CWI Syllabi

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

The CWI is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

CWI Syllabus 4

Colloquium topics in applied numerical analysis

Volume 1

J.G. Verwer (ed.)

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

1980 Mathematics Subject Classification: 65XX06
ISBN 90 6196 281 1

PREFACE

The colloquium "Topics in Applied Numerical Analysis" was held at the Department of Numerical Mathematics of the Centre for Mathematics and Computer Science during the academic year 1983/1984. The aim of this colloquium was to draw attention to the widespread use of numerical mathematics in scientific real life problems, as well as to foster co-operation between mathematicians working in an academic environment and representatives from industries and institutes where the numerical solution of real life problems is studied.

These two volumes contain, in complete form, the papers presented by the speakers among the participants. Also it contains a contribution by J.L.O. Vranckx who, at the last minute, was unable to attend the colloquium in person.

The greater part of the papers deal with the numerical solution of a certain mathematical problem from practice. It was very interesting for the participating mathematicians to attend the lectures of the practitioners and to see the wide range of difficult technical problems which arise in, e.g., the engineering sciences.

The success of the colloquium was due principally to the speakers. To all of them I extend my sincere thanks and appreciation.

October 1984

J.G. Verwer (ed.)

CONTENTS VOLUME 1

CONTENTS VOLUME 2

NAMES AND ADDRESSES OF THE AUTHORS

H. Akse
Landbouwhogeschool Wageningen, Sectie Proceskunde,
De Dreijen 12, 6703 BC Wageningen.

G.L.M. Augenbroe
Technische Hogeschool Delft, Afd. Civiele Techniek,
Vakgroep Bouwfysica, Stevinweg 1, 2628 CN Delft.

A.O.H. Axelsson
Katholieke Universiteit Nijmegen, Mathematisch Instituut,
Toernooiveld, 6525 ED Nijmegen.

M. Bakker
Centrum voor Wiskunde en Informatica,
Kruislaan 413, 1098 SJ Amsterdam.

P.M. van den Berg
Technische Hogeschool Delft, Afd. Elektrotechniek,
Postbus 5031, 2600 GA Delft.

J.J. Bisschop
Shell Research B.V.,
Badhuisweg 3, 1031 CM Amsterdam.

J.W. Boerstoel
Nationaal Lucht- en Ruimtevaartlaboratorium,
Postbus 90502, 1006 BM Amsterdam.

B.J. Braams
FOM-instituut voor Plasma-Fysica,
Postbus 1207, 3430 BE Nieuwegein.

R. de Bruin
Rekencentrum der Rijksuniversiteit Groningen,
Postbus 80, 9700 AV Groningen.

J.G.M. de Bruin
Neratoom B.V.,
Postbus 93244, 2509 AE 's-Gravenhage.

A. van Calster
Universiteit van Gent, Laboratorium voor Electronica,
Sint Pietersnieuwstraat 41, 9000 Gent, België.

C. Flokstra
Waterloopkundig Laboratorium "De Voorst",
Voorsterweg 28, 8316 PT Marknesse.

R.H.J. Gmelig Meyling
Universiteit van Amsterdam,
Instituut voor Toepassingen der Wiskunde,
Roetersstraat 15, 1018 WB Amsterdam.

J.P. Hollenberg
Rekencentrum der Rijksuniversiteit Groningen,
Postbus 800, 9700 AV Groningen.

F.J. Jacobs
Koninklijke Shell Exploratie & Productie Laboratorium,
Postbus 60, 2280 AB Rijswijk.

A. Kassies
Nationaal Lucht- en Ruimtevaartlaboratorium.
Postbus 90502, 1006 BM Amsterdam.

A. Kooistra
Instituut TNO voor Wiskunde, Informatieverwerking en Statistiek,
Postbus 297, 2501 DD 's-Gravenhage.

I. Kuijper
Neratoom B.V.,
Postbus 93244, 2509 AE 's-Gravenhage.

C.G. van der Laan
Rekencentrum der Rijksuniversiteit Groningen,
Postbus 800, 9700 AV Groningen.

D. Loret
Universiteit van Gent, Laboratorium voor Electronica,
Sint Pietersnieuwstraat 41, 9000 Gent, België.

G. de Mey
Universiteit van Gent, Laboratorium voor Electronica,
Sint Pietersnieuwstraat 41, 9000 Gent, België.

S.J. Polak
Nederlandse Philips Bedrijven B.V., ISA-ISC-TIS/CARD,
Gebouw SAQ 2, 5600 MD Eindhoven.

N. Praagman
Svasek B.V.,
Heer Bokelweg 146, 3032 AD Rotterdam.

W.H.A. Schilders
Nederlandse Philips Bedrijven B.V., ISA-ISC-TIS/CARD,
Gebouw SAQ 2, 5600 MD Eindhoven.

G.S. Stelling
Dienst Informatie Verwerking, Rijkswaterstaat,
Nijverheidsstraat 1, 2288 BB Rijswijk.

A.G. Tijhuis
Technische Hogeschool Delft, Afdeling der Elektrotechniek,
Postbus 5031, 2600 GA Delft.

G.K. Verboom
Waterloopkundig Laboratorium,
Rotterdamseweg 185, 2600 MH Delft.

J. Vranckx
Agfa-Gevaert, N.V., Mathematisch Centrum 3523,
Septestraat 27, B-2510 Mortsel, België.

A. Wachters
ISA-ISC-TIS/CARD, Nederlandse Bedrijven B.V.,
Gebouw SAQ 2, 5600 MD Eindhoven.

J.B.T.M. Willemse
Dienst Informatieverwerking, Rijkswaterstaat,
Nijverheidsstraat 1, 2288 BB Rijswijk.

W. Zijl
Dienst Grondwaterverkenning TNO,
Postbus 285, 2600 AG Delft.

GAS-SOLID ADSORPTION SIMULATION

H. AKSE

Adsorption of one or more components from gaseous mixtures is a typical class of transient chemical- and environmental engineering processes.

Removal of undesired components from a gaseous stream takes place in a fixed bed of porous adsorbent particles. Fixed bed adsorbers for such purposes present a frequent but difficult problem for process engineers.

Apart from the assessment of a cyclic process of adsorption and desorption the mathematical modelling of the physical phenomena is rather difficult and complicated.

The set of partial differential equations containing the mass- and heat balanses of the system is far from lineair in character. In the literature numerical solutions are found for such a set of differential equations using the Runga-Kutta or modified Euler discretization procedures. Such procedures, however, are of the explicit type. For the system under consideration this will mean that rather large computation times are needed. Even the explicit Cranck-Nicholson discretization procedures needs for this system too much computer time and therefore is too expensive for frequent use in a technical context.

Besides the necessity of reducing the complexity of the system, without loosing sight of the reality, a fast algorithm is needed in order to reduce both computation time and computation costs.

In cooperation with the Centre for Mathematics and Computer Science at Amsterdam, the Netherlands, a new procedure is set up using an implicit scheme following the Newton Raphson procedure.

The reduction of computation time and costs compared with those using above mentioned procedures are considerable, while the differences in the results for either of the procedures are very small.

1. INTRODUCTION

A theoretical analysis of simultaneous heat and mass transfer during transient adsorption on porous granular solid particles in a fixed bed has been used to identify the significant processes determining the dynamic behaviour. The practical interest for mass transfer processes in fixed beds has resulted in numerous papers on the subject in general. The specific interest for adsorption as a fixed bed mass transfer operation has, however, increased only recently. The presently increasing demand for high purity materials in many industrial sectors together with the increasing needs to remove harmful components from waste flows in order to prevent environmental pollution have caused the mentioned increase of interest.

The unique advantage of adsorption processes is their ability to reduce concentrarion of adsorbable components to extremely low values combined with a relatively high adsorptive capacity of the used adsorbent. Otherwise stated, adsorption processes are extremely selective between the solvent or carrier gas and the minor constituants to be adsorbed. Adsorption works for concentrations so low that other separation methods-absorption extraction, destillation etc.-will fail.

Adsorption is usually applied in a cyclic fixed bed process. A cycle consists of a loading phase, followed by a regeneration phase. In the loading phase the fluid enters one end of the bed and passes from bed layers with relative high adsorbate load to layers with lower loads. Along this path adsorbate is transferred from fluid to fixed bed. A concentration profile and a particle load profile are slowly progressing in the fluid flow direction, so never will anywhere in the bed exist a steady state. This transient character of the process, combined with the generally complex sorption equilibrium relation has precluded the application of analytical solutions for adsorption column performance prediction.

Only for a few special simplified cases [1,2,3,4] analytical or semi-analytical solutions for concentration and particle load as functions of time and spatial location in the bed are available.

A strictly fundamental approach of the dynamic behaviour of an adsorption column is inappropriate for equipment design goals. Nevertheless, some attempts have been made in this direction [5,6]. The resulting very complex models will inevitably surpass the potentialities of even the fastest computer system, available to day. Apart from this, experience has

shown that a high level of complexity will not with certainty generate sufficient agreement with experimental results. Reality is known to be complex but apparently it is complex in an other way than presumed in some extremely complex models.

Simplification of the adsorption process model implying a significant reduction in computer costs can be attained without loss of agreement with reality. Therefore a set of simplifications has been introduced in order to develop a convenient mathematical model for practical design purposes. The development of such a convenient mathematical model for the drying of moist (compressed) air for industrial application is performed in cooperation with DELAIR [7].

The following simplifying assumptions are made:

1. The gas is in plug flow
2. The longitudinal dispersion for heat and mass in the gas are negligible.
3. There is only one component in the gas to be adsorbed.
4. Pressure drop along the bed is negligible.
5. Removal of adsorbate does not affect the gas flow rate.
6. Physical properties in the system are constant.
7. The gas is incompressible.

Adsorptions from gaseous mixtures are highly exothermal processes. These processes will probably not be isothermal even not approximately. Negligible heat loss to the surroundings, leading to an adiabatic process is not improbable. Generally, however, these processes are nor isothermal neither adiabatic. The rise in temperature, due to the release of heat of adsorption causes a decreasing adsorption capacity. In designing gas-solid adsorbers the effects of this heat release have to be taken into account.

Between the concentration of the sorbate in the fluid and the amount of uptake by the solid, the loading of the sorbent, exists a thermodynamically determined equilibrium correlation, the 'sorptionisotherm'.

(1) $q = q(c)$ *).

*) See list of symbols.

As stated before this equilibrium is also affected by temperature:

$$q = q(c,T_s). \tag{2}$$

The mathematical representation of the relations may be proportional, rectilinear or curvilinear. Most sorptionisotherms are curvilinear.

As shown in the Appendix sorptionisotherms comprise several parameters. These parameters are determined by optimal fitting of experimental data. In literature more than hundered different sorption equations have been reported for a variety of materials [12]. However, for the system watervapor-silicagel or watervapor-activated aluminiumoxyde adequate information is still lacking.

Uhlmann [9] has used a Freundlich isotherm in his work (watervapor-silicagel):

$$q = k_1(T_s).\left\{\frac{P.p_d}{P-p_d}\right\}^{k_2(T_s)} \tag{3}$$

in which

$$c = \frac{M_d}{M_a}.\frac{p_d}{P-p_d}\ .$$

From the measured isotherms for several temperature levels the heat of adsorption is determined according to Clausius-Clapeyron. The best fit represents an empirical relation between the heat of adsorption and the loading of the sorbent:

$$\Delta H_{tot} = 2.84.10^6 + \frac{10^6}{127.5\ .\ \bar{q}+0.556}\ [J/kg]. \tag{4}$$

In the work described here, a Polanyi type isotherm is used:

$$p_d = p_d^s.\exp(E_q/RT_s) \tag{5}$$

in which:

$$E_q = A_q + B_q.\ell n\ \bar{q}\ [J/mol]. \tag{6}$$

Whilst

$$c = .622\ \frac{p_d}{P-p_d}. \tag{7}$$

The heat of adsorption leads to:

$$(8) \qquad \Delta H_{tot} = \Delta H_{con}(T_s) + \frac{E_q}{M_d} \cdot 1000 \ [J/kg].$$

2. MODEL OF THE PROCESS

Consider a fluid containing a dilute adsorbable component flowing through a fixed bed of gramular solid as shown in figure 1

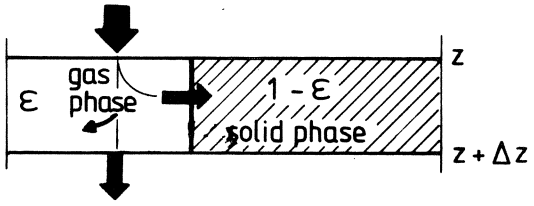


Fig. 1. Column segment with mass flows

From an overall mass balance over a fixed length element of the bed with voidage ε:

$$(9) \qquad \varepsilon\rho_f \cdot \frac{\partial \bar{c}}{\partial t} + u_s\rho_f \cdot \frac{\partial \bar{c}}{\partial z} + (1-\varepsilon)\rho_s \cdot \frac{\partial \bar{q}}{\partial t} = 0.$$

An exact description of the mass transfer process from the bulk of the flowing fluid through the fluid boundary layer adjacent to the particle into the pores and from these onto the adsorptive pore wall surface eventually followed by surface diffusion transport into the particle would be inappropriately complex. Sufficient agreement with reality might be obtained in describing the process with only one mass transfer parameter, an "overall" mass transfer coefficient. This route towards simplification is called the "boundary layer"-concept.
From figure 2 it is developed as follows:

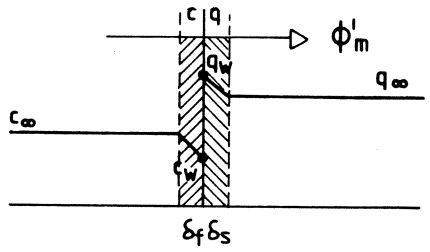


Fig. 2. Mass transfer to the boundaries

The mass transfer across the boundary layers with thicknesses δ_f and δ_s reads:

$$(10) \qquad \Phi'_m = (1-\varepsilon).\rho_s.\frac{\partial \bar{q}}{\partial t} = \rho_f.k_f.a(\bar{c}-c_w) = \rho_s.k_s.a(q_w-\bar{q}),$$

where k_f and k_s are partial mass transfer coefficients and k_s is derived by Glückauf [14] for a spherical particle with constant concentration changing rate:

$$(11) \qquad a.k_s = \frac{15\mathcal{D}}{r^2}$$

with the sorption isotherm known:

$$(12) \qquad q_w = q_w(c_w)$$

Elimination of q_w and c_w from (10) and (12) is feasible:

$$(13) \qquad (1-\varepsilon).\rho_s.\frac{\delta \bar{q}}{\delta t} = \rho_f.k_{of}.a(\bar{c}-c^*) = \rho_s.k_{os}.a(q^*-q)\ldots$$

with

$$(14) \qquad q^* = q^*(\bar{c}) \quad \text{or} \quad \bar{q} = \bar{q}(c^*).$$

In equation (13) k_{of} and k_{os} are now overall or lumped parameters for mass transfer refering either to the gas phase or to the solid phase. The concentration c^* for instance is defined as the gas concentration which would be in equilibrium with the mean solid loading $\bar{q}$. The overall mass transfer coefficient k_{of} relates the mass transferflux to the difference between $\bar{c}$ and c^*

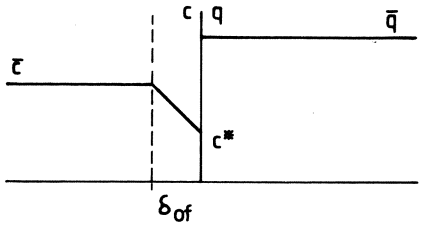


Fig. 3. Mass transfer model for the "boundary"-layer concept

Hence, by this manupulation the two boundary layers δ_f and δ_s are now combined in one apparent extended fluid boundary layer δ_{of} (figure 3). Such an approach works very well as argued by Cooney [11], Ferrell c.s. [13] and Uhlmann [9].

The concentration on loading $\bar{q}$ in the solid not being readily measurable it has to be derived from the gas phase concentration change by means of a mass balance. The left hand side of equation (13):

$$(13) \qquad (1-\varepsilon)\rho_s.\frac{\partial \bar{q}}{\partial t} = \rho_f.k_{of}.a.(\bar{c}-c^*)$$

serves as the accumulation equation for the sorbate in the solid phase.

