor may as well integrate its subsystem as rapidly as possible using *chaotic* iteration[3]. At the completion of each integration, the solution can be placed in common memory or sent to all processors that need it. However, if some subsystems have relatively little activity, this may not be a very efficient method since many processors will be re-integrating systems that are unchanged from the prior iterate. An alternative is to schedule integrations from a pool of integration tasks that are "ready" to be executed, where "ready" means that there is some reason to believe that their next iterate will be different from the previous one. In an experimental implementation[1, 2] ready meant that one or more of the *inputs* to a subsystem had changed since that subsystem was last integrated, where an input was a variable appearing in the differential equations defining a subsystem.

In this implementation the following strategy was adopted:

- When a subsystem became ready for integration, it was *posted* to a list of subsystems that could be integrated together with an initial integration time, called a start time. If the same subsystem was posted more than once between the times it was integrated, the posting with the earliest start time was retained.
- Initially, the values of all waveforms were set to constants equal to the initial values at t_0.
- All subsystems were posted for integration starting from t_0.
- The subsystem with the earliest posted starting time was scheduled when any processor became available. (Initially, the effect was to schedule subsystems in an arbitrary order until all processors were in operation.) When a integration was scheduled, it occupied a processor until the integration of the current window was complete and all data updates required as a result had been completed. Also, when it was scheduled, its posting was removed from the list prior to the start of the integration so that any further changes in its inputs would cause a re-integration.

- As an integration of a subsystem proceeded, the integrator checked to see if the new iterate differed from the previous iterate by more than a given tolerance. Until that happened, the integrator was said to be in the *validate phase*. The last integration step in the validate phase was saved as the *change* time. The integrator then entered the *integrate phase* and integrated over a window of some length.
- When the integration of the subsystem window was complete, the change time was posted as the start time for all *consumers* of the subsystem (a consumer is another subsystem that uses any of the variables of this *producer* subsystem).
- The subsystem also posted the end of the just-finished integration as its own start time unless this was equal to the end of the interval of integration for the problem.

The integration was complete when no integrations were in progress and no start times were still posted.

No heuristic was found for selecting window length that had a good theoretical basis. The program terminated a window integration when the number of integration steps in the current window reach a predetermined level (determine by storage considerations) or when the change from the previous iterate exceeded some multiple of the integration tolerance. (The rationale for the latter was that the each iterate tended to grow rapidly in time very far from the region in which it was accurate and there seemed to be little point in computing into the region where it was blowing up. Since the iterate integration was often stopped before the end of the total interval of integration, the solution was assumed to be constant beyond that point for the computation of other iterates.

The experimental implementation was on a shared-memory processor so the data was shared. There were two major problems to deal with. The first was to develop a structure that allows easy updating of waveforms recognizing that each new iterate may use a different step size sequence than the previous one. If fixed step sizes were used it would be easy to store the information in an array where each cell corresponded to the value at a particular time. For variable steps it was necessary to use a doubly-chained list with each entry containing the time and solution information for that time. Because many processors had access to the same set of data, the old segment of the list could not be deleted when a new set of values had been generated. Instead, the new part of the list was attached to the forward pointer of the old list starting at the appropriate time point. This did not require locking provided that the new part of the list had valid pointers before it was hooked in. The old part of the list that had been replaced by the new part might still be in use by another processor, so it was left in memory with its reverse pointer still connected back to the most recent list. The old part was not removed until it was known that no processor could still be using it. This was accomplished by keeping track of each old segment and the consumer processors that could still be using it. Each time that a consumer accessed the start of the data for the subsystem the fact was noted for all old segments not yet returned to the free memory list.

The waveform was represented by function values and derivatives at each mesh point so that cubic interpolation could be used by a consumer. This meant that there was little point in integrating to higher than third order.

The only locking needed in the implementation proposed was in setting the change times of consumer subsystems and in adjusting the use data associated with old list segments.

References

[1] S. Aslam, Asynchronous Waveform Relaxation Methods for Ordinary Differential Equations on Multiprocessors, PhD Thesis, Department of Computer Science, Univ of Illinois, 1990.

[2] S. Aslam and C. W. Gear, Asynchronous Iterations of Ordinary Differential Equations on Multiprocessors, Report #1525, Dept of Computer Science, Univ. of Illinois, Jul 1989.

[3] G. M. Baudet, Asynchronous Iterative Methods for Multiprocessors, *JACM* **25**, #2, pp 226-244, Apr 1978.

[4] Duff, I. S. et al, Some Remarks on Inverses of Sparse Matrices, AERE Harwell Report CSS 17, (1985).

[5] C. W. Gear, Parallel Solution of ODEs, in *Real-Time Integration methods for Mechanical System Simulation*, ed E. J. Haug and R. C. Deyo, NATO ASI Series F, Vol 69, Springer-Verlag, Berlin, pp 233-248, 1989.

[6] C. W. Gear and Fen-Lien Juang, The Speed of Waveform Methods for ODEs, in *Applied and Industrial Mathematics*, ed R. Spigler, pp37-48, Kluwer Academic Pub., Netherlands, 1991.

[7] J. L. Gustavson, G. R. Montry, and R. E. Benner, Development of Parallel Methods for a 1024-processor Hypercube, *SIAM J. Sci. and Stat. Comp.*, **9**, #4, pp 609-638, Jul 1988.

[8] K. R. Jackson and S. P. Norsett, The Potential for Parallelism in Runge-Kutta Methods, Part 1: RK Formulas in Standard Form, Tech Report # 1239/90, University of Toronto, Dept of Computer Science, Nov 1990.

[9] Juang, F., Accuracy Increase in Waveform Relaxation, Dept of Computer Science Report #1466, University of Illinois at Urbana Champaign, 1988

[10] Juang, F. L., "Waveform methods for ordinary differential equations," PhD Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, November, 1989.

[11] Juang, F., Gear, C. W., Accuracy Increase in Waveform Gauss Seidel, Dept of Computer Science Report #1518, University of Illinois at Urbana Champaign, June, 1989.

[12] Kuznetsov, Y. A., New Algorithms for Approximate Realization of Implicit Difference Schemes, *Sov. J. Numer. Anal. Math. Modeling*, **3**, #2, pp 99-114, 1988.

[13] W. L. Miranker and W. Liniger, Parallel Methods for the Numerical Integration of Ordinary Differential Equations, *Math. Comp.* **21**, pp 303-320, 1967.

[14] Nevanlinna, O., "Remarks on Picard-Lindelöf iteration", REPORT-MAT-A254, Helsinki University of Technology, Institute of Mathematics, Finland, December 1987.

[15] Shampine, L., Gear, C. W., A User's View of Solving Stiff Ordinary Differential Equations, *SIAM Review*, **21**, #1, pp 1-17, Jan, 1979.

[16] L. F. Shampine and H. A. Watts, Block Implicit One-Step Methods, *J. Math. Comp.* **23**, pp 731-740, Oct. 1969.

[17] P. J. van de Houwen and B. P. Sommeijer, Iterated Runge-Kutta Methods on Parallel Computers, *SIAM J. Sci. Stat. Comput.* **12** #5, pp 1000-1028, Sept, 1991.

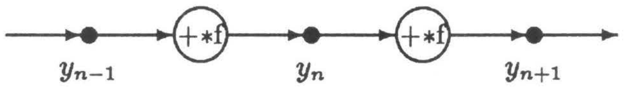

Figure 1: Information flow in ODE

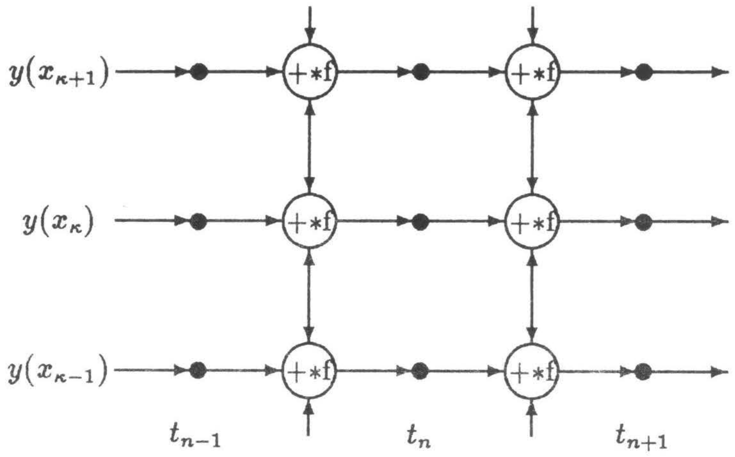

Figure 2: System of equations

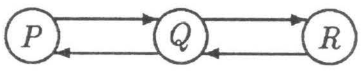

Figure 3: Three coupled equations

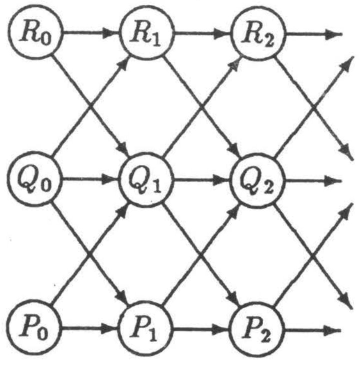

Figure 4: Iterative solution of three equations

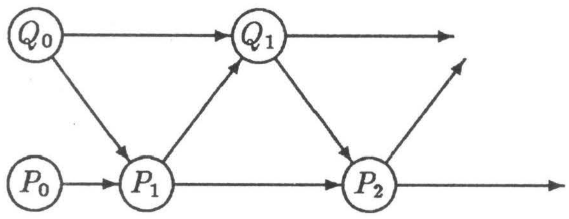

Figure 5: Gauss-Seidel iteration for two equations

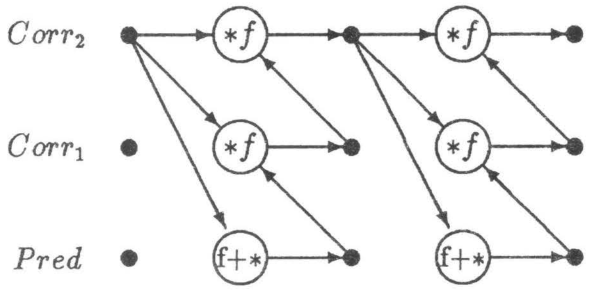

Figure 6: Standard predictor-corrector process

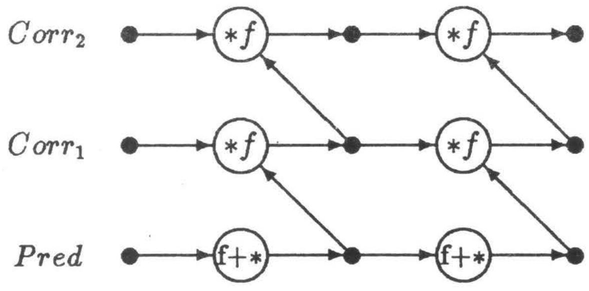

Figure 7: Predictor-corrector modified for parallelism

Parallelism across time in ODEs[*] Extended Abstract

C. W. Gear[†]& Xu Xuhai[‡]

August 12, 1991

1 Introduction

There is no natural parallelism across time in ODEs, so to exploit massive parallelism for a small system of equations it is necessary to use iterative techniques in time, such as waveform. The Picard method, for example, allows each integration to be performed in $O(\log N)$ time over N time steps but its convergence is poor for any but almost quadrature problems ($\partial f/\partial y$ small). Generalized Picard, or waveform, may have much faster convergence but less parallelism. This paper considers parallelism across time, explores a proposal made in an earlier paper[1], and reports on some tests made on that method.

Information in initial value problems flows naturally in increasing time. Until we have an approximate value for $y(t_n)$ we cannot compute $f(y(t_n))$ to get an approximation to $y'(t_n)$, and the only way we can get the first good approximation to $y(t_n)$ is by extrapolation from nearby, earlier points. Hence, the problem seems inherently serial. However, there are some special cases in which other options are possible.

A method that is massively parallel across time for ODEs must simultaneously evaluate the right hand side of the differential equation $y' = f(y,t)$ at many different time points t_n. These evaluations require approximations of the solution, $y(t_n)$, to compute a better approximation, so such a method is essentially an iterative method, even if very few iterations are used. We will name the successive iterates $y_n^{(m)}$ for $m = 0, 1, \cdots$. For each value of m the approximation can be thought of as a discrete approximation to a "waveform," so such a method is really a waveform method [7]. However, we usually think of waveform methods as iterations on a space of differentiable functions (that are later discretized in the numerical solution process) whereas our viewpoint here is of a series of discrete approximations. The number, N, of discrete points, t_n, $n = M+1, \cdots, M+N$, handled simultaneously is the parallelism of the method. (In this sense, a conventional $P(EC)^q$ method is a method with parallelism one and q iterations.

[*]Work done while at University of Illinois, Department of Computer Science. Supported in part by US DOE under grant DOE DEFG02-87ER25026 and by NSF under grant DMS 87-03226.

[†]NEC Research Institute, 4, Independence Way, Princeton, NJ 08540

[‡]Mathematics Department, Wuhan University, PRC.

We are concerned with

$$y' = f(y,t); \qquad y(0) = y_0.$$

First, consider the quadrature case in which $f = f(t)$ is independent of y. In that case, we can evaluate all $f(t_n)$ in parallel and then compute

$$y(t_n) = y(t_0) + \sum_{i=0}^{n} \omega_i f(t_i)$$

where the ω_i are suitable quadrature coefficients. This can be computed for all $n = 1, \cdots, N$ in $O(\log(N))$ time on N processors. We cannot add N terms in less than $O(\log(N))$ time, so this is lower limit on speed of parallelism across time for ODEs. The major question is can we even come close to $O(\log(N))$ for ODEs?

In the case of linear problems the answer is "yes." If $f(y,t)$ is linear in y, then knowledge of $f(\hat{y}, t_n)$ for some $\hat{y}$ and of $\partial f / \partial y$ provides full knowledge of the system at t_n. The details are straightforward and appear in the full paper.

What about nonlinear ODEs? The obvious approach is to linearize and use iteration. Iteration is the basis of the *waveform methods* which can be written as

$$y^{(m+1)} = f(y^{(m)}) + G(y^{(m+1)}, y^{(m)}) \tag{1}$$

where G is any suitable smooth function such that $G(z,z) = 0$ chosen to make the integration of (1) for the next iterate $y^{(m+1)}$ inexpensive. In the waveform Newton method we set

$$G = f_y(y^{(m)})(y^{(m+1)} - y^{(m)}) \tag{2}$$

to reflect the linear behavior of f about the current solution $y^{(m)}$). Since (1) plus (2) gives a linear ODE, the method described above can be used to integrate it. Hence the cost is $O(Q\log(N))$, where Q is the number of iterations. However, the number of iterations depends on N (except for linear problems). If N is increased to lengthen the interval of integration and the amount of parallelism, convergence will be slowed because, under exact integration, each iteration increases the order of the power series approximation being developed by the method (see Juan[5] and Gear and Juang[3]). If N is increased to reduce stepsize over a constant interval so as to increase integration accuracy, more iterations will be needed to increase iteration accuracy correspondingly.