An other simplifying concept found in the literature is the so called "particle"-concept [8]. In this concept the diffusional mass transfer resistance within the particle is assumed to be the major part of the total mass transfer resistance. The fluid phase boundary layer resistance is expressed as a minor correction on the diffusional resistance within the particle. This approach is less simple because a description of the concentration profile within the particle is now needed [8]. From figure 4 one may conclude:

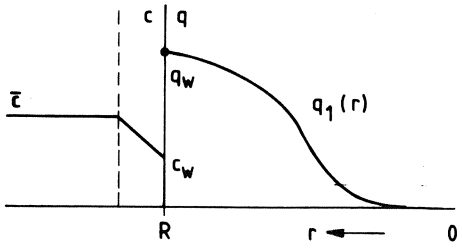


Fig. 4. Mass transfer model the "practicle-concept"

$$(15) \qquad \Phi''_m = \frac{(1-\varepsilon)}{a}.\rho_s.\frac{\partial \bar{q}}{\partial t} = \rho_f.k_f.(c-c_w) = \rho_s.\mathcal{D}_s\frac{\partial q}{\partial r}\Big|_{r=R}.$$

The accumulation in the solid for the "particle"-concept now reads:

$$(16) \qquad \frac{\partial q_i}{\partial t} = \nabla_r\{\mathcal{D}(q_i).\nabla_r q_i\}$$

or for a diffusivity independent of q_i (16) reads:

$$(17) \qquad \frac{\partial q_i}{\partial t} = \frac{\mathcal{D}}{r^2}.\frac{\partial}{\partial r}(r^2\frac{\partial q_i}{\partial r}),$$

with the boundary conditions:

$$(18) \qquad \mathcal{D}\rho_s \frac{\partial q_i}{\partial r}\Big|_{r=R} = k_f\ \rho_f(c-c_w),$$

and

$$\frac{\partial q_i}{\partial t} = 3\mathcal{D}\frac{\partial^2 q_i}{\partial r^2}\Big|_{r=0}.$$

The mean loading of the solid is given by:

$$(19) \qquad \bar{q} = \frac{3}{R}\int_0^R r^2 q_i(r)dr.$$

For the heat transfer analogous considerations can be made. This yields for the "boundary layer"-concept:

$$(20) \qquad \varepsilon\rho_f c_{pf}.\frac{\partial T_g}{\partial t} + u_s\rho_f c_{pf}.\frac{\partial T_q}{\partial z} + \alpha.a.(T_g-T_s) + \pi d_b\alpha_w(T_g-T_w) = 0,$$

$$(21) \qquad (1-\varepsilon)\rho_s c_{ps}.\frac{\partial T_s}{\partial z} = \alpha.a.(T_g-T_s) + k_{of}.a.\rho_f(\bar{c}-c^*).(-\Delta H),$$

$$(22) \qquad m_w c_{pw}.\frac{\partial T_w}{\partial t} = \alpha_w.\pi d_b.(T_g-T_w) - Ud_i\pi(T_w-T_{sur}),$$

with the initial values:

$$(23) \qquad \begin{aligned} &c(0,t) = c_0 \\ &q(z,0) = q_{ini} \\ &T_g(0,t) = T_{g,in} \\ &T_s(z,0) = T_w(z,0) = T_{sur}. \end{aligned}$$

For the "particle"-concept is found:

$$(24) \qquad \rho_s c_{ps} \cdot \frac{\partial T_i}{\partial t} = \lambda \nabla_r^2 T_i + (-\Delta H) \cdot \rho_s \cdot \frac{\partial q_i}{\partial t}.$$

with boundary conditions

$$(25) \qquad -\lambda \frac{\partial T_i}{\partial r}\Big|_{r=R} = \alpha(T_s - T_g) \text{ and } \frac{\partial T_i}{\partial t} = 3 \cdot \lambda \frac{\partial^2 T}{\partial r^2}\Big|_{r=0}.$$

From a mathematical point of view these two models are quite different in character. Equations (9), (13), (14), (20) to (23) form a nonlinear set of hyperbolic partial differential equations for the "boundary layer"-model with initial values for the dependent variables.
The equations (9), (16) to (19), (24), (25) and (20) to (22), however form a nonlinear mixed set of parabolic and hyperbolic partial differential equations.

3. NUMERICAL APPROXIMATION

Reminding the aim of this investigation - the assessment of appropriately convenient design methods-numerical approximation was applied to the simpler "boundary-layer"-concept. Obviously this concept will consume less computer time and memory than the more elaborate "particle"-concept. This is a practical advantage. A simplification of the equations can be achieved by transformation from the Eulerian to the Lagrangian form following the method of characteristics. This yields a set of nonlinear ordinary differential equations. Such a transformation may be achieved by:

$$(26) \qquad \tau = \frac{k_{of} \cdot a \cdot \rho_f}{(1-\varepsilon)\rho_s} \cdot \left(t - \frac{\varepsilon z}{u_s}\right)$$

and

$$(27) \qquad \zeta = \frac{k_{of} \cdot a}{u_s} \cdot z.$$

Introducing these new independent variables τ and ζ into the set of hyperbolic equations and also introducing new dependent variables:

$$\eta = c/c_o, \quad \xi = \bar{q}/q(c_o), \quad \theta_g = T_g/T_{in}, \quad \theta_s = T_s/T_{in}, \quad \theta_w = T_w/T_{in}$$

yields:

$$\frac{d\eta}{d\zeta} = -(\eta-\eta^*) \tag{28}$$

$$\frac{d\xi}{d\tau} = N_2.(\eta-\eta^*) \tag{29}$$

$$\frac{d\theta_g}{d\zeta} = N_3.(\theta_s-\theta_g) + N_4.(\theta_g-\theta_w) \tag{30}$$

$$\frac{d\theta_s}{d\tau} = N_5.(\theta_s-\theta_w) + N_6(\eta-\eta^*) \tag{31}$$

$$\frac{d\theta_w}{d\tau} = N_7(\theta_g-\theta_w) + N_8(\theta_w-\theta_{sur}) \tag{32}$$

$$\xi = \xi(\eta^*) \quad \text{or} \tag{33}$$

$$\eta^* = \eta^*(\xi) = g(\xi), \tag{33a}$$

with the initial conditions:

$$\begin{aligned} &\eta(0,\tau) = 1 \\ &\xi(\zeta,0) = \xi_{ini} \\ &\theta_g(0,\tau) = 1 \\ &\theta_s(\zeta,0) = \theta_w(\zeta,0) = \theta_{sur}. \end{aligned} \tag{34}$$

This set of ordinary differential equations (28) to (34) has to be solved numerically. Analytical solutions can only be achieved by setting equation (33) as a linear or proportional sorptionisotherm and with isothermal conditions. The isothermal case, the equations (30) to (32) will then vanish, is only of interest for liquid-solid adsorption. Numerical solutions are found in the literature using Runga-Kutta and/or Euler algorithms [11,14]. These algorithms are used in an explicit scheme, because of the nonlinear character of the system. Uhlmann [9] has developed a special algorithm. The efficiency of this algorithm however is hardly tractible. All these published numerical procedures require an inattractive amount of computer time and memory. A new algorithm has been developed using a simple backwards discretization scheme [17]:

$$\frac{d\eta}{d\zeta} = \frac{\eta(\zeta,\tau)-\eta(\zeta-\Delta\zeta,\tau)}{\Delta\zeta} \tag{35}$$

$$\frac{d\xi}{d\tau} = \frac{\xi(\zeta,\tau)-\xi(\zeta,\tau-\Delta\tau)}{\Delta\tau} \; . \tag{36}$$

Introduction of these equations into the set of nonlinear ordinary differential equations (28) to (32) yields a set of nonlinear equations:

$$\eta(\zeta,\tau) = \frac{\Delta\zeta}{1+\Delta\zeta} \cdot [g\{\xi(\zeta,\tau)\} + \frac{1}{\Delta\zeta} \cdot \eta(\zeta-\Delta\zeta),\tau] \tag{37}$$

$$\xi(\zeta,\tau) = N_2 \cdot \Delta\tau \cdot [\eta(\zeta,\tau) - g\{\xi(\zeta,\tau)\}] + \xi(\zeta,\tau-\Delta\tau) \tag{38}$$

and

$$\theta_g(\zeta,\tau) = \frac{N_4 \cdot \Delta\zeta}{1+(N_7+N_8) \cdot \Delta\tau} \cdot \{\theta_w(\zeta,\tau-\Delta\tau) + N_8 \cdot \theta_{sur} \cdot \Delta\tau\} + \theta_g(\zeta-\Delta\zeta,\tau) +$$

$$+ \frac{N_3 \cdot \Delta\zeta}{1+N_5 \cdot \Delta\tau} \cdot \{\theta_s(\zeta,\tau-\Delta\tau) + N_6 \cdot \Delta\tau[\eta(\zeta,\tau) - g\{\xi(\zeta,\tau)\}] \tag{39}$$

$$\theta_s(\zeta,\tau) = \{\theta_g(\zeta,\tau) \cdot N_5 \Delta\tau + \theta_s(\zeta,\tau-\Delta\tau) +$$

$$N_6 \cdot \Delta\tau[\eta(\zeta,\tau) - g\{\xi(\zeta,\tau)\}]\}/(1+N_5\Delta\tau) \tag{40}$$

$$\theta_w(\zeta,\tau) = \{\theta_g(\zeta,\tau) \cdot N_7 \cdot \Delta\tau + \theta_w(\zeta,\tau-\Delta\tau) +$$

$$N_8 \cdot \theta_{sur} \cdot \Delta\tau\}/(1+(N_7+N_8)\Delta\tau) . \tag{41}$$

The sorptionisotherm*):

$$g\{\xi(\zeta,\tau)\} = \eta^*(\zeta,\tau) = \eta^*\{\xi(\zeta,\tau)\}. \tag{42}$$

This set of equations can only be solved iteratively because of the nonlinearity, which is enclosed in the equations (37) and (38). Following an iterative procedure according to Newton-Raphson we need a function that has a zero point for the solution searched. To achieve this we replace

*) The dependence on temperature is not expressed in (42)

$\eta(\zeta,\tau)$ in (38) for the expression given in (37) with the result:

$$f\{\xi(\zeta,\tau)\} = 0 = \xi(\zeta,\tau) - \xi(\zeta,\tau-\Delta\tau) + \tag{43}$$
$$+ \frac{\Delta\tau . N_2}{1+\Delta\zeta} [g\{\xi(\zeta,\tau)\} - \eta(\zeta-\Delta\zeta,\tau)].$$

Letting $\xi_o(\zeta,\tau)$ a plausible starting value for (43) the Newton-Raphson gives:

$$d\xi_i(\zeta,\tau) = \frac{f\{\xi_i(\zeta,\tau)\}}{f'\{\xi_i(\zeta,\tau)\}} \tag{44}$$

where

$$d\xi_i(\zeta,\tau) = \xi_i(\zeta,\tau) - \xi_{i+1}(\zeta,\tau) \tag{45}$$

and

$$f'\{\xi_i(\zeta,\tau)\} = 1 + \frac{N_2.\Delta\tau}{1+\Delta\zeta} \cdot g'\{\xi_i(\zeta,\tau)\}. \tag{46}$$

The algorithm only holds when $f' \neq 0$. This condition is surely fulfilled because the derivative of g with respect to ξ cannot be negative.
After each iteration value for ξ all the other dependent variables are calculated using equations (37) and (40) to (42) until $d\xi$ has reached a given minimum value (eg. 10^{-5}). The iteration is very fast and converges rapidly because of the fact that $\xi \leq 1$. For every new step $\xi(\zeta,\tau-\Delta\tau)$ will act as a new plausible starting value for (43).
Consumption of computer time and costs for the new numerical procedure has been proved to be very low in comparison with the above mentioned RK or Euler procedures. For a simulation of 24 hours of the real process the new procedure needs only 20 à 25% of the computer time needed for the RK procedure and 15-20% of the computer time needed for the particle concept.
Recently the "method of lines" has been introduced in literature [10,16] to solve the original non-linear set of hyperbolic partial differential equations, discussed in this paper. The spatial derivatives are discretized into N + 1 grid points using biased upwind five point difference formulae. For a set of n hyperbolic equation this yealds n*(N+1) ordinary differential equations in time. They can be solved simultaneously by

some standard software package such as DGEAR. Reported needs for computer time indicates that the method of lines will consume at least twice as much computer time in comparison with the new procedure.
A block-diagram of the program according to the new procedure is given in figure 5.

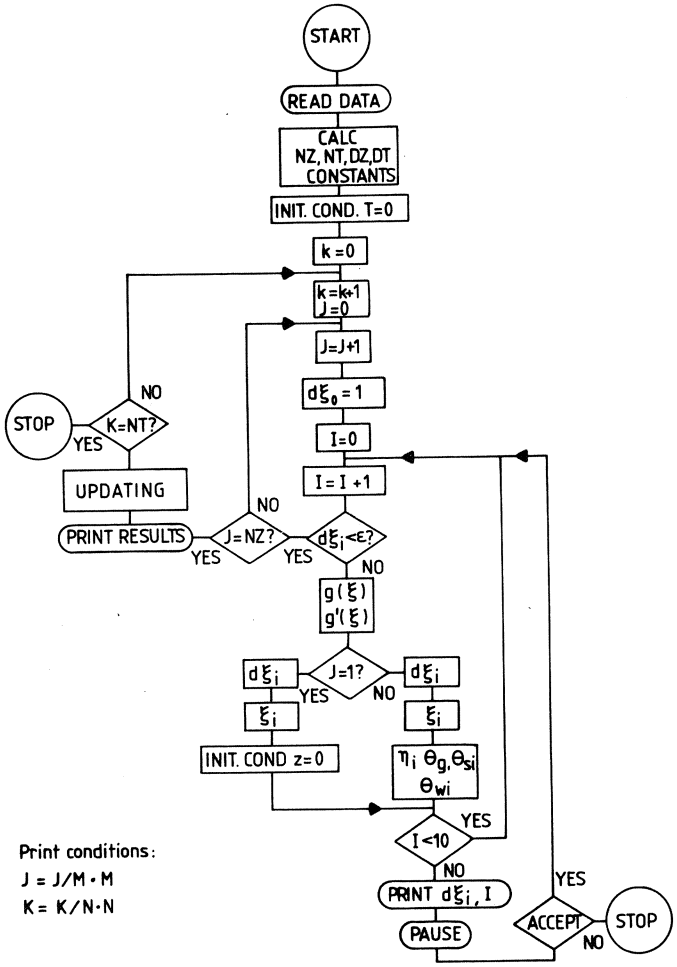


Fig. 5. Block diagram for the new procedure

4. RESULTS AND DISCUSSION

The results obtained from the new procedure appear to have no significant deviation from those obtained from the other mentioned procedure as shown in the figures 6 to 8. The fully drawn curves in figure 6 and 7 respectively represent the concentration - and temperature as a function of time in the fixed bed as obtained from the RK-procedure [15]. The dots represent the result as obtained with the new procedure.

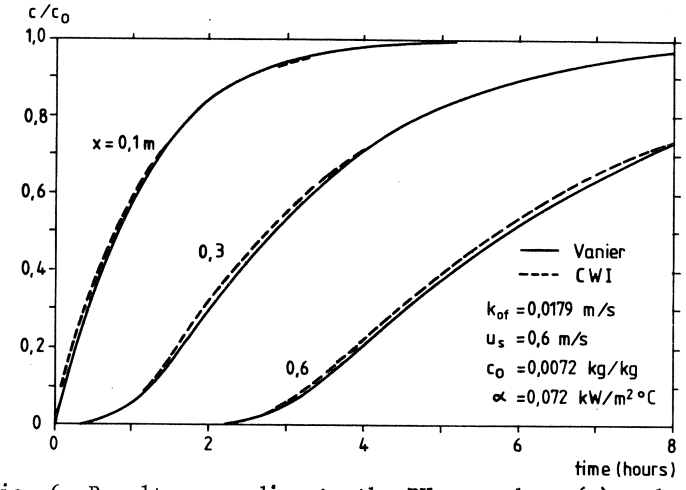


Fig. 6. Results according to the RK-procedure (-) and the new procedure (---) for the concentration profiles

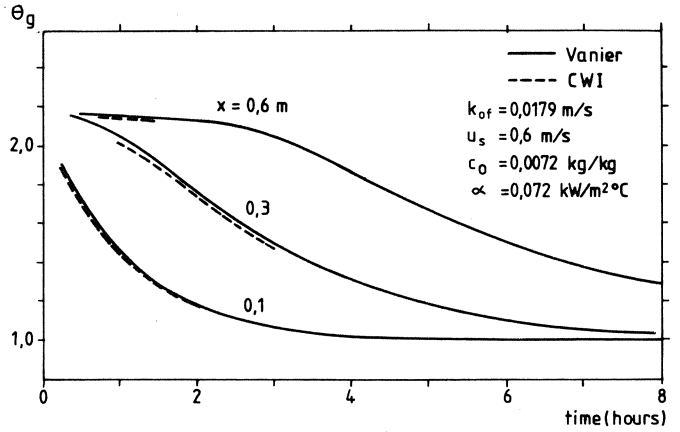


Fig. 7. As figure 6, temperature profiles

In figure 8 a comparison is made between the "boundary-layer"-concept and the "particle"-concept. Curve 2 represents a concentration profile with respect to time for the "particle"-concept for some location in the fixed bed, while the triangles represent the result for "the boundary"-layer-concept according to the new procedure. This means that a typical combination of a diffusivity (eg. .3,2.10^{-10} m^2/s) and partial mass transfer coefficient (eg, 0.12 m/s) in the "particle"-concept may be combined to an "overall"-mass transfer coefficient (eg. 0.18 m/s) for the "boundary"-layer concept. Decreasing the partial mass transfer coefficient in the "particle"-concept from .12 to .018 m/s results in a significant change in the shape of the concentration profile. A high value of the resistance for mass transfer within the particle will result in a minor influence of the boundary layer mass transfer coefficient, k_f, as shown in curve 1.