The nature of the dependence of Q on N cannot be determined easily. If we assume, for the constant interval case, that the iteration is linearly convergent, we can make the following argument (in which equality is to be interpreted as "order of"): suppose we want an accuracy of ϵ and want equal errors from truncation and iteration. If a p^{th} order method is used, we have $h^p = \epsilon$. Since $N = 1/h$ we have $N^p = 1/\epsilon$. If the convergence rate is $r < 1$, then to obtain an iteration error ϵ we must iterate until $r^Q = \epsilon$ or

$$Q = O(\log(\epsilon)/\log(r)) = O(\log(N))$$

if r is independent of h, and hence of ϵ. This would yield an overall speed of $O(\log^2(N))$, but the assumptions are questionable.

Alternatively, consider N being increased to increase the interval of integration and parallelism. Here we need a different assumption on convergence. Suppose the iteration error after Q iterations is $O(N^Q/Q!)$ based on the nature of the convergence of waveform which introduces at least one more correct term in the power series at each iterate.. This leads to a Q which is $O(N)$. Again, the assumptions are questionable, but this estimate suggests that the iteration will not be valuable in this situation. (The waveform methods are popular for classes of problems, particularly nonlinear electrical network models, but the reasons for their success seem to be due to considerations to do with ease of implementation for highly adaptive methods, effective use of paging from secondary memory, and their ability to reap benefits from the special structure of such problems that leads to the one way flow of much information.)

2 The Blended Waveform Method

This leaves open the question whether or not there are other methods that can come close to $O(\log(N))$ speed in parallelism across time. We have been experimenting with a new method that might be useful for some classes of problems. It is based on the following observation:

There are two types of problems for which parallelism is easy: quadrature and nonlinear equations. For the former we have already seen that integration can be performed in $O(\log(N))$ time with N processors. The latter, namely solving the problem $f(y,t) = 0$ for N different values of t, can be done in $O(1)$ time with N processors. These two problems are, in a sense, limiting cases of an ODE when the stiffness changes from zero to infinity. Consider the class of problems $y' = \lambda f(y,t) + g(t)$ for variable λ. Assume that f is "well behaved" in the sense that its Jacobian has a reasonable condition number and norm. As λ tends to zero, the stiffness of the problem goes to zero and we get the infinitely nonstiff problem $y' = g(t)$ which is the quadrature problem. As all eigenvalues of λ become infinite, we tend to the infinitely stiff problem, $f(y,t) = 0$.

An ODE lies somewhere between these extremes, so below we propose a method that uses a blended combination of methods that would be good for the two extremes in the spirit of the blended methods of Skeel and Kong[9] although the methods proposed are very different in significant ways. As in Skeel and Kong, the blend is determined by the Jacobian $\partial f/\partial y$ of the system, and the underlying methods proposed are the BDF methods for stiff problems and the Adams methods for nonstiff problems. However, they are applied in ways that the majority of the computations can be performed in parallel, even on SIMD/vector architectures. Below we discuss the basic method as described in [1] and two variants. In a subsequent section we report on some numerical tests.

2.1 The CBA(I) Method

This is the method described in [1]. The stiff components of a problem can be computed using a backward differentiation formula (BDF). In the conventional use of BDF-based methods, the formula is applied sequentially to the time steps making the algorithm serial. For infinitely stiff problems, the prior values do not affect the current value since the problem reduces to the solution of $f(y,t) = 0$. Hence, even if we use past values from a previous iteration, we will get the correct answer for the infinitely stiff components in one iteration.

Suppose we have a discrete approximation $y_n^{(m-1)}$, $n = k, k+1, \cdots, N$ and some starting values y_j, $j = 0, \cdots, k-1$. (Usually, $k = 1$ since a variable-order, variable-step method is used and the order starts at $k = 1$.) For the initial approximation, we usually choose $y_n^{(0)} = y_{k-1}$ for $n \geq k$. We could compute a BDF approximation, $z_n^{(m)}$, using

$$z_n^{(m)} = \Sigma_B y_n^{(m-1)} + h\beta_0^B f(z_n^{(m)}) \tag{3}$$

where

$$\Sigma_B w_n = \sum_{i=1}^{k} \alpha_i^B w_{n-i},$$

and β_0^B and the $\{\alpha_i^B\}$ are the BDF coefficients of order k. Since this in general is a nonlinear equation, it can be approximated by

$$z_n^{(m)} = \Sigma_B y_n^{(m-1)} + h\beta_0^B[f(y_n^{(m-1)}) + J_1(z_n^{(m)} - y_n^{(m-1)})] \tag{4}$$

where J_1 is an approximation to the Jacobian chosen in part for its convenience in solving the linear equations. (We could solve (3) by iterating a form of (4) to convergence.) Note that all z values can be computed in parallel because they are mutually independent. (Since the calculations for all $z_n^{(m)}$ are identical, they could be done on an SIMD or vector architecture.)

Equation (4) should be chosen when the ODE system is not small and it is desirable to partition the equations across different processors. In that case, J_1 would be chosen as an approximation to the Jacobian of f that is zero in blocks representing information that is expensive to communicate. J_1 should also be chosen so that $[I - h\beta_0 J_1]x = b$ is easy to solve.

The extremely stiff parts are computed very accurately by either (3) or (4). At the same time, the derivatives of the extremely nonstiff components are also computed extremely accurately when $f(z_n^{(m)})$ is evaluated because, in that case, f does not depend on the value of z to any great extent.

The nonstiff components are best calculated by a quadrature formula. Standard nonstiff integration formulas such as Adams formulas are good for quadrature, so it seems reasonable to compute an Adams approximation, $y_n^{A,(m)}$, using

$$y_n^{A,(m)} = y_{n-1}^{(m)} + h\Sigma_A \tilde{f}(z_n^{(m)}) \tag{5}$$

where

$$\Sigma_A w_n = \sum_{i=0}^{k} \beta_i^A w_{n-i},$$

the β_i^A are the Adams-Moulton coefficients, and

$$\tilde{f}(z_n^{(m)}) = f(y_n^{(m-1)}) + J_1(z_n^{(m)} - y_n^{(m-1)}),$$

i.e., $\tilde{f}$ is the approximation used for the derivative in (4). (We should use $\tilde{f} = f$ if (3) is used.) We would like to combine the two solutions $z_n^{(m)}$ and $y_n^{A,(m)}$ so that we select the stiff components from $z_n^{(m)}$ and the non stiff components from $y_n^{A,(m)}$. To do this we combine them using

$$\begin{aligned}
[I - h\gamma J_2]y_n^{(m)} &= y_n^{A,(m)} - h\gamma J_2 z_n^{(m)} \\
&= y_{n-1}^{(m)} + h\sum_A \tilde{f}(z_n^{(m)}) - h\gamma J_2 z_n^{(m)}
\end{aligned} \tag{6}$$

This uses a blended combination of the two solutions based on the size of the matrix J_2. In practice, J_2 is an approximation to the Jacobian, chosen for its sparsity and the ease of solving linear equations involving J_2. It may or may not be the same approximation as the J_1 used in (4). The coefficient γ will be chosen to obtain desirable stability properties for $y_n = \lim_{m\to\infty} y_n^{(m)}$ and rate of convergence as $m \to \infty$.