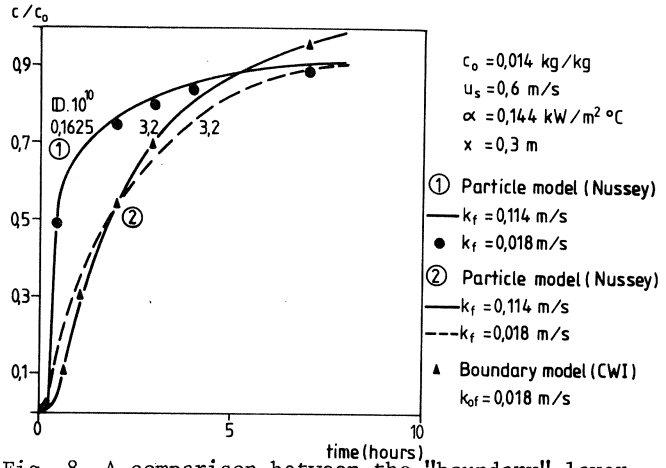


Fig. 8. A comparison between the "boundary"-layer- and the "particle"-concept

Apart from the distinction between those models with respect to the need of computer time, the "particle"-concept is more unfavourable because adequate information on diffusivities of watervapor into adsorbents like silicagel will still fail. For a particular set of inlet conditions the shape of the concentration profiles will be significantly affected by the shape of the sorptionisotherm and the mass transfer parameter(s). Quantitative estimation of diffusivities from the gaseous state is an eleborate operation for materials like silicagel with an at random structure.

This aspect of the "particle"-concept gives therefore too many incertainties to make it a recommandeble model.

For the "boundary"-layer model however a number of incertancies is still left eg. the overall mass transfer coefficient, k_{of}, and the shape of the sorptionisotherm, represented by the constants A_q and B_q in the Polanyi equation (6). In figure 9 some experimental results are shown. The fully drawn curves give the results of the model according to the new procedure while the dotted lines represents experimental results for a number of locations in the bed. This results are obtained by monitoring local concentration during operation of the process. It may be seen from figure 9 that agreement between the theoretical and experimental results is still poor for the higher concentration levels. This disagreement might be caused by failing measurement of moist content in the air

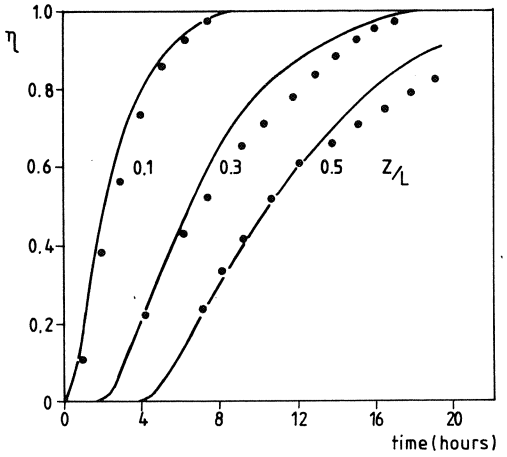


Fig. 9. A comparison between the theoretical (new procedure) and the experiment

This measurement of watervapor concentration in moist air has proved to be rather difficult. Adequate and reliable instrumentation for this purpose is still lacking. During the passage of a breakthrough profile watervapor concentration in the air will vary from 5.10^{-6} kg H_2O/kg dry air to 0.015 kg H_2O/kg dry air within a period of 6 to 8 hours. This is quite a large range in a fairly short time, indeed.

The main fails of most of the moisture measuring instruments are:

- instability with respect to time (ageing) frequent calibration is needed. The calibration operation is rather elaborate.
- Measuring range too small
- slow responsility
- hysteresis.

As stated above incertacnties also remain on the value of the overall mass transfer coefficient and on the isotherm. Therefore the interpretation of the results as given in figure 9 will be questionable until the availability of reliable instrumentation for the measuring of watervapor concentration is realized.

LIST OF SYMBOLS

a	Specific area of the solid particles	[m^2/m^3 of bed]
A	Crossectional area of the column	[m^2]
A_q, B_q	Constants in equation 6	[J/mol]
c	Watervapor concentration in fluid phase	[kgH_2O/kg dry air]
c_p	Specific heat	[J/kg.K]
d_b	Diameter of column	[m]
d_i	Thickness of column insulation	[m]
$\mathcal{D}$	Diffusivity of watervapor in solid (mean)	[m^2/s]
E_q	Heat of adsorption	[J/mol]
ΔH_{tot}	Total heat release	[J/kgH_2O]
ΔH_{cond}	Heat of condensation	[J/kgH_2O]
k	Masstransfer coefficient	[m/s]
k_1, k_2	Temperature coefficients in sorption isotherm	[-]
M	Molecular weight	[kg/mol]
P	Total pressure	[Pa]
p	Partial pressure	[Pa]
q	Solid loading	[kgH_2O/kg solid]
r, R	(Partial) radius of spherical particle	[m]
R	Gasconstant	[J/mol.K]
T	Temperature	[C,K]
$t, \Delta t$	Time, timestep	[s]
U	Heat transmission coefficient	[$W/m^2.K$]
u	Fluid velocity	[m/s]
$z, \Delta a$	Spatial coordinate (step)	[m]
α	Heat transfer coefficient	[$W/m^2.K$]
η	Dimensionless concentration in fluid	[-]
ξ	Dimensionless loading of solid	[-]
ε	Bedporosity	[m^3/m^3bed]
ζ	Dimensionless spatial coordinate	[-]
τ	Dimensionless time coordinate	[-]
θ	Dimensionless temperature	[-]
ρ	Density	[kg/m^3]
λ	Heat conductivity of solid	[W/m.K]
Φ''_m	Massflux of watervapor	[kg/m^3.s]
ϕ'_m	Massflux of watervapor	[kg/m^2.s]

SUBSCRIPTS

d	watervapor
g	bulk of fluidum
o	inlet conditions
s	solid phase, superfical
f	fluidum phase
w	at the boundary fluid-solid, fluid-column wall
os	overall value boundary-layer solid side
of	overall value boundary-layer fluidum side
i	location within solid phase, interstitial
a	air
sur	surroundings condition

SUPERSCRIPTS

*	equilibrium state in overall mass transfer concept
-	mean value in the bulk (fluid or solid)
s	saturation state

REFERENCES

[1] ROSEN, J.B., I.E.C. 46 (1954) 1590.

[2] ROSEN, J.B., J. Chem. Phys. 20 (1952) 387.

[3] SCHUMANN, T.E.W., J. FRANKLIN Inst. 208 (1929) 405.

[4] TIEN, CHI & THODOS, A.I.Ch.E-J 5 (1959) 373.

[5] MEYER, D.A. & T.W. WEVER, A.I.Ch.E-J 13 (1967) 457.

[6] MAY-LING WANG, cs, Computer and Chem. Eng. 4 (1980) 85.

[7] DELAIR B.V., Prinsenbeek (The Netherlands) Systems and equipement for separation of moisture and traced components from air, gases, liquids and hydrocarbons. Instrument air conditioning, dehumidification of ambiant and process air. Heat recovery from air and gasstreams.

[8] NUSSEY, C., Ph D Thesis University of Leeds 1967.

[9] UHLMANN, H.J., Dissertation Universität Stuttgart 1976.

[10] SIRCAR, S., cs, IEC-Proc. Des. Dev. 22 (1983) 10.

[11] COONEY, D.O., IEC-Proc. Des. Dev. 13 (1974) 368.

[12] BERG, VAN DEN C., Dissertation Agriculture University Wageningen (The Netherlands) 1981.

[13] FERRELL, J.K., cs, IEC-Proc. Des. Dev. 15 (1976) 114.

[14] GLÜCKAUF, E., Trans. Far. Soc. 51 (1955) 1540.

[15] VANIER, C., Ph. D. Thesis Syracuse University USA 1974.

[16] YIU WAH WONG, cs, AICHE Symposium series 78 (1982) no. 219.

[17] BAKKER, M., Institute for Mathematics and Computer Science, Amsterdam (the Netherlands).

APPENDIX Examples of sorptionisotherms (A=A(T),B=B(T))

Henry : $q = A.c$ (proportional)

.... : $q = A.(c+1/B)$ (rectilinear)

Laugmuir : $q = \frac{A.c}{1+B.c}$ (curvilinear)

Meyers : $c = (q/A).\exp(B.q/A)$ (curvilinear)

Volmer : $c = \frac{q}{A-B.q} \exp(\frac{Bq}{A-Bq})$ (curvilinear)

Freundlich: $q = A.c^{B}$ (curvilinear)

Toth : $q = \{\frac{(A.c)^{n}}{1+(B.c)^{n}}\}^{1/n}$ (curvilinear).

A SURVEY OF VECTORIZABLE PRECONDITIONING METHODS FOR LARGE SCALE FINITE ELEMENT MATRIX PROBLEMS

O. AXELSSON

Direct solution methods for partial differential equation problems suffer from fill-in to such an extent that many problems, such as those derived from problems in three space dimensions on a fine mesh, cannot be solved even on presently available supercomputers.

Iterative methods on the other hand do not suffer from fill-in and with effective preconditioned and accelerated iterative methods one may derive algorithms of almost optimal order of computational complexity.

Traditional preconditioned methods are based on incomplete (pointwise) factorisations of the given sparse matrix and result in quite efficient algorithms on a scalar computer. Many of these methods are however not vectorizable in their original form. Furthermore, it is shown that some methods that have been proposed in recent years to overcome this, cannot be of a particularly small order of computational complexity.

Newly developed approximate factorization methods based on approximations of the inverses of diagonal block matrices, are however vectorizable to a large extent and at the same time possess a low order of computational complexity. Two variants of such methods are discussed:

i) For block matrices ordered lexicographically;

ii) For block matrices ordered according to a cyclic reduction ("odd-even") method.

We extend these methods for the application to partial differential equations in three space dimensions.

1. INTRODUCTION

We shall consider the numerical solution of very large but sparse linear algebraic systems

* The research reported in this paper was partly supported by the North Atlantic Treaty Organization, Brussels, through Grant No. 648/83.

$$A\underline{x} = \underline{b}, \quad \underline{x},\underline{b} \in R^N. \tag{1.1}$$

Although the methods we shall present do not always require this, for ease of presentation we shall assume that A is a nonsingular diagonally dominant M-matrix (i.e. $a_{ij} \leq 0$, $i \neq j$ and the entries of the inverse, A^{-1} are positive).

Such matrices may, for instance, arise when we apply a difference method or the lowest order finite element method for a diffusion equation. They also arise at each correction step of a defect - correction method for convection - diffusion problems, where the correction operator is derived from an artificial diffusion or upwind difference (or finite element) operator.

The order N may be very large. For example, for a scalar equation in three space dimensions (3D) on a $64 \times 64 \times 64$ mesh we get $N \approx 250000$. However, A has typically a band structure or skyline structure and in each row (and column) of the matrix, only a few, typically 5 or 7 entries are nonzeros.

Direct solution methods based on a factorization, $A = \hat{L}\hat{U}$ of A into lower and upper triangular factors, $\hat{L}$ and $\hat{U}$, respectively, produce, however, fill-in within most of the band or skyline and are hence costly. For very large problems, the matrix must be stored on peripheral storage and the I/O (i.e., disk file read/write) costs tend to dominate. On a scalar computer the factorization cost is $O(N^2)$, $d = 2$ and $O(N^{7/3})$, $d = 3$, respectively, where d is the space dimension. Demand of storage and forward and backward solution costs are $O(N^{1.5})$ and $O(N^{5/3})$ for $d = 3$, respectively.

To see what this means in practice, consider the above mentioned (64^3) mesh. Then we would need a storage of $2N^{5/3} = 2.64^5 \approx 2.10^9$ words for just one scalar problem, which is clearly not realistic on presently available computers.

Special ordering techniques (such as nested dissection, see George and Liu [10]) may reduce the operation count significantly, in particular for large 2D problems. For a 3D problem, however, factorization costs still $O(N^2)$ and storage increases as $O(N^{4/3})$.

Another, sometimes severe, drawback of direct solution methods is that round-off errors and errors in given data tend to increase as $O(h^{-3})$, where h is a stepsize parameter (see, for instance, Axelsson and Barker

[3]). This means that multiple precision arithmetric computations are mostly needed which adds even more to the overall costs (storage and hence I/O, and arithmetics).

Iterative solution methods on the other hand do not suffer from fill-in and with effective preconditioned and accelerated methods one may derive algorithms of almost optimal order of computational complexity.

Let $C = LU$ be an approximation, for instance, an incomplete factorization of A. C is often called a preconditioning matrix.

A basic iterative method has the form

$$C\underline{\delta}^{\ell+1} = -\underline{r}^{\ell}, \quad \underline{x}^{\ell+1} = \underline{x}^{\ell} + \underline{\delta}^{\ell+1}, \quad \ell = 0,1,2,\ldots, \tag{1.2}$$

of a *defect-correction method* where $\underline{r}^{\ell} = A\underline{x}^{\ell} - \underline{b}$ is the defect or residual and $\underline{\delta}^{\ell+1}$ is the correction at stage ℓ. $\underline{x}^{o}$ is arbitrary but a good choice is $\underline{x}^{o} = C^{-1}\underline{b}$.

Consider the splitting

$$A = C - R \tag{1.3}$$

of A, where R is the defect matrix. R is sparse, frequently even much sparser than A. Then (1.2) takes the form

$$C\underline{x}^{\ell+1} = R\underline{x}^{\ell} + \underline{b}, \quad \ell = 0,1,2,\ldots$$

which converges if and only if $\rho(C^{-1}R) < 1$, where $\rho(.)$ is the spectral radius. The theory of (quasi) regular splittings may be applied (see Varga [23] and Berman, Plemmons [6]). The rate of convergence as measured in the number of iterations to reach a relative error, $\|\underline{x}-\underline{x}^{k}\|/\|\underline{x}-\underline{x}^{o}\| \leq \varepsilon$ is

$$k \cong \ell n(1/\varepsilon)/\ell n(1/\rho_o), \tag{1.4}$$

where $\rho_o = \|C^{-1}R\|$.

For second order elliptic problems and if $C = D_A$, the (block) diagonal part of A, one gets $\rho_o = 1/(1+\zeta h^2)$, for some positive ζ, independent of h. Hence $k = O(h^{-2})$ which is unacceptable.

The efficiency of the simplest iterative methods may be improved in two ways

(i) by a proper choice of the preconditioning matrix C,

(ii) by use of some accelerated form of iterative method.

This paper deals with the first topic. As far as the second is

concerned we shall here just make a few comments.

An accelerated iterative method of the one-step Chebyshev type, takes the form

$$C(\underline{x}^{\ell+1}-\underline{x}) = -\tau_\ell(A\underline{x}-\underline{b}), \quad \ell = 0,1,2,\ldots .$$

Here $\{\tau_\ell\}$ is a sequence of acceleration parameters (taken in a proper order for reasons of stability). For the calculation of τ_ℓ we need to know bounds for the extreme eigenvalues. Conjugate gradient or conjugate direction methods do not need such information. The classical methods are for symmetric matrices but various generalizations to nonsymmetric problems exists, such as truncated and/or restarted minimum residual methods, Galerkin type methods, etc. For a recent survey, see Saad and Schultz [19]. For symmetric positive definite (SPD) problems, one may prove that $\|\underline{x}-\underline{x}^k\|_{A^{\frac{1}{2}}} \le \varepsilon\|\underline{x}-\underline{x}^0\|_{A^{\frac{1}{2}}}$ if

$$k = \text{int}\{\tfrac{1}{2}\kappa^{\frac{1}{2}}\ell n\frac{2}{\varepsilon} + 1\} \tag{1.5}$$

where $\kappa = \max_i\lambda_i/\min_i\lambda_i$, the spectral condition number and $\|x\|_{A^{\frac{1}{2}}} = \{\underline{x}^TA\underline{x}\}^{\frac{1}{2}}$, the so-called energy norm. Here $\{\lambda_i\}$ are the set of eigenvalues of $C^{-1}A$. The number of iterations in (1.5) is an upper bound, frequently it is overly pessimistic. For an SPD problem with $C = D_A$ as before we get now $k = O(h^{-1})$, an order of improvement over the basic method. This may be further improved upon by use of so-called modified incomplete factorization methods (see Axelsson and Barker [3]). One may prove that then (again for second order elliptic problems), $k = O(h^{-\frac{1}{2}})$. This results in a total cost of $O(N^{1.25})$, $d = 2$ and $O(N^{1.17})$, $d = 3$. Storage is of optimal order, $O(N)$, typically we need about twice as much storage as needed for A alone. For time dependent problems with time step say $k = O(h)$, we may improve the above costs to $O(N^{1.125})$, $d = 2$ and $O(N^{1.083})$, $d = 3$, respectively. This means that such iterative methods are of almost optimal order of computational complexity.

A further advantage with iterative methods is that, with a special implementation of the calculation of residuals (as sum of differences), round-off errors increase little ($O(h^{-1})$) or not at all. Hence the method is (almost) numerically stable for all values of h (for details, see Axelsson and Barker [3]).

An alternative to preconditioned iterative methods for partial differential equation problems are multigrid methods of various forms. In

particular, combinations of the preconditioning and multigrid techniques seem to result in very robust algorithms. This is, however, not the topic of this paper.

The main purpose of the present paper is to present preconditioning methods which are vectorizable, i.e., for which most of the computations may be done in parallel on a pipelined computer or on an array of processors.

In section 2 we discuss shortly the classical (pointwise) preconditioning iterative methods and point out why they are not vectorizable in their original form. We also show that some methods which have been proposed in recent years to overcome this, cannot be of a particularly small order of computational complexity.

In section 3 we discuss some newly proposed polynomial preconditioners which may result in improvements of the methods for certain pipelined arrays of computers.

Finally in sections 4 and 5 we discuss some newly developed approximate factorization methods based on approximate inverses and show that by a proposed modification, these methods become highly vectorizable at the same time possess a low order of computational complexity.

We present a new algorithm for the calculation of a sparse approximation of the inverse of matrices which typically arise in difference or finite element approximations of partial differential equations in two space dimensions. This enables us to extend an algorithm based on repeated odd-even reduction orderings for the calculation of a sparse preconditioner to the case of three dimensional problems. Each diagonal block of a three dimensional problem has namely the structure of a two dimensional problem.

2. PRECONDITIONING BY INCOMPLETE (POINTWISE) FACTORIZATION

The classical incomplete factorization methods as originally discussed in Varga [22], Meijerink and Van der Vorst [17] and in Axelsson and Munksgaard [5] can be presented shortly in the following form:

(i) Incomplete factorization by position

Before factorization is started, choose a set J of index pairs which is a subset of the complete set $\{(i,j),\ 1 \le i \le N,\ 1 \le j \le N\}$. Frequently $(i,j) \in J$ if and only if $a_{ij} \neq 0$. J always includes the set of diagonal indices, (i,i), $1 \le i \le N$.

The incomplete factorization principle is the following.

At the r'th stage we use the pivot row $a_{rj}^{(r)}$, $j = r, r+1,\ldots,N$ to eliminate nonzeros in the lower triangular part, $j = r$, $i = r+1,\ldots,N$ of the remaining part of the matrix. We get $a_{ij}^{(r+1)} = a_{ij}^{(r)} - a_{ir}^{(r)} (a_{rr}^{(r)})^{-1} a_{rj}^{(r)}$, $r + 1 \le i \le N$, $j \ge r + 1$, if $(i,j) \in J$. Otherwise, if $(i,j) \notin J$ we neglect the possible nonzero entry, or in the modified version of this algorithm, we add it to the diagonal (see Gustafsson [11]).

In this way, fill-in is neglected and hence cannot give rise to fill-in caused by previous fill-ins.

(ii) Incomplete factorization by value

This is performed as above but nonzero entries are accepted only if their absolute values are large enough

$$|a_{ij}^{(r+1)}| \ge c\{a_{ii}^{(r)} a_{jj}^{(r)}\}^{\frac{1}{2}},$$

where $c > 0$ is a (small) parameter.

The existence of such factorizations follows easily for M-matrices (i.e., it follows that all pivot entries are nonzero). More general matrix problems may be solved by use of the idea of spectral equivalence or by a defect-correction method, cf. section 1. For details and an application, see Axelsson and Barker [3].

Combined with proper acceleration methods, such methods perform quite well (see Section 1). Unfortunately, they are not directly vectorizable because the factorization must be done recursively (row by row) and the forward and backward solutions of $Lz = \underline{b}$ and $U\underline{x} = \underline{z}$, respectively, must also be solved recursively (see, however, remark in Section 5).

An attempt to overcome this has been taken by Van der Vorst [24]. He writes

$$C = (I-\tilde{L})D(I-\tilde{U}) \tag{2.1}$$

where $\tilde{L}$, $\tilde{U}$ are strictly lower and upper triangular, respectively. Then

$$C^{-1} = (I-\tilde{U})^{-1}D^{-1}(I-\tilde{L})^{-1}$$

and $(I-\tilde{U})^{-1}$ and $(I-\tilde{L})^{-1}$ are now approximated by truncated Neumann series.

We shall present a generalization of this to block matrices and perform

an analysis different from the one in [24]. Note that in that paper no analysis is performed regarding the spectral condition number.

On Approximate Inverse Preconditioning Based on Neumann Series Expansion of (Incomplete) Factorizations

As has been said, the usual preconditioners require the solution of linear systems with triangular matrices which is a recursive process and hence in general not well suited for vectorization. If we could derive a preconditioner for which we use an approximation Q of the inverse A^{-1}, the resulting preconditioning would be on the form

$$(2.2) \quad Q(A\underline{x}-\underline{b}) = 0$$

which is vectorizable, because it demands only *matrix times vector multiplications*.

At this point we remark that an efficient way of representing matrices for parallel processors is by sub-diagonals instead of by rows or columns, see Madsen et al. [16].

One such form of preconditioner is the polynomial preconditioner to be discussed in Section 3. Another possibility would be the following.

Construct at first a preconditioning or possibly a complete factorization of the form

$$(2.3) \quad C = (B^{-1}-L)B(B^{-1}-U)$$

where L, U are strictly lower and upper (block) triangular and B is a (block) diagonal nonsingular matrix.

We assume that $\|BU\| < 1$ and $\|LB\| < 1$. (This would be the case if, for instance, the given matrix A is (block) diagonally dominant.) This implies that I - LB and I - UB are nonsingular.

If C is constructed from a pointwise (incomplete) factorization method or a pointwise SSOR method, then B would be a diagonal matrix. If C is constructed from a blockwise (incomplete) factorization method, such as the one to be described in Section 4, or from a blockwise (line) SSOR method, then B would be a block diagonal matrix. It follows from (2.3) that

$$C = (I-LB)B^{-1}(I-BU)$$

and

$$C^{-1} = (I-BU)^{-1}B(I-LB)^{-1}.$$

By approximating $(I-BU)^{-1}$ and $(I-LB)^{-1}$ by a truncated Neumann series we arrive at the following approximation Q of the inverse A^{-1}:

$$(2.4) \quad Q = [I + BU + (BU)^2 + \ldots + (BU)^m]\, B\, [I + LB + (LB)^2 + \ldots + (LB)^m],$$

where we have used m + 1 terms in the two Neumann series expansions.

To simplify the analysis of the inverse matrix preconditioner Q, we assume now further that B is symmetric and that $U = L^T$. Note that then C is SPD.

Let

$$(2.5) \quad F = (I-LB)^{-1}, \quad R = F - [I + LB + \ldots + (LB)^m].$$

Then $(I-BU)^{-1} = F^T$, $C^{-1} = F^T BF$ and

$$(2.6) \quad Q = (F-R)^T B(F-R) = (I-F^{-1}R)^T C^{-1}(I-F^{-1}R).$$

Further

$$(2.7) \quad I - F^{-1}R = F^{-1}(F-R) = (I-LB)(I+LB+\ldots+(LB)^m) = I - (LB)^{m+1}.$$

From (2.6) and (2.7) it finally follows

$$(2.8) \quad Q = (I-E)^T C^{-1}(I-E)$$

where $E = (LB)^{m+1}$.

If A is a block tridiagonal matrix, then LB is blocksubdiagonal, and hence nilpotent. $(LB)^{m+1}$ has then nonzero blocks at the (m+1)'st subdiagonal away from the main diagonal.

We want to estimate the condition number of QA, which shall be done by somewhat heuristic arguments. We have

$$\frac{\underline{x}^T A^{-1}\underline{x}}{\underline{x}^T Q\underline{x}} = \frac{\underline{x}^T C^{-1}\underline{x}}{\underline{x}^T Q\underline{x}} \, \frac{\underline{x}^T A^{-1}\underline{x}}{\underline{x}^T C^{-1}\underline{x}}.$$

We shall consider three cases

(i) C is a (unmodified) incomplete factorization of A

(ii) C is a (modified) incomplete factorization of A

(iii) C is a complete factorization of A.

For case (i), it is easily seen that the largest value of $\underline{x}^T A^{-1}\underline{x}/\underline{x}^T C^{-1}\underline{x}$, i.e., the largest eigenvalue of CA^{-1}, is taken for a vector close to the first eigenvector of A (i.e., $[\sin \pi x_i \sin \pi y_j \sin \pi z_k]$ for a model Laplacian problem on a unit cube). This eigenvalue is $O(h^{-2})$, $h \to 0$. It is easily seen that $\underline{x}^T C^{-1}\underline{x}/\underline{x}^T Q\underline{x} \geq 1$ for this eigenvector. (Note that $U \geq 0$, $L \geq 0$.) It follows from (2.8) that an estimate of this is

$$(2.9) \qquad \underline{x}^T C^{-1}\underline{x}/\underline{x}^T Q\underline{x} \geq 1/(1-\|E\|^2)$$

where $\|E\| \leq \|LB\|^{m+1}$.

Since the smallest value of $\underline{x}^T A^{-1}\underline{x}/\underline{x}^T C^{-1}\underline{x}$ is $O(1)$, $h \to 0$ and because $\underline{x}^T C^{-1}\underline{x}/\underline{x}^T Q\underline{x} = O(1)$ for (oscillating) vectors for which this value is taken, we conclude that the spectral condition numbers are at any rate not improved by the use of Neumann series, and the estimate (2.9) indicates in fact that they are somewhat larger. The numerical tests reported in [24] also indicate this. For large matrices, a condition number $O(h^{-2})$ means at least $O(h^{-1})$ accesses of the matrix in the iterative method (one access per iteration). This is unacceptable*.

For case (ii) the analysis cannot be performed as above for case (i), because the largest value of $\underline{x}^T A^{-1}\underline{x}/\underline{x}^T C^{-1}\underline{x}$ is close to 1 and taken for a "smooth" vector, whereas the smallest is $O(h)$, $h \to 0$ (and taken for a more "oscillating" vector). We leave the question open if the condition number in this case is improved or not by use of Neumann series.

Finally, consider case (iii). It suffices to consider the tridiagonal central difference matrix with $a_{i,i-1} = -1$, $a_{i,i} = 2$, $a_{i,i+1} = -1$, $i = 1,2,\ldots,n$. Then $\ell_{i,i-1} = 1$, $\ell_{i,i+1} = 1$ and $b_{ii} = i/(i+1)$, $i = 1,2,\ldots,n$. We get $e_{i+1,i} = (b_{i,i})^{m+1}$, $i = 1,2,\ldots,n-1$, otherwise $e_{ij} = 0$. We have $(E^TE)_{ii} = (b_{i-1,i-1})^{2(m+1)}$, $i = 2,\ldots,n$, otherwise $(E^TE)_{ij} = 0$. Note that $\|E^TE\|_2 = (1-\frac{1}{n})^{2(m+1)}$ and hence, by (2.9),

$$\underline{x}^T C^{-1}\underline{x}/\underline{x}^T Q\underline{x} \geq \frac{n}{2(m+1)}.$$

---

*) However, for the approximate block matrix factorization methods to be described in Section 4, the actual condition number for most practically sized problems may not be very large and hence the method may become competitive then.

Hence the spectral condition number of QC = QA is at least $O(h^{-1})$, but it decreases most likely approximately as m^{-1}. This indicates that it does not pay off to calculate approximations of inverses of (almost) exact factorizations of difference matrices.

3. ON POLYNOMIAL PRECONDITIONERS

For the solution of (1.1) there is a more direct way of getting a preconditioner which will only need matrix-vector multiplications, than the one described in Section 2. It is the polynomial preconditioner $P_r(A)$ of A^{-1} to be defined below.

Let P_r be a polynomial of degree r and consider the solution of

$$P_r(A)(A\underline{x}-\underline{b}) = 0 \tag{3.1}$$

by some iterative method. If we apply, say, the conjugate gradient method, we get a polynomial preconditioned method, PPCG-method. Various forms of such methods are discussed in Johnson et al. [14] and in Saad [20].

In fact, we may substitute $\tilde{A} = C^{-1}A$ for A (and $\tilde{\underline{b}} = C^{-1}\underline{b}$ for $\underline{b}$) in (3.1) to get a "doubly" preconditioned method.

For simplicity, in this section we shall assume that A (and C) are SPD.

At first we shall compare the use of the conjugate gradient method directly on $A\underline{x} = \underline{b}$ with its application on (3.1), using in both cases in total the same number of matrix-vector multiplications by A. Hence we have:

Method a: Apply k steps of CG on $A\underline{x} = \underline{b}$, where $k = (r+1)s$, s an integer. Let the resulting approximations be $x^{(k)}$.

Method b: Apply s steps of CG on (3.1). Let $\underline{y}^{(s)}$ be the resulting approximation.

It is well-known (see, for instance, [12] and [3]) that the conjugate gradient method has the following optimality property:

$$\|\underline{x} - \underline{x}^{(k)}\|_{A^{\frac{1}{2}}} = \min_{z\in K_k(\underline{r}^0,A)} \|\underline{x} - (\underline{x}^0+\underline{z})\|_{A^{\frac{1}{2}}} \tag{3.2}$$

where $K_k(\underline{r}^0,A) = \text{SPAN}\{\underline{r}^0, A\underline{r}^0, \ldots, A^k\underline{r}^0\}$ is the so-called Krylov space.

Similarly, for method b we get

$$\|\underline{x} - \underline{y}^{(s)}\|_{A^{\frac{1}{2}}} = \min_{\underline{w}\in K_s(\underline{r}^0,P_r(A)A)} \|\underline{x} - (\underline{x}^0+\underline{w})\|_{A^{\frac{1}{2}}}. \tag{3.3}$$

Since $\underline{w} \in K_s(\underline{r}^o, P_r(A)A)$ it follows that $\underline{w} = P_{s(r+1)}(A)\underline{r}^o$ for some polynomial $P_{s(r+1)}$, i.e., $\underline{w} \in K_k(r^o, A)$. Hence, because of (3.1) and (3.2),

$$\|\underline{x} - \underline{x}^{(k)}\|_{A^{\frac{1}{2}}} \leq \|\underline{x} - \underline{y}^{(s)}\|_{A^{\frac{1}{2}}}.$$

Note that for the calculation of $\underline{x}^{(k)}$ and $\underline{y}^{(s)}$ we have performed the same number of matrix vector multiplications. Hence, whatever choice of P_r we make, $x^{(k)}$ is at least as accurate as $\underline{y}^{(s)}$ and it would seem that the PPCG method can offer no advantage over the PCG method. However, the PPCG method may be more efficient when we consider the error in other norms such as the L_2-norm or the maximum norm.

Note also that there appears fewer inner products in method b than in method a. Finally, if we have an array of pipelined processors available, and if A is so large that it must be stored on peripheral storage, then method b may need much fewer accesses to A than method a. This was pointed out be Saad [20] in a recent report.