Eq (6) has the form

$$y_n^{(m)} = S_n y_{n-1}^{(m)} + G(y_{\{n-i\}}^{(m-1)}) \tag{7}$$

where the matrix

$$S_n = [I - h\gamma J_2]^{-1} \tag{8}$$

has the dimension of the system of ODEs. Although equation (7) is a linear recurrence relation, N steps of it can be solved simultaneously in $\log N$ time, as discussed earlier, because all coefficients and the inhomogeneous term can be evaluated in parallel.

A number of questions have to be asked about this proposed method. These include

- Does the iteration converge?
- To what ODE method does the iteration converge?
- Is that method zero stable and consistent?
- What are its large-h stability properties?

The brief answers, elaborated in the full paper, are that the iteration converges provided only that J_1 is close enough to the system Jacobian, and the limit of the iteration is a stable and consistent method for some choices of the underlying methods, with reasonable properties such as stiff stability.

2.2 Method CBA(II)

A major drawback of CBA(I) is that (7) has the dimension of the ODE system, and the matrix S_n incurs a number of operations proportional to the cube of the system dimension. Thus, it is only of value for small systems unless the matrix S_n has a suitable sparse structure such as block diagonal. S_n has the structure of $[I - J_2]^{-1}$ so J_2 would have to be block diagonal. This motivated the CBA(II) method which is obtained from method CBA(I) by modifying (5) to

$$y_n^{A,(m)} = y_{n-1}^{A,(m)} + h\Sigma_A \tilde{f}(z_n^{(m)}). \tag{9}$$

Use of $y_{n-1}^{A,(m)}$ instead of $y_{n-1}^{(m)}$ means that (9) provides a recurrence relation for $y_n^{A,(m)}$ which does not require matrix operations.

2.3 Method ICBA

Method CBA(II) (and to a lesser extent, CBA(I)) got into difficulty because some of the iterates for z_n and y_n were far from a reasonable value for an argument of f. To avoid this problem, the computed values can be forced to be solutions of equations involving f directly.

For z this can be handled by using (3) instead of (4).

We note that J_2 is supposed to be an approximation to f_y so that $J_2 y_n^{(m)}$ can be approximated by $f(y_n^{(m)}) - f(w) + J_2 w$ for a w near $y_n^{(m)}$. This allows us to rewrite (6) as

$$y_n^{(m)} - h\gamma f(y_n^{(m)}) = y_n^{A(m)} - h\gamma[J_2(z_n^{(m)} - w) + f(w)]. \tag{10}$$

Clearly, $w = z_n^{(m)}$ is a good choice because of the cancellation. Eq (10) can then be solved by iterating

$$[I - h\gamma J_2]y_n^{(m,j+1)} = y_n^{A(m)} - h\gamma[f(z_n^{(m)}) - f(y_n^{(m,j)}) + J_2 y_n^{(m,j)}] \tag{11}$$

We call this the iterated CBA method, or ICBA.

Note that method ICBA could be used with CBA(I) or CBA(II) formulations. However, the parallelism is lost if CBA(I) formulation is used because, in that case, (5) can not be used to find $y_n^{A(m)}$ until $y_{n-1}^{(m)}$ has been computed using the iteration (11). Hence, we consider only the extension of CBA(II) to ICBA.

The absolute stability properties of ICBA are exactly the same as those of the underlying CBA(II) method, since these regions are computed under the assumption that $J_1 = J_2 = f_y = \lambda I$.

3 Conclusion

A series of numerical tests were performed. Constant step sizes were used to remove the influence of an automatic step control mechanism. The number of iterations needed to achieve "convergence" was recorded, where convergence was defined as a difference of successive iterates being less than a specified tolerance ϵ.

To reduce machine time, the iteration was performed over a floating "window" of time points, where the start of the window was the first point at which the iteration had not converged, and the end was the last point for which the difference of the last two iterates had not exceeded $E\epsilon$. (E was normally set to 10^3. Increasing it increases the number of iterations, but also increases the parallelism.)

Tests were performed on five examples. The results are reported in the full paper.

The data obtained does not lead us to be optimistic about the potential of the proposed methods except under very special circumstances. If the dimension of the problem is small, Method CBA(I), which will then require of the same order of work per iteration as Method CBA(II), is competitive with it on linear problems because of the reduced number of iterations. The large number of iterations needed indicate that the methods will be very inefficient, so only if speed is of paramount importance will it be worth while to consider the use of parallelism of this sort.

References

[1] C. W. Gear, Parallel solution of ODEs, in *Real-Time Integration methods for Mechanical System Simulation*, ed E. J. Haug and R. C. Deyo, NATO ASI Series F, Vol 69, Springer-Verlag, Berlin, pp 233-248, 1989.

[2] C. W. Gear, Massive Parallelism across Space in ODEs, this conference.

[3] C. W. Gear and Fen-Lien Juang, The Speed of Waveform Methods for ODEs, in *Applied and Industrial Mathematics*, ed R. Spigler, pp37-48, Kluwer Academic Pub., Netherlands, 1991.

[4] K. R. Jackson and S. P. Norsett, The Potential for Parallelism in Runge-Kutta Methods, Part 1: RK Formulas in STandard Form, Tech Report # 1239/90, University of Toronto, Dept of Computer Science, Nov 1990.

[5] Juang, F. L., "Waveform methods for ordinary differential equations," PhD Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, November, 1989.

[6] W. L. Miranker and W. Liniger, Parallel Methods for the Numerical Integration of Ordinary Differential Equations, *Math. Comp.* **21**, pp 303-320, 1967.

[7] Learasmee, E., Ruehli, A. E., and Sangiovanni-Vincentelli, A. L., *The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits*, IEEE Trans on CAD of IC and Systems, **1**, #3, pp 131-145, July 1982.

[8] L. F. Shampine and H. A. Watts, Block Implicit One-Step Methods, *J. Math. Comp.* **23**, pp 731-740, Oct. 1969.

[9] Skeel, R. D. and Kong, K-Y., Blended Linear Multistep Methods, ACM Trans. Math. Software, Dec 1977, pp326-345.

[10] P. J. van de Houwen and B. P. Sommeijer, Iterated Runge-Kutta Methods on Parallel Computers, *SIAM J. Sci. Stat. Comput.* **12** #5, pp 1000-1028, Sept, 1991.

The Generalized Schur Decomposition of an Arbitrary Pencil $A - \lambda B$: Theory, Algorithms, Software and Applications *Extended Abstract*

Bo Kågström [*]
Institute of Information Processing
University of Umeå
S-901 87 Umeå Sweden

August 28, 1991

1 Introduction

In this two part talk we will discuss theory, algorithms and software for computing the *generalized Schur decomposition* of an arbitrary matrix pencil $A - \lambda B$ and error bounds for its generalized eigenvalues and eigenspaces, called reducing subspaces. We will also discuss different applications originating from systems and control theory, and singular systems of differential equations. In this extended abstract we give a background to the two part talk and its content. Further, we also give some references to our and related work (for a more complete list of work in this area see for e.g. references in [6, 7]).

Outline of the talk:

1. *Matrix pencils - some algebraic theory*
   Regular and Singular Cases. Kronecker Canonical Form.
2. *Why is the singular case harder than the regular?*
   Reducing subspaces. Generic pencils.
3. *Generalized Schur decomposition*
   GUPTRI form and GUPTRI algorithm.
4. *Perturbation theory for singular $A - \lambda B$*
   Generalization and extention of the regular case.
   Error bounds for reducing subspaces (2 Cases).
   Error bounds for generalized eigenvalues.