As far as the actual choice of preconditioner is concerned we can say the following. Let the eigenvalues of A be contained in the interval $0 < a \leq x \leq b$. From what has been said in Section 1 about the rate of convergence of the conjugate gradient method, it may be reasonable to let P_r be such that the spectral condition number of $P_r(A)A$ is minimized. Hence P_r should be a best approximation of the following minimization problem.
Find P_r such that

$$\min_{P_r \in \pi_r} \frac{\max_{a \leq x \leq b} |P_r(x)x|}{\min_{a \leq x \leq b} |P_r(x)x|},$$

is taken. Note that such a polynomial is unique only up to a multiplicative constant. We may normalize P_r such that, for instance,

$$\min_{a \leq x \leq b} |P_r(x)x| = 1.$$

Then we have:
Find P_r such that

$$\min_{P_r \in \pi_r} \max_{a \leq x \leq b} |P_r(x)x|$$

under the constraint $\min_{a \leq x \leq b} |P_r(x)x| = 1$.

It is easy to see that the solution of this problem can be derived from the Chebyshev approximation problem. Find P_r such that

$$\delta := \min_{\hat{P}_r \in \pi_r} \max_{a \le x \le b} |1 - \hat{P}_r(x)x|$$

is taken. The solution is as well-known

$$\hat{P}_r(x) = \frac{1}{x}\left[1 - T_{r+1}\left(\frac{b+a-2x}{b-a}\right) T_{r+1}\left(\frac{b+a}{b-a}\right)\right]. \tag{3.4}$$

Our normalization gives

$$P_r(x) = \frac{1}{1-\delta} \hat{P}_r(x).$$

Because the conjugate gradient method has the property of actually converging faster when the eigenvalues are clustered, it was pointed out in Saad [20], that letting a in (3.4) be larger than the smallest eigenvalue, actually decreased the total number of iterations. In both [14] and [20] it was further found that a least square approximation polynomial on the interval [0,b] gave a good preconditioner.

4. APPROXIMATE BLOCK-MATRIX FACTORIZATION METHOD; LEXICOGRAPHIC ORDERING

We shall consider an approximate factorization method for matrices partioned into block form that leads to highly parallel algorithms.

We limit the study to block tridiagonal matrices. For more general block matrices the method may be extended along the lines discussed in Axelsson [2]. Hence consider the matrix

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & & \bigcirc \\ A_{2,1} & A_{2,2} & A_{2,3} & \\ \cdot & \cdot & \cdot & \\ & \cdot & \cdot & \cdot \\ & & \cdot & \cdot \\ \bigcirc & & A_{n,n-1} & A_{n,n} \end{bmatrix}$$

where $A_{i,j}$ is a $m_i \times m_j$ matrix. Each block is a sparse matrix banded along a main diagonal. The diagonal block matrices are diagonally dominant and

$A_{i,j}$, $j \neq i$ are nonpositive matrices.

We consider at first the following block-matrix factorization,

$$(4.1)\qquad LU = \begin{bmatrix} X_1^{-1} & & & \bigcirc \\ A_{2,1} & X_2^{-1} & & \\ \cdot & \cdot & & \\ & \cdot & \cdot & \\ & & \cdot & \cdot \\ \bigcirc & & A_{n,n-1} & X_n^{-1} \end{bmatrix} \begin{bmatrix} I & (X_1 A_{1,2}) & & \bigcirc \\ & I & (X_2 A_{2,3}) & \\ & & \cdot & \cdot \\ & & & \cdot \\ \bigcirc & & & I \end{bmatrix}$$

$$= \begin{bmatrix} X_1^{-1} & & A_{1,2} & \bigcirc \\ A_{2,1} & (X_2^{-1}+A_{2,1}X_1A_{1,2}) & A_{2,3} & \\ \cdot & \cdot & \cdot & \\ & \cdot & \cdot & \cdot \\ & & \cdot & \cdot \\ \bigcirc & A_{n,n-1} & & (X_n^{-1}+A_{n,n-1}A_{n-1,n}) \end{bmatrix}$$

where $X_1^{-1} = A_{1,1}$, i.e., $X_1 = A_{1,1}^{-1}$, and $X_r^{-1} = A_{r,r} - A_{r,r-1}X_{r-1}A_{r-1,r}$, i.e.,

$$(4.2)\qquad X_r = (A_{r,r} - A_{r,r-1}X_{r-1}A_{r-1,r})^{-1}, \quad r = 2,3,\ldots,n.$$

Then $A = LU$.

Because A is an M-matrix, it follows that the diagonal block or in fact all main diagonal blocks are M-matrices. Also the Schur matrix compliments $A/A_{1,1} = A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}$ are M-matrices and by induction it follows in particular that X_r are positive matrices. Similar considerations are valid for the approximate factorization methods to be described later. For details, see Axelsson [2]. Hence all inverses that occur exist.

Note that the block diagonal matrices in L in (4.1) appear as inverses. This differs from the methods considered in Concus et al. [7] and Axelsson et al. [4]. Our present modification leads to highly parallel algorithms, where we have avoided solutions of all linear systems with the block diagonal matrices.

The factorization (4.1) with X_r defined by (4.2) gives a complete

factorization in which the matrices X_r are in general full matrices. This factorization would cost Σm_i^3 "flops" and the forward and backward solutions would cost $2\Sigma m_i^2$. Because of the largely increased demand of storage, this method is not practical for our purposes.

We consider now an approximate factorization method where X_r are chosen as sparse matrices, usually bandmatrices.

DEFINITION 4.1. Let G be a square matrix and let p be a nonnegative integer. Then $[G]^{(p)}$ denotes the bandmatrix of halfbandwidth p, located symmetrically about the main diagonal and whose entries within the band equal the corresponding ones in G, i.e.,

$$[G]^{(p)}_{i,j} = \begin{cases} G_{i,j}, & |i-j| \le p \\ 0, & \text{otherwise.} \end{cases}$$

Consider now the approximation of the inverse of a diagonally dominant M-matrix G with halfbandwidth p by $[G^{-1}]^{(p)}$. The following theorems shows that this may be an accurate approximation.

THEOREM 4.1. *Let* G *be a diagonally dominant* M*-matrix with real eigenvalues and smallest rowsum* $a > 0$ *and let* P *be a matrix that transforms* G *to diagonal form. Let* $b = \|G\|$. *Then the entries* $d_{i,j}$ *of the inverse* $D = G^{-1}$ *decay exponentially away from the diagonal at least as fast as* $Cr^{|i-j|/p}$, *where* $r = (1-\kappa^{-\frac{1}{2}})/(1+\kappa^{-\frac{1}{2}})$, $\kappa = b/a$ *and* $C = \max\{a^{-1}, (1+r^{\frac{1}{2}})^2/(2ar)\}$ $\|P\|\|P^{-1}\|$.

PROOF. This is a slight extension of proposition 2.2 in Demko et al. [8]. Its proof is straightforward. In Theorem 2.4 of the same paper an important extension to matrices with complex eigenvalues appears.

We shall now show how the entries of $[G^{-1}]^{(p)}$ may be calculated with maximal efficiency. In particular, it follows that we do not have to calculate any entries outside the band when G itself is a bandmatrix with halfbandwidth p.

The algorithm is based on an idea in Takahishi et al. [21], see also [9]. Here we extend this (trivially) to the case of block-diagonal matrices.

THEOREM 4.2. *Let* G *be a nonsingular matrix of the form* $G = (I-\tilde{L})B^{-1}(I-\tilde{U})$, $L = I - \tilde{L}$, $U = I - \tilde{U}$, *where* B, $\tilde{I}$ *and* $\tilde{U}$ *are given,* B *is blockdiagonal and* $\tilde{L}$, $\tilde{U}$ *are strictly lower and upper block triangular, each with block bandwidth* 2p+1.

Then

(i) $G^{-1} = BL^{-1} + \tilde{U}G^{-1}$

(ii) $G^{-1} = U^{-1}B + G^{-1}\tilde{L}$.

If the upper blocktriangular part of G^{-1} is calculated from (i) and (ii), respectively, then we may calculate any banded block-matrix part of G^{-1} with no need to calculate any entries outside this band or any entries of L^{-1} and U^{-1}.

If B is symmetric and $\tilde{U} = \tilde{L}^T$, then it suffices to use only one of (i) or (ii).

PROOF. Since $G^{-1} = (I-\tilde{U})^{-1}B(I-\tilde{L})^{-1} = (I-\tilde{U})^{-1}BL^{-1}$, we get $(I-\tilde{U})G^{-1} = BL^{-1}$, which proves (i). Similarly, (ii) follows.

Let n be the number of block rows. The block entries of G^{-1} may be calculated in the following order (note that $(BL^{-1})_{r,r} = B_{r,r}$ and that $(\tilde{U}G^{-1})_{n,n} = (G^{-1}\tilde{L})_{n,n}$ =):

(4.3)
$$\begin{cases} \text{For } r = n, n-1, \ldots, 1 \text{ do} \\ (G^{-1})_{r,r} = B_{r,r} + \Sigma_{s=1}^{\min(p,n-r)} \tilde{U}_{r,r+s}(G^{-1})_{r+s,r} \\ \text{For } k = 1,2,\ldots,p \text{ do} \\ (G^{-1})_{r-k,r} = \Sigma_{s=1}^{\min(p,n-r+k)} \tilde{U}_{r-k,r-k+s}(G^{-1})_{r-k+s,r} \\ (G^{-1})_{r,r-k} = \Sigma_{t=1}^{\min(p,n-r+k)} (G^{-1})_{r,r-k+t}\tilde{L}_{r-k+t,r-k} \end{cases}$$

For the calculation of X_r we consider at first the case of a 2D problem. Then for some $p \geq 0$, instead of (4.2) we let

(4.4)
$$\begin{aligned} X_1 &= [A_{1,1}^{-1}]^{(p)} \\ X_r &= [(A_{r,r} - A_{r,r-1}X_{r-1}A_{r-1,r})^{-1}]^{(p)}, \quad r = 2,3,\ldots,n. \end{aligned}$$

In this way all the blockmatrices X_r in (4.1) are bandmatrices with bandwidth 2p + 1. For a 2D partial differential equation, p = 1 is a natural choice, because then $A_{r,r}$ has the same bandwidth (=3). This choice was made in references [7] and [4], but with algorithms special for the

choice p = 1 and in [7] only for SPD matrices. Numerical tests indicate that p > 1 may lead to more competitive methods. Anyhow, there is clear evidence that the new block matrix algorithms are superior to the pointwise even on a sequential computer. It should be clear from what will be said about the SOLVE part of the method that they vectorize much metter.

Numerical tests also indicate that they are more robust for various scales of Reynold numbers and various flow directions in convection-diffusion problems, for instance.

Consider now a 3D problem.

Let $X_r^{(d)}$ indicate the block-diagonal matrices we get for a discretized partial differential equation in d space dimensions, and let p_1 indicate the halfbandwidth of a scalar matrix and p_2 the block halfbandwidth of a block matrix.

At each stage r of the factorization of a 3D problem we propose to construct $X_r^{(3)}$ in two steps.

Step 1: Calculate an approximate factorization of

$$H_r = A_{r,r} - A_{r,r-1} X_{r-1}^{(3)} A_{r-1,r} \tag{4.5}$$

of the form as in (4.1) but written on the form

$$G_r = (I-L\tilde{B})B^{-1}(I-B\tilde{U}) \tag{4.6}$$

where $B_{r,r} = X_r^{(2)}.X_r^{(2)}$ is calculated as in (4.4), say with $p = p_1 = 1$.

Step 2: Calculate the $p = p_2$ block-bandwidth part of G_r^{-1} by the use of algorithm (4.3).

Note that in this algorithm only matrix multiplications occur. Hence we do not need any inverses of blockmatrices. Again, the most practical choice would be $p_2 = 1$.

In this way we can calculate a sparse approximation, with sparsity structure similar to that of a 2D (difference) matrix, *of the inverse* of the block-diagonal matrices H_r in (4.5) which occur during the approximate factorization of a 3D problem.

A similar method for the approximate factorization of 3D matrix problems, nested factorization method, was used in Appleyard et al. [1]. Note, however, that there the trick of using inverses in the diagonal

matrices as in (4.1), was not used. Accordingly, there appears three levels of nested factorizations, i.e., also in line block levels, in [1].

It remains to consider the SOLVE part (forward and backward substitution) of our method. At each step of the iterative method we solve typically a linear system $LU\underline{\delta} = \underline{\rho}$ where LU is defined in (4.1). This is done in two steps.

Step 1. (forward substitution)

$$L\underline{z} = \underline{\rho} \text{ or } \underline{z}_1 = X_1\underline{\rho}_1, \quad \underline{z}_r = X_r(\underline{\rho}_r - A_{r,r-1}\underline{z}_{r-1}), \quad r = 2,3,\ldots,n.$$

Step 2. (backward substitution)

$$U\underline{\delta} = \underline{z} \text{ or } \underline{\delta}_n = \underline{z}_n, \quad \underline{\delta}_r = \underline{z}_r - X_r A_{r,r-1}\underline{\delta}_{r-1}, \quad r = n-1, \quad n-2,\ldots,1.$$

Note that we need only matrix-vector multiplications, i.e., no linear systems occur.

In addition to possible inner products and recursions associated with the particular iterative method used, the cost per iteration step is the cost for one SOLVE, i.e., about

$$2nm(2p+1) \quad \text{and} \quad 2n_1n_2m(2p+1)$$

plus the costs for $A_{r,r-1}$ x vector and $A_{r-1,r}$ x vector, for a $n \times m$ (2D) and $n_1 \times n_2 \times m$ (3D) mesh problem, respectively.

The number of iterations (with p fixed, independent of h) and $C = LU$ modified by its diagonal, so that $C\underline{e} = A\underline{e}$, $\underline{e} = (1,1,\ldots,1)^t$, will be $O(h^{-1})$ for SPD matrix problems (cf. Axelsson [2]). A proof of this will appear elsewhere.

The advantage of the method presented in this section is that there will appear no fill-in of any block-matrices in a 2D problem and if we choose $p_2 = 1$, this is also true for a 3D problem.

The disadvantage is that the factorization needs a recursive (i.e., sequential) calculation of the diagonal block matrices X_r. Hence the length of the FACTORIZATION is $O(n)$ and $O(n_1n_2)$ on a $n \times m$ and $n_1 \times n_2 \times m$ mesh, respectively.

This means in particular that if we have a structure of many *identical blocks* of order m, we do not take advantage of this fact in the above

factorization method, which is based on a lexicographic ordering of the blocks or superelements.

Note, however, that the SOLVE-part may (for p = 1) be performed in a cyclic-reduction (odd-even) fashion for bidiagonal matrices, similar to the way we shall apply it in Section 5 for tridiagonal block matrices.

In the final section we shall consider another ordering where we may calculate most of the occurring blocks in parallel and where the factorization cost decreases by a factor $[^2\log n]/n$ even on a scalar computer if the superelement blocks are all identical.

We remark finally that in Kincaid et al. [25], another form of sparse approximate inverse is proposed.

5. APPROXIMATE FACTORIZATION AND CYCLIC REDUCTION ORDERINGS

For simplicity consider a structure consisting of n superelements (not necessarily of identical size) as illustrated in Figure 5.1. (It will be clear from the presentation that we may use the method to be presented here on much more general geometries. Also the actual size of the elements may be made larger or smaller according to convenience for the computer storage actually used.)