[*]This is joint work with JAMES DEMMEL, Computer Science Division and Mathematics Department, University of California, Berkeley, CA 94720

5. *Robustness of the computed GUPTRI form and associated error bounds*

6. *Systems theory applications*
   Controllable subspace. Unobservable subspace. Uncontrollable modes.

7. *Numerical examples*

The generalized Schur decomposition is a natural generalization of the Schur form of a square matrix A, and reducing subspaces are a natural generalization of invariant subspaces. A major contribution is that our error bounds apply to situations where the generalized eigenvalues and reducing subspaces of $A - \lambda B$ are *ill-posed*, i.e. can change discontinuously as a function of A and B. For error bounds to make sense, we need to impose extra conditions on A and B. The extra conditions we choose are motivated by applications in systems and control theory, and are automatically verified by our software.

Just as the Jordan Canonical Form (JCF) $J = R^{-1}AR$ describes the invariant subspaces and eigenvalues of a square matrix A in full detail, there is a Kronecker Canonical Form (KCF) $S - \lambda T = P^{-1}(A - \lambda B)Q$ which describes the generalized eigenvalues and generalized eigenspaces of a pencil $A - \lambda B$ in full detail. Here P and Q are square nonsingular matrices, and $S - \lambda T$ is block diagonal. If $A - \lambda B$ is square and $\det(A - \lambda B)$ is not identically zero, then $A - \lambda B$ is called *regular*, and its KCF $S - \lambda T$ has just Jordan-like blocks on its diagonal. If $\det(A - \lambda B)$ is identically zero, or if $A - \lambda B$ is not square, then it is called *singular*, and other kinds of blocks can appear on the diagonal of $S - \lambda T$.

Of course we cannot guarantee to compute the JCF of A stably, because the matrix R in $J = R^{-1}AR$ may be arbitrarily ill-conditioned. Instead, we compute the Schur canonical form $H = UAU^*$ where U is unitary, U^* is the conjugate transpose of U, and H is upper triangular. U's unitarity means the algorithm (the QR algorithm) can be implemented stably, but H will be much denser than J and so more work must be done to compute A's invariant subspaces from U than from R. Similarly, the P and Q reducing $A - \lambda B$ to its KCF may be arbitrarily ill-conditioned, so we also need to resort to unitary transformations at the price of a denser canonical form. We call this generalized Schur form GUPTRI (Generalized UPper TRIangular) form for short. We discuss all these canonical forms in more detail in Section 2.

If $A - \lambda B$ is m by n, where $m \neq n$, then for almost all A and B it will have the same KCF, depending only on m and n. In particular, it will have only trivial reducing subspaces and no generalized eigenvalues at all. Thus there are no error bounds to compute. Only if $A - \lambda B$ satisfies a special condition (lies in a particular manifold) does it have nontrivial reducing subspaces and generalized eigenvalues. And only if it is perturbed so as to move continuously within that manifold do its reducing subspaces and generalized eigenvalues also move continuously and satisfy interesting error bounds.

Requiring $A - \lambda B$ and its perturbations to lie in a manifold, a set of measure zero, might seem like a very strong requirement. But it is exactly what is required in many control and systems theoretic problems such as computing controllable subspaces and uncontrollable modes; we return to this example later.

Algorithms for computing GUPTRI form (or similar forms) have been proposed by several authors in the past which are numerically stable, i.e. they compute the exact generalized Schur form of a pencil $A' - \lambda B'$ differing only slightly from the input $A - \lambda B$

(see e.g. [1, 2, 3, 12, 13, 14, 9, 10, 11, 16, 18]. The theory justifying the error bounds was published by Demmel and Kågström in [3, 4, 5].

We have written a software package for computing both the GUPTRI form and error bounds for its associated reducing subspaces and generalized eigenvalues. The software is written in Fortran 77, and available from Netlib [8].

2 Kronecker Canonical Form and Reducing Subspaces

The *Kronecker Canonical Form* (KCF) of a pencil $A - \lambda B$ is defined as follows. Suppose $A - \lambda B$ is m by n with complex entries ($A, B \in C^{m \times n}$). Then there exists a nonsingular m by m matrix P and a nonsingular n by n matrix Q such that

$$P^{-1}(A - \lambda B)Q = S - \lambda T \tag{2.1}$$

where $S = \text{diag}\,(S_{11}, \ldots, S_{bb})$ and $T = \text{diag}\,(T_{11}, \ldots, T_{bb})$ are block diagonal. $S_{ii} - \lambda T_{ii}$ is m_i by n_i. We can partition the columns of P and Q into blocks corresponding to the blocks of $S - \lambda T$: $P = [P_1, \ldots, P_b]$ where P_i is m by m_i, and $Q = [Q_1, \ldots, Q_b]$ where Q_i is n by n_i. Each block $S_{ii} - \lambda T_{ii}$ must be of one of the following forms: $J_j(\alpha)$, N_j, L_j or L_j^T. First we consider

$$J_j(\alpha) \equiv \begin{bmatrix} \alpha - \lambda & 1 & & \\ & \cdot & \cdot & \\ & & \cdot & 1 \\ & & & \alpha - \lambda \end{bmatrix} \quad \text{and} \quad N_j \equiv \begin{bmatrix} 1 & -\lambda & & \\ & \cdot & \cdot & \\ & & \cdot & -\lambda \\ & & & 1 \end{bmatrix} \tag{2.2}$$

$J_j(\alpha)$ is simply a j by j Jordan block, and α is called a ***finite generalized eigenvalue.*** N_j is a j by j block corresponding to an *infinite generalized eigenvalue* of multiplicity j. The $J_j(\alpha)$ and N_j blocks together constitute the *regular structure* of the pencil. All the $S_{ii} - \lambda T_{ii}$ are regular blocks if and only if $A - \lambda B$ is a regular pencil. $\sigma(A - \lambda B)$ denotes the eigenvalues of the regular part of $A - \lambda B$ (with multiplicities), and is called the *spectrum* of $A - \lambda B$.

The other two types of diagonal blocks are

$$L_j \equiv \begin{bmatrix} -\lambda & 1 & & \\ & \cdot & \cdot & \\ & & -\lambda & 1 \end{bmatrix} \quad \text{and} \quad L_j^T \equiv \begin{bmatrix} -\lambda & & \\ 1 & \cdot & \\ & \cdot & -\lambda \\ & & 1 \end{bmatrix} \tag{2.3}$$

The j by $j+1$ block L_j is called a ***singular block of right (or column) minimal index*** j. It has a one dimensional right null space, $[1, \lambda, \ldots, \lambda^j]^T$, for any λ. The $j+1$ by j block L_j^T is a ***singular block of left (or row) minimal index*** j, and has a one dimensional left null space for any λ. The left and right singular blocks together constitute the ***singular structure*** of the pencil, and appear in the KCF if and only if the pencil is singular.

Suppose $A - \lambda B$ is regular, so that only $J_j(\alpha)$ and N_j blocks appear. Let σ_i be the spectrum of $S_{ii} - \lambda T_{ii}$. The subspaces $\mathbf{P}_i$ and $\mathbf{Q}_i$ spanned by the columns of P_i and Q_i, respectively, are called the ***left and right deflating subspace*** of $A - \lambda B$ corresponding to σ_i. As shown in [15], a pair of subspaces $\mathbf{P}$, $\mathbf{Q}$ is deflating for $A - \lambda B$ if and only if $\mathbf{P} = A\mathbf{Q} + B\mathbf{Q}$

and they have the same dimensions; in this way they are natural generalizations of invariant subspaces of a square matrix A (in which case $B = I$, $\mathbf{P} = \mathbf{Q}$ and $\mathbf{P} = A\mathbf{P} + \mathbf{P} \supset A\mathbf{P}$).