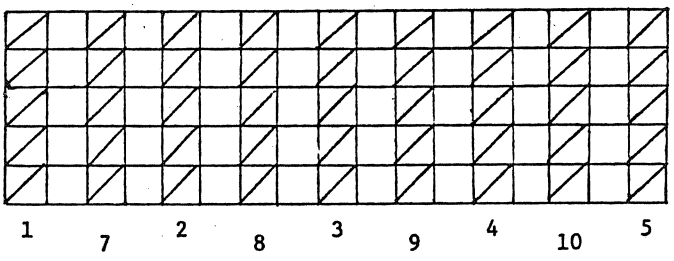
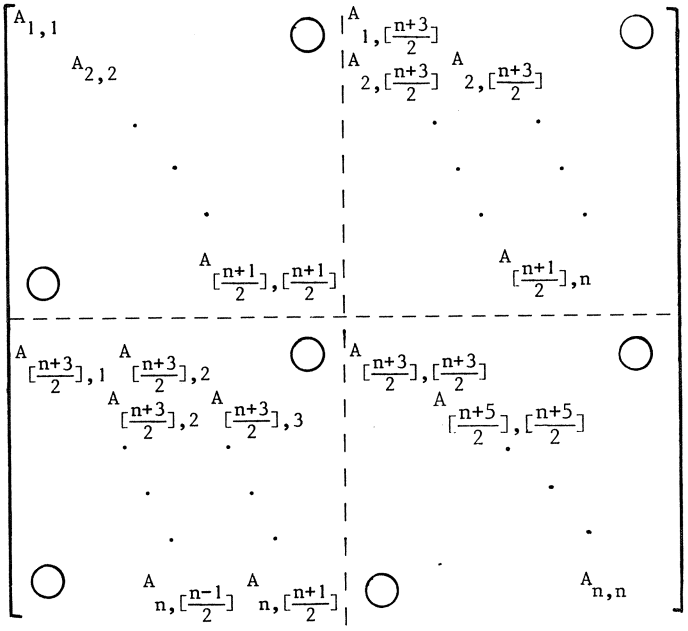


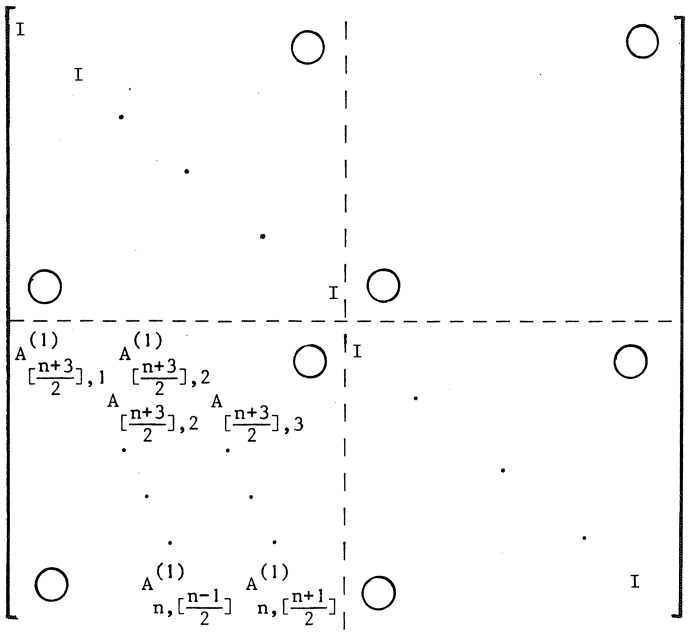
Figure 5.1. A finite element structure with odd-even ordering.

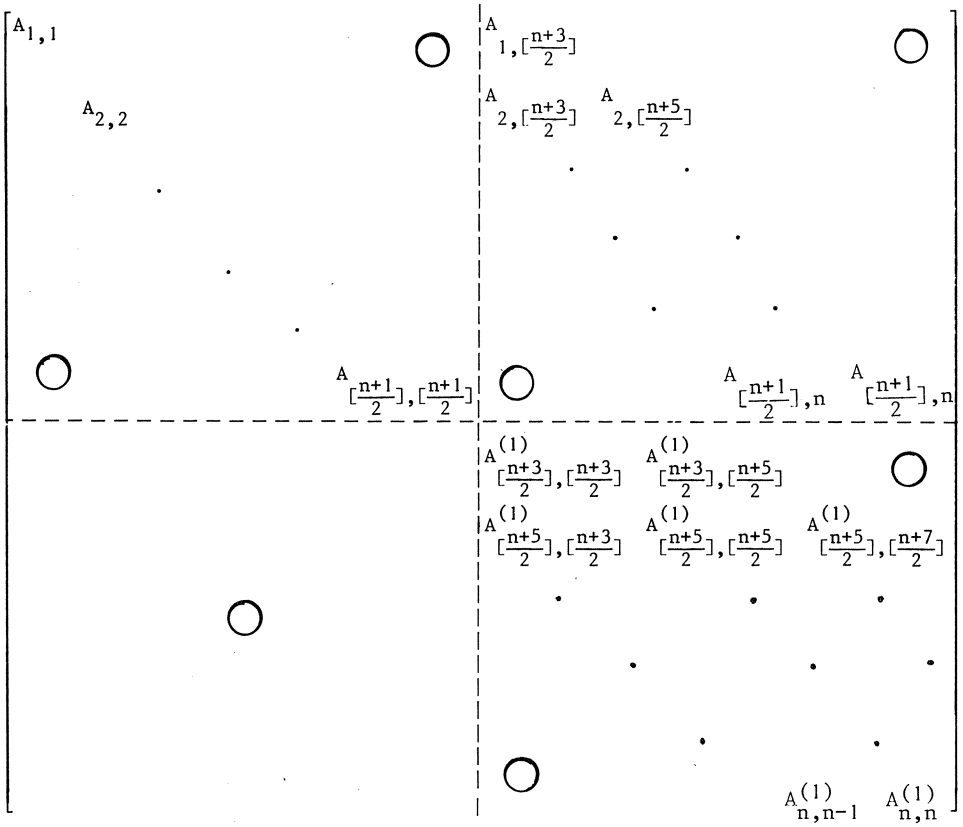
If we use an odd-even ordering of these elements as indicated in Figure 5.1, the corresponding finite element (or finite difference) matrix takes the following form. For simplicity we consider only the case where n is odd. Note that this is the same matrix that arises in a one-way dissection ordering, see George and Liu [10].

(5.1) A=



(5.2) $A = L^{(1)}U^{(1)} =$





Here

(5.3a) $$A^{(1)}_{r,j} = A_{r,j}\left(A^{(1)}_{j,j}\right)^{-1}, \quad j = r - [\tfrac{n+1}{2}], \quad r - [\tfrac{n-1}{2}], \quad r = [\tfrac{n+3}{2}],\ldots,n$$

(5.3b) $$A^{(1)}_{r,r} = A_{r,r} - A^{(1)}_{r,r-[\frac{n+1}{2}]}A_{r-[\frac{n+1}{2}],r} - A^{(1)}_{r,r-[\frac{n-1}{2}]}A_{r-[\frac{n-1}{2}],r},$$
$$r = [\tfrac{n+3}{2}],\ldots,n,$$

(5.3c) $$A^{(1)}_{r,r+1} = -A^{(1)}_{r,r-[\frac{n-1}{2}]}A_{r-[\frac{n-1}{2}],r+1}, \quad r = [\tfrac{n+3}{2}],\ldots,n-1,$$

(5.3d) $$A^{(1)}_{r-1,r} = -A^{(1)}_{r-1,r-[\frac{n-1}{2}]}A_{r-[\frac{n-1}{2}],r}, \quad r = [\tfrac{n+5}{2}],\ldots,n.$$

Note that the calculation of the block-matrices in (5.3a) may be done

in parallel. Having calculated these the blockmatrices occurring in (5.3b), (5.3c) and (5.3d) may be calculated in parallel. If the original blocks were identical along each subdiagonal within each of the four big blocks, then the same is valid for the matrices L and U. Hence, in such a case, it suffices to calculate at most five blockmatrices.

We note that a new block tridiagonal matrix has arisen in the lower big block $A^{(1)}$ of $U^{(1)}$. The number of blocks in this are only $[\frac{n-1}{2}]$. Now, *in the backward substitution process* we factorize this block in the same way as A was factorized, i.e., we use the same odd-even type of reordering of the groups of unknowns which belong to $A^{(1)}$. Hence $A^{(1)} = L^{(2)}U^{(2)}$ where $L^{(2)}$, $U^{(2)}$ have the same structure as $L^{(1)}$, $U^{(1)}$ in (5.2) and in $U^{(2)}$ we get a new big block, now of order $[\frac{n-1}{4}]$. We repeat the above recursive factorization process until eventually, a "big" block matrix has arisen with only one block. Frequently we may find it convenient to stop even earlier. The number of recursion steps, i.e., the depth of the recursion, is now at most $[^2\log n]$, compared to n for the method in Section 4. Hence, for a problem with identical substructures it suffices to calculate at most $5[^2\log n]$ block-matrices, of which only $[^2\log n]$ involve inverse matrix calculations.

The recursive application of the block L, U factorization method as described above would correspond to a cyclic reduction ordering of the original structure. In such an ordering, we take at first each second block (the odd ones), then each second of the remaining ones, etc., repeated until few or perhaps only one superelement remains. The fill-in of this method is illustrated in Figure 5.2. We prefer, however, to present the method in the above recursive way.

The forward substitution may be carried along in the same way and parallel to the calculation of the block matrices. At the final level, say s, we solve the linear system with matrix $U^{(s+1)}$. We may then calculate the other unknowns by nesting up the sequence $U^{(s)}$, $U^{(s-1)}$,...$U^{(1)}$ of U-matrices.

Alternatively, we may start from the beginning and calculate the solution of the two problems which arise when we use the final block as a separator. Then this may be repeated and at each stage we then solve for a sequence of smaller and smaller sized problems. (The size of the subproblems decrease as $n/2^r$, r = 1,2,...,3 of each stage.) The advantage with this latter approach would be that we do not need to store any of the intermediate matrices $L^{(k)}$, $U^{(k)}$.

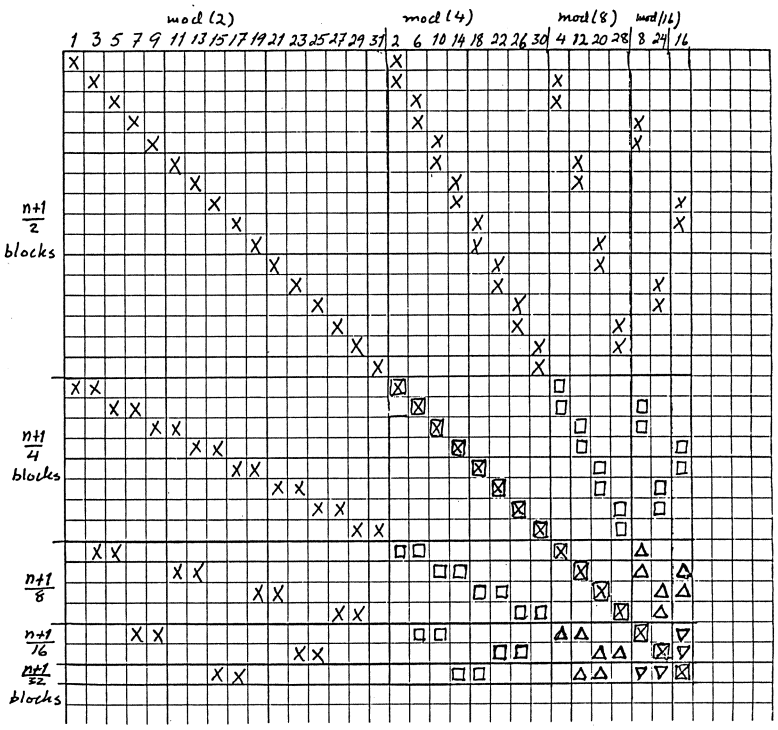


Figure 5.2. Fill-in; cyclic reduction ordering (case $n = 2^s - 1$)
□ Fill-in at stage 1, Δ Fill-in stage 2, ∇ Fill-in at stage 3.

The entries of the original matrix may be assembled and stored from the beginning, or alternatively recalculated when needed. The disadvantage with the above repeated odd-even reduction algorithm as compared to the method in Section 4, based on the lexicographic ordering, is that we get fill-in of new blocks in the two subdiagonals of the lower part of the matrices $U^{(k)}$. This disadvantage would be partly offset if we never store these matrices longer than they are needed for the calculation of the factors at the next stage, in the way as described above.

The most efficient choice of method will depend on the size and architecture of the actual computer memory to be used and on its operating system.

The above method will lead to full block matrices $A_{ij}^{(k)}$ in the same way as (4.2) did, because the inverse $(A_{ij}^{(k)})^{-1}$ are in general full. The method described above is only a modification of the method described by Heller [13]. Heller notes that the entries of the off diagonal blocks often

decay fast during the recursion and one may hence stop at an early stage.

However, more important, in order to preserve sparsity we may use the approximate inverse methods described in the previous section. Hence in (5.3) we use $(A_{jj}^{(1)})^{-1} \cong [(A_{jj}^{(1)})^{-1}]^{(p)}$ for a 2D problem and the similar approximation for a 3D problem.

The resulting approximate factorization is then used as a preconditioner for an iterative method as described earlier. Such a method was recently presented by Kershaw [15] for a 2D problem and with p = 1. See also Rodrigue and Wolitzer [18]. We shall now present a modification of that method which will enable us to extend it to 3D problems.

A New Version of the Repeated Odd-Even Reduction Ordering

We consider again a structure such as that in Figure 5.1 with its finite element or finite difference matrix $A = A^{(0)}$ partioned into block form (5.1) by the odd-even ordering. Hence

$$A^{(0)} = \begin{bmatrix} D^{(0)} & E^{(0)} \\ F^{(0)} & G^{(0)} \end{bmatrix} \tag{5.4}$$

where

$$D_{ii}^{(0)} = A_{ii}^{(0)}, \quad i = 1,2,\ldots,[\tfrac{n+1}{2}], \quad D_{ij}^{(0)} = 0, \quad i \neq j,$$

$$E_{i,i}^{(0)} = A_{i,i+[\frac{n+1}{2}]}^{(0)}, \quad i = 1,2,\ldots,[\tfrac{n-1}{2}]$$

$$E_{i,i-1}^{(0)} = A_{i,i+[\frac{n-1}{2}]}^{(0)}, \quad i = 2,\ldots,[\tfrac{n+1}{2}], \quad E_{ij}^{(0)} = 0, \text{ otherwise}$$

$$F_{i,i}^{(0)} = A_{i,i-[\frac{n+1}{2}]}^{(0)}, \quad i = [\tfrac{n+3}{2}],\ldots,n,$$

$$F_{i,i+1}^{(0)} = A_{i,i-[\frac{n-1}{2}]}^{(0)}, \quad i = [\tfrac{n+3}{2}],\ldots,n, \quad F_{ij}^{(0)} = 0, \text{ otherwise}$$

$$G_{i,i}^{(0)} = A_{i,i}^{(0)}, \quad i = [\tfrac{n+3}{2}],\ldots,n, \quad G_{ij}^{(0)} = 0, \quad i \neq j.$$

We factorize now $A^{(0)}$ as $L^{(1)}U^{(1)}$ where

$$L^{(1)} = \begin{bmatrix} X^{(1)^{-1}} & 0 \\ F^{(0)} & I \end{bmatrix}, \quad U^{(1)} = \begin{bmatrix} I & X^{(1)}E^{(0)} \\ 0 & A^{(1)} \end{bmatrix}$$

where

$$X_{i,i}^{(1)} = \left(A_{ii}^{(0)}\right)^{-1}, \quad i = 1,2,\ldots,[\tfrac{n+1}{2}], \quad X_{i,j}^{(1)} = 0, \quad i \neq j$$

and

$$A^{(1)} = G^{(0)} - F^{(0)}X^{(1)}E^{(0)}.$$

Note that $A^{(1)}$ is block tridiagonal.
For $A^{(1)}$ we use the same type of odd-even reordering. After factorization of the permuted matrix, for simplicity still denoted by $A^{(1)}$, we have $A^{(1)} = L^{(2)}U^{(2)}$, where

$$A^{(1)} = \begin{bmatrix} D^{(1)} & E^{(1)} \\ F^{(1)} & G^{(1)} \end{bmatrix}, \; L^{(2)} = \begin{bmatrix} X^{(2)^{-1}} & 0 \\ F^{(1)} & I \end{bmatrix}, \text{ and } U^{(2)} = \begin{bmatrix} I & X^{(2)}E^{(1)} \\ 0 & A^{(2)} \end{bmatrix}$$

Here $X^{(2)} = (D^{(1)})^{-1}$ and $A^{(2)} = G^{(1)} - F^{(1)}X^{(2)}E^{(1)}$. For $A^{(2)}$ we use the same process and this is repeated recursively at most $[^2\log n]$ times.

As before, in order to preserve sparsity, we approximate the inverses by

$$X_{ii}^{(k)} = \left[\left(D_{ii}^{(k-1)}\right)^{-1}\right]^{(p)} \tag{5.5}$$

for a line type matrix $D_{ii}^{(k-1)}$ and by the earlier described method for a matrix $D_{ii}^{(k-1)}$ on a 2D-type structure. The resulting matrix (which is not explicitly calculated) will then be used as a sparse preconditioner for an iterative method as described earlier. It is important to note that when the original block matrices are diagonally dominant, this property remains valid for the matrices $D_{ii}^{(k)}$, $k = 1,2,\ldots,[^2\log n]$. For a model Laplacian problem on a unit square with Dirichlet boundary conditions and discretized by the usual five point difference method, it is easy to see that the rowsums of $D_{ii}^{(k)}$ are bounded below by 2^{1-k}, $k = 0,1,\ldots,[^2\log n]$. It could be advisable to let the bandwidth p in (5.5) increase at each stage k like $p = p_k = 2^k$, say.