The situation is somewhat more complicated in the case of singular $A - \lambda B$. With regular pencils, different choices of P_i and Q_i must span the same spaces $\mathbf{P}_i$ and $\mathbf{Q}_i$; this is no longer true with singular pencils. For example, in

$$P(S - \lambda T)Q^{-1} = \begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -\lambda & 1 & 0 \\ 0 & 0 & 1-\lambda \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & x \\ 0 & 0 & 1 \end{bmatrix}^{-1} \tag{2.4}$$

the spaces spanned by Q_2 (the last column of Q) and P_2 (the last column of P) vary depending on x. To overcome this ambiguity, we use *reducing subspaces*, as introduced in [17]. $\mathbf{P}$ and $\mathbf{Q}$ are left and right reducing subspaces of $A - \lambda B$, respectively, if $\mathbf{P} = A\mathbf{Q} + B\mathbf{Q}$ and $\dim\mathbf{P} = \dim\mathbf{Q} - \#(L_j$ blocks in the KCF). In terms of the KCF, given any subset σ_1 of the spectrum of $A - \lambda B$, there is pair of left and right reducing subspaces $\mathbf{P}$ and $\mathbf{Q}$ defined as follows: $\mathbf{Q}$ is spanned by all the Q_i where $S_{ii} - \lambda T_{ii} = L_j$ and those regular $S_{ii} - \lambda T_{ii}$ with spectrum in σ_1. Similarly, $\mathbf{P}$ is spanned by all the P_i for the same $S_{ii} - \lambda T_{ii}$. When σ_1 is empty, the corresponding set of reducing subspaces is called *minimal*, and when σ_1 contains the whole spectrum the reducing subspaces is called *maximal*.

Since there is no limit on the condition numbers of the P and Q reducing $A - \lambda B$ to its KCF, we limit outselves to unitary P and Q to maintain stability. This lets us reduce $A - \lambda B$ to an upper triangular form called GUPTRI form, which is given by

$$P^*(A - \lambda B)Q = \begin{bmatrix} A_r - \lambda B_r & * & * \\ 0 & A_{reg} - \lambda B_{reg} & * \\ 0 & 0 & A_l - \lambda B_l \end{bmatrix} \tag{2.5}$$

where $*$ denotes arbitrary conforming submatrices. Here $A_r - \lambda B_r$ has only L_j blocks in its KCF, indeed the same L_j blocks as $A - \lambda B$. $A_{reg} - \lambda B_{reg}$ is upper triangular and regular, and has the same regular structure as $A - \lambda B$. Finally, $A_l - \lambda B_l$ has only L_j^T blocks in its KCF, the same ones as $A - \lambda B$. In fact,$\cdot A_r - \lambda B_r$ and $A_l - \lambda B_l$ have more complicated block upper triangular forms, often called *staircase forms*, from the sizes of whose blocks the exact singular structure can be read. This block structure and algorithms for computing the GUPTRi form will be discussed in more detail during the talks.

Suppose the eigenvalues on the diagonal of $A_{reg} - \lambda B_{reg}$ are ordered so that the first k, say, are in σ_1 and the remainder are outside σ_1. Let $A_r - \lambda B_r$ be m_r by n_r. Then the left and right reducing subspaces corresponding to σ_1 are spanned by the leading $m_r + k$ columns of P and leading $n_r + k$ columns of Q, respectively.

3 Error Bounds for Reducing Subspaces and Generalized Eigenvalues

The conventional method of computing error bounds is to combine perturbation theory and error analysis. For example, if we were computing an eigenvalue $\lambda(A, B)$ of the square, regular pencil given by A and B, we would derive an inequality of the form:

$$\text{for small enough } \epsilon,\ \|(\delta A, \delta B)\|_E \le \epsilon \text{ implies } |\lambda(A + \delta A, B + \delta B) - \lambda(A, B)| \le \kappa \cdot \epsilon \tag{3.1}$$

Here κ is called a *condition number*. This approach does not work for singular pencils because the eigenvalues (or other quantities we might want to compute) are *ill-posed*. This means that no inequality like (3.1) can hold, because the eigenvalues might be changed completely by arbitrarily small δA and δB. For example, a nonsquare pencil $A - \lambda B$ has no eigenvalues at all for almost all A and B, so almost any perturbation of an $A - \lambda B$ with eigenvalues will make them disappear.

This means we need to impose extra conditions on δA and δB in order to guarantee that $(A + \delta A) - \lambda(B + \delta B)$ continues to have eigenvalues (or whatever feature it is we are interested in) which depend smoothly on δA and δB. The quantities for which we will compute error bounds as well as the extra conditions needed on δA and δB are motivated by applications in systems and control theory: We want error bounds for left and right reducing subspaces, as well as selected eigenvalues of the regular part. For the reducing subspace error bounds, we require that $(A + \delta A) - \lambda(B + \delta B)$ have reducing subspaces of the same dimension as $A - \lambda B$. For the eigenvalue error bounds, we additionally require that $(A + \delta A) - \lambda(B + \delta B)$ have the same number of eigenvalues in its selected regular part as $A - \lambda B$. Both of these conditions are automatically verified by the software. During the talk we will reveiw these generalized eigenspace and eigenvalue error bounds and algorithms for computing them.

We illustrate this with an example from systems theory: computing the *controllable subspace* and *uncontrollable modes* of a linear control system $\dot{x} = Fx + Gu$. Here F is an p by p matrix, G is an p by k matrix, u and x are time dependent vectors, and $\dot{x}$ is the time-derivative of x [5]. The controllable subspace is the subspace of vectors in which x can be "steered" by choosing the input u appropriately; it turns out to be essentially the minimal right reducing subspace of the pencil $[G|F - \lambda I]$. The uncontrollable modes are the eigenvalues of the solutions of $\dot{x} = Fx$ which cannot be controlled by choosing u; they are the eigenvalues of the regular part of $[G|F - \lambda I]$. Almost all control systems are *completely controllable*, which means the controllable subspace is trivial (all of $\mathbf{R}^p$ or $\mathbf{C}^p$) and there are no uncontrollable modes; this corresponds to a KCF of $[G|F - \lambda I]$ with only L_j blocks.

Still, control engineers wish to compute the controllable subspace and uncontrollable modes, despite their somewhat nebulous existence. The reason is that a system is interesting precisely when it is uncontrollable (or close to it) and has a nontrivial controllable subspace and uncontrollable modes. The existence of an uncontrollable system close to one's own, and the location of its uncontrollable modes indicates how stable one's own system will be. An unstable uncontrollable mode (one in the right half plane) means the system will exhibit growing and uncontrolled oscillations. Our software will supply the engineer the following information:

> Within a user supplied distance of the input control system, the software tries to determine whether any uncontrollable systems exist. If one is found, it will be exhibited. This implies that there is a whole surface of uncontrollable systems within the user supplied distance of the input. All of them have an uncontrollable subspace and uncontrollable modes within certain computable intervals supplied by the software.