Note finally that parallelism in the algorithms presented in this survey may be utilized in various ways, depending on the type of computer which is available or will be available in the future.

The matrix-vector multiplications occurring for each block can typically be pipelined, i.e., performed in a similar way as on a factory assembly-line.

Note that we are thinking about problems with very large blocks. In a 3D problem each block has the same order as a 2D matrix. Hence, even on a supercomputer with a very long pipeline, this facility can be utilized efficiently.

There may alternatively be an array of identical processor units under common control available. Each of these may perform the same operation simultaneously but on different data stored in their own memories. In this case we would let each processor work on one block row in the matrix, both during factorization and during the forward and backward substitutions in the repeated odd-even reduction ordering method. The computer time will then be proportional to $[^2\log n]$ times the computer time to perform the (perhaps pipelined) matrix-vector multiplications for each row.
For a thorough presentation of various types of available computers for parallel computing, see Hockney and Jesshope [26]. For valuable comments on the effect of changing the data storage format, but not changing the basic algorithm, on the computer times on a vector computer, see Kincaid et al. [25].

REFERENCES

[1] APPLEYARD, J.R., I.M. CHESHIRE & R.K. POLLARD, *Special techniques for fully-implicit simulators*, Technical report, Computer Science and Systems Division, AERE Harwell, July 1981, to appear in Proceedings, The 1981 European Symposium On Enhanced Oil Recovery, Bournemouth, England, September, 1981.

[2] AXELSSON, O., *A general incomplete block-matrix factorization method.* Report 8337, Department of Mathematics, Catholic University, Toernooiveld, Nijmegen, The Netherlands, 1983.

[3] AXELSSON, O. & A.V. BARKER, *Finite element solutions of boundary value problems. Theory and Computation*. Academic Press, 1984.

[4] AXELSSON, O., S. BRINKKEMPER & IL'IN V.P., *On some versions of incomplete block-matrix factorization iterative methods*, to appear in Lin. Alg. Appl. 1984.

[5] AXELSSON, O. & N. MUNKSGAARD, *A class of preconditioned conjugate gradient methods for the solution of a mixed finite element discretization of the biharmonic operator*, Internat. J. Numer. Methods Engrg. 14 (1979), 1001-1019.

[6] BERMAN, A. & R.J. PLEMMONS, *Nonnegative matrices in the mathematical sciences*, Academic Press, New York, 1979.

[7] CONCUS, P., G.H. GOLUB & G. MEURANT, *Block preconditioning for the conjugate gradient method.* Report LBL-14856, Lawrence Berkeley Laboratory, University of California, 1982.

[8] DEMKO, S., W.F. MOSS & P.W. SMITH, *Decay rates for inverses of band matrices*, to appear in Math. Comp..

[9] ERISMAN, A.M. & W.F. TINNEY, *On computing certain elements of the inverse of a sparse matrix.* Comm. ACM 18 (1975), 177-179.

[10] GEORGE, A. & J.W. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, 1981.

[11] GUSTAFSSON, I.A., *Class of first order factorization methods.* BIT 18 (1978), 142-156.

[12] HAGEMAN, L.A. & D.M. YOUNG, *Applied Iterative Methods.* Academic Press, New York 1981.

[13] HELLER, DON, *Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems.* SIAM J. Numer. Anal. 13 (1976), 484-496.

[14] JOHNSON, O.G., C.A. MICCHELLI & G. PAUL, *Polynomial preconditioners for conjugate gradient calculations*, RC 9208, IBM Th. J. Watson Research Center, Yorktown Heights, New York, 1982.

[15] KERSHAW, D.S., *The solution of simple linear tridiagonal systems and vectorization of the ICCG algorithm on the CRAY-1*, pp. 85-99, in: *Parallel Computations* (G. Rodrigue, editor), Academic Press, New York, 1982.

[16] MADSEN, N.K., G.H. RODRIGUE & J.I. KARUSH, *Matrix multiplication by diagonals on a vector/parallel processor*, Inform. Proc. Letters, 5 (1976), 41-45.

[17] Meijerink, J.A. & H.A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp. 31 (1977), 148-162.

[18] RODRIGUE, G. & D. WOLITZER, *Incomplete block cyclic reduction*, 2nd IMACS International Symposium on Parallel Computation, Newark, Delaware, 1981.

[19] SAAD, Y. & M.H. SCHULTZ, *Conjugate gradient-like algorithms for solving nonsymmetric linear systems.* Research Report RR-283, Yale University, Deparment of Computer Science, 1983.

[20] SAAD, Y., *Practical use of polynomial preconditionings for the conjugate gradient method.* Research Report RR-283, Yale University, Department of Computer Science, 1983.

[21] TAKAHISHI, K., J. FAGAN & M.S. CHEN, *Formation of a sparse bus impedance matrix and its application to short circuit study,* pp. 63-69, 8th PICA Conf. Proc. 1973, Minneapolis, Minn.

[22] VARGA, R.S., *Factorization and normalized iterative methods.* In: *Boundary value problems in differential equations* (R.E. Langer, ed.), pp. 121-142. University of Wisconsin Press, Madison, Wisconsin, 1960.

[23] VARGA, R.S., *Matrix Iterative Analysis,* Prentice Hall, 1962.

[24] VAN DER VORST, H., *A vectorizable variant of some ICCG methods.* SIAM J. Sci. Stat. Comput. 3 (1982), 350-356.

[25] KINCAID, D.R., T.C. OPPE & D.M. YOUNG, *Vector computations for sparse linear systems,* Report CNA-189, Center for Numerical Analysis, The University of Texas at Austin, Austin, Texas. February, 1984.

[26] HOCKNEY, R.W. & C.R. JESSHOPE, *Parallel Computers,* Adam Hilger, Ltd., Bristol, England, 1981.

TEMPERATURE CALCULATIONS IN BUILDINGS

G.L.M. AUGENBROE

1. INTRODUCTION

In view of the growing need for energy conservation measures in buildings encountered in recent years it came as no surprise that computer programs for predicting temperatures and energy flows inside buildings were to receive considerable attention.
The need to incorporate the use of solar energy into the building has undoubtedly also caused this field to become one of growing importance. Moreover, sophisticated computer programs have enabled the development of building designs concurrently with the design of new strategies of climate control and solar energy equipment operation whereas thorough assessments of human comfort conditions inside new buildings came within reach.

Let us first give an account of the complexity of the problem at hand i.e. the phenomena that govern heat transport processes in buildings. Generally speaking a building consists of many components such as walls, floors, rooms etc. in which non stationary three-dimensional temperature fields will arise from inside and outside "loads".
Apart from exterior boundary conditions constituted by outside temperatures and solar radiation for example, we must also account for heating and cooling loads occurring inside the building, e.g. heat production from appliances, lighting, occupants or from solar radiation entering through transparent surfaces.
In actually simulating the temperature behaviour we are faced with the additional problem of adequately specifying control actions (either operated automatically or manually) aiming at optimal comfort conditions and minimal fuel consumption.

So it turns out that the problem we are dealing with is of a complex nature. Even more so if we become aware of the fact that we must solve a heat conduction problem in many solid components of irregular 3D geometry

each of which is additionally linked to other components by radiative and convective heat exchange and also solve the more difficult heat and mass transport problem inside rooms and other non-solid components. Even if we could establish exact models of every single component (which we cannot incidentally) and perform the tremendous task of assembling all of them into one very large building model it would hardly be feasible to use it in actual computations.
Also, we should be well aware of the fact that many of the phenomena cannot be described very accurately as they are liable to all kinds of unpredictable disturbances.
This is one of the main reasons that a rather extensive modelling task with the primary aim at simplifying our problem should be performed, as will be shown in section 2.

With the previous in mind it is not hard to guess that many different computer programs for this field of problems exist. They differ in the user groups they are intended for, the modelling approaches taken and the solution techniques employed. The intended user-friendliness (i.e. low level of user-contribution to modelling tasks) of the majority of these programs restricts its use to rather simple pre-defined classes of problems.
In section 3 we will explore the capabilities of a computer program contrarily designed for almost unlimited use, i.e. allowing the user to perform his own modelling.
Some of the mathematical backgrounds of this finite element-based package are elucidated in section 4.
Section 5 will introduce two examples from its field of application. In section 6 we will, apart from drawing some conclusions, give an outlook onto future developments.

2. MODELLING PROCESS

In order to render the temperature calculation problem in buildings solvable with reasonable efforts we are obliged to introduce several simplifying assumptions with regard to the afore stated general problem. One should realize of course that requirements on accuracy, being generally quite modest, often enable one to choose intuitively based, sometimes rather crude models.

Some of the standard modelling approaches will now be presented.

Standard models.

Fig. 2.1(a) shows several building components with dimensions inherent in the standard problem. Obviously heat transport can be expected to occur mainly in the directions perpendicular to the walls as shown in fig. 2.1(b).

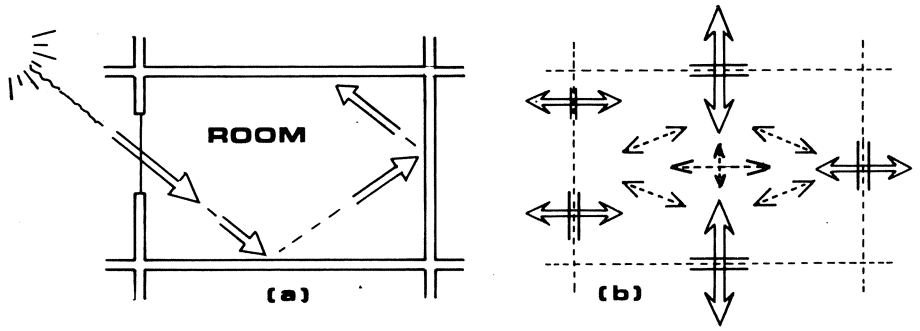


Fig. 2.1. Standard modelling of several buildings components

It is now assumed that no lateral heat transport in the walls occurs, implying that

- heat conduction to connecting walls is neglected
- radiative heat exchange (q_{rad}) with other components is evenly distributed along the surface
- the same assumption holds for the convective heat exchange (q_{conv}) with the surrounding air:

$$(2.1) \qquad c\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(\lambda\frac{\partial u}{\partial x}\right) = 0$$

with boundary condition at a surface (i):

$$(2.2) \qquad -\lambda\left(\frac{\partial u}{\partial n}\right)^i = q^i_{conv} + q^i_{rad} + q^i_{load}$$

where we have introduced:

$u(x,t)$ = wall temperature
x = coordinate perpendicular to wall surface
t = time
$c(x,u)$ = heat capacity of wall material
$\lambda(x,u)$ = heat conductivity of wall material
n = outward normal to wall surface

Assuming the room air to have uniform temperature u_r along with a convective heat exchange coefficient α_{conv} with which q_{conv} can be characterized it follows that:

$$q^i_{conv} = \alpha^i_{conv}(u_i - u_r). \tag{2.3}$$

It should be noticed that the uniform air temperature assumption by no means gives a very accurate picture of the actual temperature distribution. Due to stratification effects significant vertical gradients can occur. Some work is being done to develop more refined models in which heat and mass flow across the room are better accounted for. q_{rad} can be expressed by a summation over all exchanges between surface i and all other room-facing surfaces (j):

$$q^i_{rad} = \sum_j q^i_{rad,j} \tag{2.4}$$

where

$$q^i_{rad,j} = \psi_{i,j}\sigma(u_i^4 - u_j^4) \tag{2.5}$$

σ = Boltzmann constant for "gray" body radiation
$\psi_{i,j}$ = "exchange factor" between surfaces i and j

The 4th order terms in the radiation exchange can usually without much loss of accuracy be linearized to

$$q^i_{rad,j} = \alpha_{si,j}\psi_{i,j}(u_i - u_j) \tag{2.6}$$

$\alpha_{si,j}$ = linearization factor.

For the computation of $\psi_{i,j}$, which requires a matrix-inversion in case of reflective surfaces, we refer to [1].

In (2.2) q_{load} represents an extra boundary heat flux originating from incident solar radiation or other sources.

Writing (2.1)-(2.5) for all walls and adding equation (2.7), expressing the room heat balance:

$$(2.7) \qquad cap_r \frac{\partial u_r}{\partial t} = \sum_j A_j q^j_{conv} + Q_r$$

where cap_r = room heat capacity
A_j = area of surface j
Q_r = all extraneous sources to the room

we arrive at a set of coupled p.d.e. that can be solved numerically. Before thus proceeding it is still needed to complete the model by specifying additional information of which we mention:

- $q^i_{load}(t)$
  Accurate determination of solar irradiation on external and internal surfaces is obviously one of the key operations for accurate simulation results. In case of surrounding buildings even the calculation of irradiances on external surfaces can be quite difficult.
- $\alpha^i_{conv}(u_i,u_r)$
  The temperature difference-dependency of α^i_{conv} in case of natural convection should be accounted for along internal surfaces
- $\alpha^i_{conv}(v_{wind})$
  The wind speed dependency of α^i_{conv} in case of forced convection should be accounted for along external surfaces.
- $Q_r(t,u_r)$
  Specification of heat supply from different sources:
  - lighting, persons, appliances
  - heating and cooling equipment
  - heat supply caused by solar radiation intercepted by internal components e.g. furniture that are not modelled separately but attributed to the room heat capacity cap_r.

  REMARK. It is obvious that an internal description of the heat supply equipment, i.e. handling it as an added component is quite often indispensable. It is however usually not regarded as part of the standard modelling approach.

- Controls regarding the operation of supply equipment (Q_r), shading devices (Q_r,q^i_{load}), occupant behaviour (Q_r) and the like.
- Climatic data.

The foregoing offers a fairly complete view of the standard modelling approach usually applied to the problem of temperature calculations in buildings.
Most of the existing computer packages use it explicitly, although they may differ on the following points:

- employment of further simplifications eventually leading to the simplest of equations, solvable with the aid of a hand calculator.
- discretization of the differential equations, for example by finite differences, finite elements or others.
- solution method for solving the resulting equations.

Most of these computer packages suffer however from a severe lack of flexibility when it comes to problems that require extended modelling facilities, i.e. containing non-standard components.
Unfortunately this happens to be the case in most of the solar energy designs.
Among these non-standard components we mention gravel beds, solar collectors, phase change storage components.
Also, when wanting to specify detailed control strategies, time-dependencies of system parameters or other out of the ordinary information, one is usually at a loss.
The next section will introduce a computer program designed to alleviate most of the drawbacks.

3. BFEP COMPUTER PROGRAM

BFEP is a finite element-based computer program intended for the calculation of temperatures in buildings.
Its philosophy was adopted from AFEP [2], a sophisticated general-purpose finite element program. Like AFEP, BFEP consists of a (yet much smaller) library of FORTRAN-coded subroutines [3]. Due to its modular approach, the user can define any load, climate, control, algorithm etc. in a user-written main program and user-subroutines. Alternatively he can simply select standard options by supplying appropriate input data. The actual computation stage is preceded by a separate preparation stage, the latter lending itself to interactive data generation.
So two separate stages are distinguished:

- Modelling stage:
  the physical problem is schematized and translated into a mathematical

model to be enhanced by physical data.
All modelling is based on the finite element technique, whether for standard space discretization or by specifying "fictive" elements for defining lumped heat exchange equations between distinctive nodes. BFEP offers the possibility of computer aided development of the input data by activating so-called building model data generation subroutines that actually constitute the major part of the FORTRAN-source. Non-standard components can be supplied by specifying appropriate (user-)elements.
For very complex components a slightly different approach should preferably be taken. A separate investigation at the single component-level using AFEP is then performed first.
The results are consequently used to obtain a fairly simple model, preferably "1-dimensional" and containing as few nodes as possible, depending on the accuracy required.
In some cases one must take recourse to an alternative approach in trying to establish simple models from physical experiments.
Alternatively, if so required by accuracy-demands one might still want to add the full-sized component model to the building model.

- Computation stage:
  the global equations are built from the input model; the first step of the semi-discrete finite element technique will lead to the global system.

$$(3.1) \qquad M\,\underline{\dot{u}} + S\underline{u} = \underline{q}$$

with initial condition

$$(3.2) \qquad \underline{u}(0) = \underline{u}_0.$$

The set of o.d.e's (3.1) must consequently be solved by an appropriate time integration method.
Meanwhile other problem-defining data can be transferred through the user-supplied software.
For the tedious task of calculating q_{load} BFEP makes use of the computer program SIBE [4] which uses a fast and unique method to calculate solar loads on internal and external surfaces in any built environment.