4 Robust Software

The software package for computing the generalized Schur decomposition with error bounds for its generalized eigenvalues and reducing subspaces has been developed in Fortran 77 and is available from Netlib [8]. In total the package comprises 54 routines but there are only four of them that a normal user will call: **GUPTRI**, **REORDR**, **BOUND** and **EVALBD**. **GUPTRI** computes a generalized Schur decomposition and implements an improved version of algorithm RGQZD [11, 3] for computing reducing subspaces and generalized eigenvalues of a singular pencil $A - \lambda B$. The eigenvalues (diagonal elements) of $A_{reg} - \lambda B_{reg}$ may appear in any order. **REORDR** will reorder the eigenvalues as specified by a user-written function. **BOUND** computes quantities that are used in error bounds for reducing subspaces and selected generalized eigenvalues of $A - \lambda B$. **EVALBD** evaluates the error bounds for reducing subspaces. A standard usage of these routines is as follows:

`CALL GUPTRI (...)`	Compute generalized Schur decomposition.
`CALL REORDR (...)`	Reorder the eigenvalues in specified order.
`CALL BOUND (...)`	Compute error bounds for selected eigen-
`CALL EVALBD (...)`	values and reducing subspaces.

GUPTRI reduces the matrices A and B to generalized Schur form via unitary equivalence transformations such that

$$A(= P^*AQ) = \begin{bmatrix} A_r & * & * & * & * \\ 0 & A_z & * & * & * \\ 0 & 0 & A_f & * & * \\ 0 & 0 & 0 & A_i & * \\ 0 & 0 & 0 & 0 & A_l \end{bmatrix}, B(= P^*BQ) = \begin{bmatrix} B_r & * & * & * & * \\ 0 & B_z & * & * & * \\ 0 & 0 & B_f & * & * \\ 0 & 0 & 0 & B_i & * \\ 0 & 0 & 0 & 0 & B_l \end{bmatrix} \tag{4.1}$$

The diagonal blocks of A and B on exit from **GUPTRI** (4.1) describe the *Kronecker structure* of the input pencil $A - \lambda B$ as follows:

$A_r - \lambda B_r$	has all right singular structure (the right minimal indices)
$A_z - \lambda B_z$	has all Jordan structure for the zero eigenvalue
$A_f - \lambda B_f$	has all finite and nonzero eigenvalues
$A_i - \lambda B_i$	has all Jordan structure for the infinite eigenvalue
$A_l - \lambda B_l$	has all left singular structure (the left minimal indices)

The computed GUPTRI form exposes the Jordan structures of the zero eigenvalue ($A_z - \lambda B_z$) and the infinite eigenvalue ($A_i - \lambda B_i$), respectively. The nonzero and finite eigenvalues of $A - \lambda B$ (if any) are in the block $A_f - \lambda B_f$ but their multiplicities or Jordan structures are not computed by **GUPTRI**. Thus, $A_{reg} - \lambda B_{reg}$ in (2.5) is built up by these three regular blocks.

Since we only make use of unitary equivalence transformations in **GUPTRI** it is straight forward to show that the resulting generalized Schur decomposition can be expressed in finite arithmetic as

$$P^*((A+\delta A)-\lambda(B+\delta B))Q = \begin{bmatrix} A_r-\lambda B_r & * & * \\ 0 & A_{reg}-\lambda B_{reg} & * \\ 0 & 0 & A_l-\lambda B_l \end{bmatrix}, \qquad (4.2)$$

where $*$ denotes arbitrary conforming submatrices. Let

$$\delta_\sigma \equiv (\mathtt{ADELTA}^2 \cdot \|A\|_E^2 + \mathtt{BDELTA}^2 \cdot \|B\|_E^2)^{\frac{1}{2}}, \qquad (4.3)$$

where $\delta_\sigma{}^2$ is the sum of the squares of all singular values interpreted as zeros (deleted singular values) during the reduction to GUPTRI form. If **ZERO** is set to true on input to **GUPTRI**, δ_σ is an accurate estimate of $\|(\delta A, \delta B)\|_E$ in (4.2). One interpretation is that **GUPTRI** computes an exact generalized Schur decomposition (with a certain Kronecker structure) for a pencil $A'-\lambda B'$ within distance δ_σ from the input pencil $A-\lambda B$. This δ_σ is an upper bound on the distance from $A-\lambda B$ to the nearest pencil with the Kronecker structure **GUPTRI** reports.

References

[1] T. Beelen and P. Van Dooren. An improved algorithm for the computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 105:9–65, 1988.

[2] T. Beelen, P. Van Dooren, and M. Verhaegen. A class of staircase algorithms for generalized state space systems. In *Proceedings of the American Control Conference*, pages 425–426, Seattle, Wash., 1986.

[3] J. Demmel and B. Kågström. Stably computing the Kronecker structure and reducing subspaces of singular pencils $A-\lambda B$ for uncertain data. In Jane Cullum and Ralph A. Willoughby, editors, *Large Scale Eigenvalue Problems*, pages 283–323. North-Holland, Amsterdam, 1986. Mathematics Studies Series Vol. 127, Proceedings of the IBM Institute Workshop on Large Scale Eigenvalue Problems, July 8-12, 1985, Oberlech, Austria.

[4] J. Demmel and B. Kågström. Computing stable eigendecompositions of matrix pencils. *Lin. Alg. Appl.*, 88/89:139–186, April 1987.

[5] J. Demmel and B. Kågström. Accurate solutions of ill-posed problems in control theory. *SIAM J. Mat. Anal. Appl.*, 9(1):126–145, January 1988.

[6] J. Demmel and B. Kågström. The generalized Schur decomposition of an arbitrary pencil $A-\lambda B$: robust software with error bounds and applications. Part I: Theory and Algorithms. *ACM Trans. Math. Software, submitted*, pages 1–15, 1991.

[7] J. Demmel and B. Kågström. The generalized Schur decomposition of an arbitrary pencil $A-\lambda B$: robust software with error bounds and applications. Part II: Software and Applications. *ACM Trans. Math. Software, submitted*, pages 1–29, 1991.

[8] J. Dongarra and E. Grosse. Distribution of mathematical software via electronic mail. *Communications of the ACM*, 30(5):403–407, July 1987.

[9] B. Kågström. On computing the Kronecker canonical form of regular $A - \lambda B$ pencils. In B. Kågström and A. Ruhe, editors, *Matrix Pencils*, pages 30–57. Springer-Verlag, Berlin, 1983. Lecture Notes in Mathematics, vol. 973, Proceedings, Pite Havsbad, 1982.

[10] B. Kågström. The generalized singular value decomposition and the general $A - \lambda B$ problem. *BIT*, 24:568–583, 1984.

[11] B. Kågström. RGSVD - an algorithm for computing the Kronecker canonical form and reducing subspaces of singular matrix pencils $A - \lambda B$. *SIAM J. Sci. Stat. Comp.*, 7(1):185–211, 1986.

[12] V. Kublanovskaya. An approach to solving the spectral problem of $A - \lambda B$. In B. Kågström and A. Ruhe, editors, *Matrix Pencils*, pages 17–29. Springer-Verlag, Berlin, 1983. Lecture Notes in Mathematics, vol. 973, Proceedings, Pite Havsbad, 1982.

[13] V. Kublanovskaya. AB-algorithm and its modifications for the spectral problem of linear pencils of matrices. *Num. Math.*, 43:329–342, 1984.

[14] V. Kublanovskaya and V. B. Chazanov. Spectral problems for matrix pencils: methods and algorithms, part I (in Russian). Preprint LOMI P-2-88, USSR Academy of Sciences, Leningrad, 1988.

[15] G. W. Stewart. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Review*, 15(4):727–764, Oct 1973.

[16] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 27:103–141, 1979.

[17] P. Van Dooren. Reducing subspaces: Definitions, properties and algorithms. In B. Kågström and A. Ruhe, editors, *Matrix Pencils*, pages 58–73. Springer-Verlag, Berlin, 1983. Lecture Notes in Mathematics, vol. 973, Proceedings, Pite Havsbad, 1982.

[18] J. H. Wilkinson. Linear differential equations and Kronecker's canonical form. In C. de Boor and G. Golub, editors, *Recent Advances in Numerical Analysis*, pages 231–265. Academic Press, 1978.

Waveform Relaxation for Circuit Simulation

A.E. Ruehli
IBM T.J. Watson Research Centre, New York

This first lecture will cover a general overview of the circuit analysis problem for VLSI and its solution by the waveform relaxation (WR) method. An overview is given of the general solution approach.

Today several codes have been successfully implemented using WR for the solution of very large VLSI circuits and substantial gains have been made in computational time as compared to the conventional incremental time (IT) approach. In this lecture, we give an overview of the techniques implemented and how they differ from the IT solution. Also, we identify the sources of the gain like the multirate properties and the matrix partitioning factors.

The partitioning or splitting step is done automatically in most WR codes at least for MOSFET transistors while the work for a more general class of transistor circuits is in the research stage. Feedback loops need to be handled by graph techniques and the hierarchy of the problem must be utilized in the solution approach. Some of the details of these issues are discussed and appropriate algorithms are given.

Recent Progress in Waveform Relaxation

A.E. Ruehli
IBM T.J. Watson Research Centre, New York

The research on waveform relaxation (WR) techniques has been under way for ten years. During this time, substantial improvements have been made in the efficiency, the algorithms, the mathematical understanding and the implementation of the approach. Still, much progress is being made at present in all the above areas by researchers in many different locations. In this talk, we give insights into the current research in WR for the circuit simulation problem.

One of the topics of interest in the circuits area for WR is the partitioning for circuits with additional parasitic nodes formed by dynamic circuit elements. In this talk we will discuss the impact of this type of circuit on the convergence behavior. It can be shown that the convergence is rapid for a window in time for an important class of circuits. These results suggest that partitioning or splitting can be further improved from the presently applied techniques.

Other aspects which will be covered also are other scheduling algorithms which improve the efficiency like Epsilon scheduling. Finally, some results will be given on the application of WR to parallel processing.

On waveform relaxation methods for solving parabolic partial differential equations

Stefan Vandewalle
Katholieke Universiteit Leuven, Department of Computer Science
Celestijnenlaan 200A, B-3001 Leuven, Belgium

The Waveform Relaxation method has been used successfully for solving very large systems of ordinary differential equations of initial-value type. Its multirate integration characteristics and the ease by which it can be implemented on parallel machines are central to its success in commercial applications, such as the simulation of VLSI-devices and of chemical distillation processes.

In this talk we consider the applicability of the waveform relaxation method for solving parabolic partial differential equations of initial-boundary value and time-periodic type. The basic idea is to apply the waveform iteration to the set of ordinary differential equations that arise in the numerical method of lines, i.e., after semi-discretization of the partial differential equation. The resulting iteration can be accelerated by using a multigrid approach. We will analyze this algorithm and compare its performance to that of standard parabolic problem solvers. It will be shown that the method has a good sequential complexity, that it can be parallelized with low parallel overheads and that it can be vectorized efficiently. Furthermore, we will illustrate that a particular version of the algorithm can be understood as a standard multigrid solver defined on a space-time grid.

We will also indicate the advantages of the waveform relaxation approach, when it is used within a shooting algorithm for solving autonomous time-periodic differential equations. The availability of an approximation of the solution profile along the whole time-interval, instead of on only one time level, results in an additional substantial reduction of computing time.

Deelnemers

1	A.O.H.	Axelsson	KUN	Nijmegen
2	A.	Bellen	Univ. Trieste	Trieste
3	A.C.	Berkenbosch	TUE	Eindhoven
4	J.W.	Boerstoel	NLR	Amsterdam
5	E.F.F.	Botta	RUG, Math. Inst.	Groningen
6	J.C.	Butcher	Univ. Auckland	Auckland/Leiden
7	F.	Chatelin	IBM	Parijs
8	K.	Dekker	TUD	Delft
9	Th.J.	Dekker	UvA, Vakgr. Comp.Syst.	Amsterdam
10	J.L.M.	van Dorsselaer	RUL, Math. Inst.	Leiden
11	J.C.H.	van Eijkeren	RIVM	Bilthoven
12	L.	Elsner	Univ. Bielefeld	Bielefeld
13	J.F.	Frankena	UT	Enschede
14	C.W.	Gear	NEC Research	Princeton
15	A.J.	Geurts	TUE	Eindhoven
16	J.A.	van de Griend	RUL, Math.Inst.	Leiden
17	P.P.N.	de Groen	VU Brussel	Brussel
18	J.	de Groot	Philips Nat.Lab.	Eindhoven
19	P.W.	Hemker	CWI	Amsterdam
20	J.A.	Hendriks	VU, Fac. W&I	Amsterdam
21	K.J.	in 't Hout	RUL	Leiden
22	P.J.	van der Houwen	CWI	Amsterdam
23	W.H.	Hundsdorfer	CWI	Amsterdam
24	J.K.M.	Jansen	TUE	Eindhoven
25	E.F.	Kaasschieter	TUE	Eindhoven
26	B.	Kagström	Univ. Umea	Umea
27	B.	Koren	CWI	Amsterdam
28	J.F.B.M.	Kraaijevanger	RUL	Leiden
29	D.G.	Liu	RUL	Beijing/Leiden
30	M.	Louter-Nool	CWI	Amsterdam
31	R.M.M.	Mattheij	TUE, Fac. W&I	Eindhoven
32	K.	Meerbergen	KU Leuven	Heverlee
33	M.H.C.	Paardekooper	KUB	Tilburg
34	K.	Potma	UvA, Vakgr. Comp.Syst.	Amsterdam
35	A.A.	Reusken	TUE, Fac. W&I	Eindhoven
36	H.J.J.	te Riele	CWI	Amsterdam
37	A.E.	Ruehli	IBM	New York
38	W.H.A.	Schilders	Philips Nat.Lab.	Eindhoven
39	R.M.S.	Schulkes	DAMTP	Cambridge
40	G.L.G.	Sleijpen	RUU, Fac. W&I	Utrecht
41	A.	van der Sluis		Utrecht
42	B.P.	Sommeijer	CWI	Amsterdam
43	M.N.	Spijker	RUL	Leiden
44	R.P.	Stevenson	RUU, Fac. W&I	Utrecht
45	Th.L.	van Stijn	Icim B.V.	Rijswijk
46	J.H.M.	ten Thije Boonkkamp	TUE, Fac. W&I	Eindhoven
47	C.R.	Traas	UT, Toegep.Wisk.	Enschede
48	R.A.	Trompert	CWI	Amsterdam
49	S.	Vandewalle	KU Leuven	Heverlee
50	A.E.P.	Veldman	RUG, Math.Inst.	Groningen
51	K.G.	Verboom	Waterloopkundig Lab.	Delft
52	J.G.	Verwer	CWI	Amsterdam
53	G.A.L.	van de Vorst	TUE, Fac. W&I	Eindhoven
54	H.A.	van der Vorst	RUU, Fac. W&I	Utrecht
55	C.	Vuik	TUD	Delft
56	P.	Wesseling	TUD	Delft
57	M.C.J.	van de Wiel	Philips Nat.Lab.	Eindhoven
58	P.M.E.J.	Wijckmans	TUE	Eindhoven
59	F.W.	Wubs	RUG	Groningen
60	L.	Wuytack	Univ.Antwerpen VIA	Wilrijk
61	P.A.	Zegeling	CWI	Amsterdam