It should be obvious that BFEP tries to impose as few restrictions as possible on the creative (modelling) actions, whereas routine actions are performed by the computer.

Needless to say that its use is limited to a group of users with sufficient know-how and experience in the application field thus prohibiting black box use by someone unaware of the limitations of the underlying models, as indeed any program should.

4. MATHEMATICAL BACKGROUNDS OF BFEP

As mentioned earlier, BFEP is based on finite element space discretization. It's element-coding was adopted from AFEP in order to guarantee compatibility of both programs on element level.
There are however some severe restrictions on BFEP which were introduced in order to retain simplicity and to keep the source volume to a minimum:

- -every node must contain only one unknown
- -in principle only 1D-elements can be used (BFEP does not support 2D or 3D mesh generators).

It was explained in the previous section that this does not give rise to strong limitations on the applicability of BFEP because most standard components can be regarded as essentially 1D whereas others can be represented by lumped models, usually requiring only 1D-elements for their representation.
Future developments are aimed at a component-based approach (CFEP) in which full use of AFEP on component-level should become possible.

Let us now proceed to showing the general features of the semi-discrete finite element technique employed throughout.
Fig. 4.1 shows two solid bodies surrounded by a fluid or gas (assumed to be transparent, e.g. air)

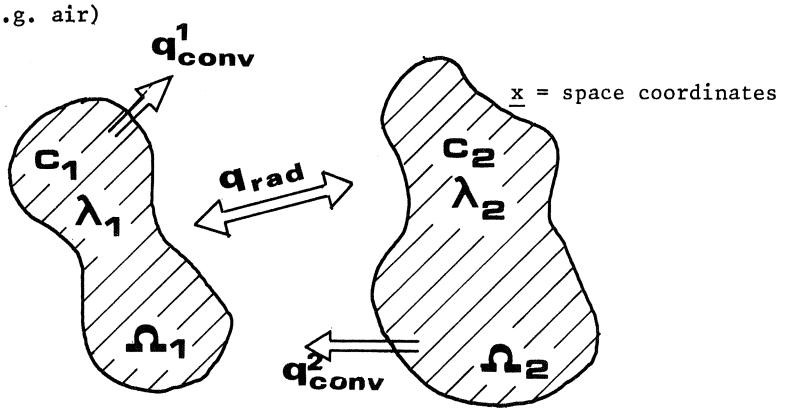


Fig. 4.1. Heat exchanges of two solid bodies surrounded by a fluid

Adopting Fourier's law for heat conduction we can state the energy balances inside the solid bodies as follows:

$$(4.1) \qquad c_1 \frac{\partial u_1}{\partial t} - \nabla\lambda_1 \nabla u_1 = q_{s,1} \qquad \underline{x} \in \Omega_1$$

$$(4.2) \qquad c_2 \frac{\partial u_2}{\partial t} - \nabla\lambda_2 \nabla u_2 = q_{s,2} \qquad \underline{x} \in \Omega_2 .$$

A heat source-term q_s has been added for reasons of generality. Disregarding the presence of boundary conditions for the moment one can apply a space discretization inside Ω_1 and Ω_2 choosing suitable elements to cover these areas.

Denoting by $\underline{u}_1(t)$ and $\underline{u}_2(t)$ the vectors containing the discrete form of the unknown temperatures $u_1(\underline{x},t)$ and $u_2(\underline{x},t)$, (4.1) and (4.2) can be put into its customary discretized form:

$$(4.3) \qquad \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \begin{bmatrix} \dot{\underline{u}}_1 \\ \dot{\underline{u}}_2 \end{bmatrix} + \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \end{bmatrix} = \begin{bmatrix} \underline{q}_1 \\ \underline{q}_2 \end{bmatrix} .$$

The mass matrices M_1, M_2 and the stiffness matrices S_1, S_2 are built by the usual element-assembly process, extending over all elements (e):

$$(4.4) \qquad M = \sum_{(e)} M^e; \quad S = \sum_{(e)} S^e .$$

The load vectors $\underline{q}_1$, $\underline{q}_2$ are formed likewise by assembling all element load vectors.

Assuming the fluid temperature u_f unknown but uniform the number of unknowns is increased by one.

The incorporation of boundary conditions concerning convective and radiative heat exchange into (4.3) can be accomplished by the addition of several special-purpose elements that are taken along in the assembly process, eventually resulting in:

$$(4.5) \qquad \begin{bmatrix} M_1 & 0 & \underline{0} \\ 0 & M_2 & \underline{0} \\ \underline{0}^T & \underline{0}^T & cap_f \end{bmatrix} \begin{bmatrix} \dot{\underline{u}}_1 \\ \dot{\underline{u}}_2 \\ \dot{u}_f \end{bmatrix} + \begin{bmatrix} S_1+A & -B & -\underline{d} \\ -B^T & S_2+C & -\underline{e} \\ -\underline{d}^T & -\underline{e}^T & s_f \end{bmatrix} \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ u_f \end{bmatrix} = \begin{bmatrix} \underline{q}_1 \\ \underline{q}_2 \\ q_f \end{bmatrix} .$$

(4.5) contains the assembled element matrices A, B, C that result from the

radiative interrelatedness of the solid components as well as the element vectors $\underline{d}$, $\underline{e}$ resulting from the convective exchange to the surrounding fluid.
The added equation containing the total fluid heat capacity cap_f ensures the heat balance of the fluid, in which a heat source q_f has been assumed. Obviously A, B, C, $\underline{d}$ and $\underline{e}$ pertain only to boundary nodes.
(4.5) can be restated to form the global system of ordinary differential equations:

$$M\dot{\underline{u}} + S\underline{u} = \underline{q}. \tag{4.6}$$

In this way any set of components can be represented by a global set of o.d.e's (4.6), any algebraic equations can be added by setting the corresponding part of M identically zero.

A few words might be in order to stipulate the fact that the matrices A, B and C are built in the customary way, by which we mean that they are taken along in the assembly.
Introducing R for that part of S that accounts for radiative exchange, we can write

$$R = \begin{bmatrix} \tilde{A} & -B \\ -B^T & \tilde{C} \end{bmatrix}. \tag{4.7}$$

Note: $\tilde{A}$ and $\tilde{C}$ differ from A and C in that they do not "contain" convective exchange between component 1 and the fluid.

We have $R = \Sigma_{(e)} R^e$, summing over all pairs of edge elements, as shown in fig. 4.2 for the 2D-case (assuming linear conforming triangular elements).

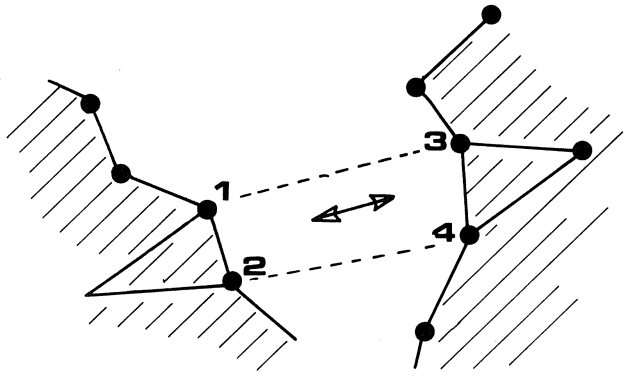
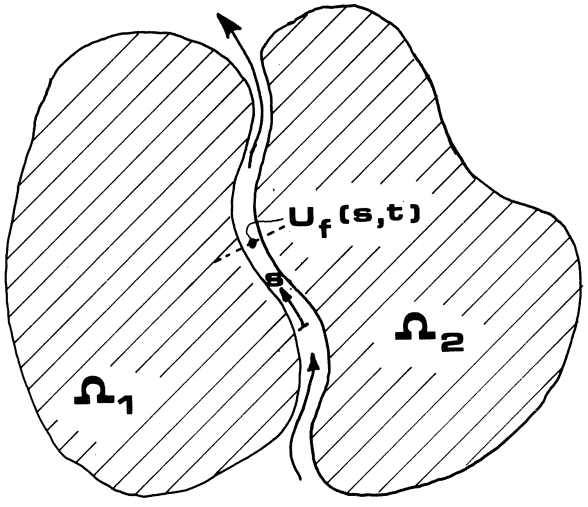


Fig. 4.2. Radiation exchange between two edge elements

It is advantageous to build R by a single line element subroutine. The line element must contain all nodes on the boundary of the enclosure. If we also add the fluid temperature-node to the line element we can build the discrete equations for all boundary conditions as well as the heat balance equation for the fluid inside the enclosure into one single element stiffness matrix S^e:

$$S^e = \begin{bmatrix} A & -B & -\underline{d} \\ -B^T & C & -\underline{e} \\ -\underline{d}^T & -\underline{e}^T & s_f \end{bmatrix} . \tag{4.8}$$

The analysis becomes somewhat more elaborate if the assumption of one single fluid temperature is no longer valid.
For instance, when dealing with a known channel flow field as in fig. 4.3 where we have to introduce an unknown function $u_f(s,t)$, representing the cross sectional average of the fluid temperture.



c_f = heat capacity of the fluid
λ_f = heat conductivity of the fluid
$\bar{v}$ = average velocity in s-direction

Fig. 4.3. Channel flow between two solid bodies

The energy balance for the fluid is expressed by

$$c_f \frac{\partial u_f}{\partial t} - \lambda_f \frac{\partial^2 u_f}{\partial s^2} + \bar{v}\, c_f \frac{\partial u}{\partial s} = \alpha_{conv} \cdot \frac{S}{A} (\bar{u}_w - u_f) + q_f \tag{4.9}$$

A = cross sectional channel area

S = boundary area of channel cross section

$\bar{u}_w$ = average temperature on channel boundary

q_f = heat source in the fluid

$\alpha_{conv}(\bar{v})$ = convective heat exchange coefficient

Using (4.9) we can write an extended version of (4.5):

$$(4.10) \qquad \begin{bmatrix} M_1 & 0 & 0 \\ 0 & M_2 & 0 \\ 0 & 0 & M_f \end{bmatrix} \begin{bmatrix} \dot{\underline{u}}_1 \\ \dot{\underline{u}}_2 \\ \dot{\underline{u}}_f \end{bmatrix} + \begin{bmatrix} S_1+A & -B & -D \\ -B^T & S_2+C & -E \\ -D^T & -E^T & S_f \end{bmatrix} \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ \underline{u}_f \end{bmatrix} = \begin{bmatrix} \underline{q}_1 \\ \underline{q}_2 \\ \underline{q}_f \end{bmatrix} .$$

$\underline{u}_f$ must be taken to be any suitable discrete form of u_f. The matrices E, D and S_f are rather straightforward discretizations of (4.9), S_f being non-symmetric due to the first derivative-occurance in (4.9). As further details of how all matrices are actually filled can be found elsewhere we will refrain from going into these matters here.

Most important to be noticed is the fact that we again arrive at the global set of o.d.e. (4.6).

Use of BFEP will consequently always comprise the following four actions:

1. Read model input (topological and physical data)
2. Build global matrices M and S by element assembly
3. Set initial conditions $\underline{u}_0$
4. Solve (4.6) by a time-marching scheme.

During step 4 user-written subroutines will be called to perform the following tasks:

- updating of $S(\underline{u},t)$
- updating of $\underline{q}(t)$, i.e. computing solar node loads with the help of SIBE and computing node loads due to other sources
- specifying control actions

For the solution of (4.6) several well-known methods can be chosen. A standard method frequently used is a P-C method with enlarged stability-region, as described in [5]. More efficient methods such as those described in [7] can also be used.

Rewriting (4.6) is a different way, introducing $\underline{f}$:

$$(4.11) \qquad \dot{\underline{u}} = \underline{f}(\underline{u},t),$$

the method from [5] can be written in the form:

$$\underline{k}_1 = h\underline{f}(\underline{u}_n, t_n)$$

$$\underline{k}_2 = hf(\underline{u}_n + \underline{k}_1, t_n + h) \tag{4.12}$$

$$\underline{u}_{n+1} = \underline{u}_n + \frac{1}{8}(7\underline{k}_1 + \underline{k}_2)$$

in which

$$\underline{u}_{n+1} \approx \underline{u}(t_{n+1})$$

$$h = t_{n+1} - t_n.$$

The explicitness of the method above along with its enlarged stability region usually guarantees reasonable efficiency.
Due to great differences in heat capacities of components the set of o.d.e. (4.6) can be extremely stiff.
The standard approach is then to perform an implicit computation on a subset of the equations of (4.6).
For instance in case of air flowing between massive solid components it is usually quite acceptable to set M_f in (4.10) identically zero.
In doing so we are left with a subset of algebraic equations that needs implicit solving in each step of the P-C algorithm.
BFEP provides in subroutines by which this can be greatly facilitated.

5. APPLICATIONS

A short account will be given of two research projects for which BFEP served as basic research tool.
The first is from the field of solar energy use in dwellings whilst the second concerns a novel way of reducing energy consumption in office buildings.
In both cases we are dealing with non-standard components and advanced control measures that require a sophisticated computer program for prediction and simulation purposes.
As both applications have been reported on elsewhere only a few features will be highlighted.

Both projects are carried out under grants from the Projectbureau for Energy Research (PBE) within the framework of the Dutch National Program

for Energy Research.

Solar Cavity Houses [6]

In so called solar cavity houses, only recently introduced by Kristinsson, heat is distributed from an air-based solar collector to the living quarters. Fig. 5.1. shows the recirculation pattern of the solar-heated air. The solar collector is directly connected to the crawl space and a cavity wall separating two neighbouring houses. The collected heat is stored in the internal building mass and will eventually become available to the living zone.

In one of the houses that were built in Leiderdorp, temperatures and heat flows are monitored by an on-line datalogging system for a period of two years.

The present study is focussing on adjusting the model input to BFEP in accordance with the gathered data and using it to investigate overall performance, comfort conditions and make an assessment of economic efficiency.

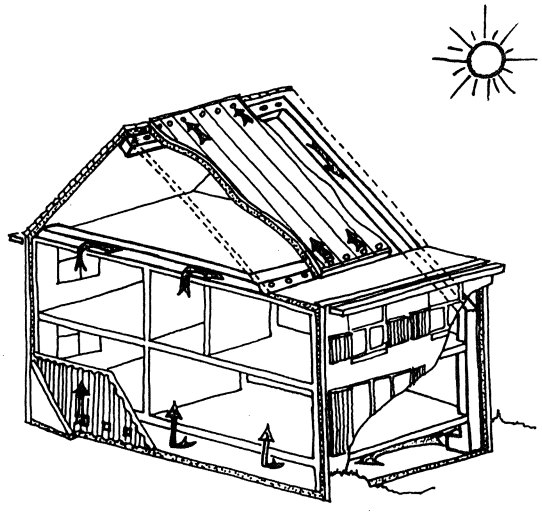


Fig. 5.1. Recirculation pattern of solar heated air

"Energon"-equipped office buildings

Fig. 5.2. shows a concrete hollow core floor slab through which the supply air is transported before it enters the rooms.
This way of using internal storage capacity has been termed Energon-principle by the firm Schokbeton which introduced it in the Netherlands. The principle aims at exploiting the internal mass of the building structure for storage of excess heat in winter and available night cooling in summer.
The sizing and control strategy in order to obtain maximum energy savings are matters to be investigated for every single application.
The first stage of the research project comprised the development of a simple lumped model representing the floor slabs. The second stage which is presently being carried out concerns a parameter study into the bearings of several building parameters on Energon-efficiencies.
During the third phase a recently constructed office building will be monitored for a period of approximately two years.

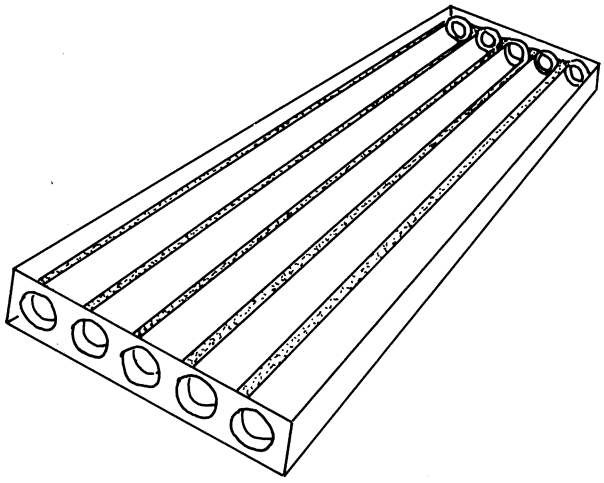


Fig. 5.2. Hollow core slab.

6. CONCLUDING REMARKS

The way in which most existing computer programs handle temperature calculation problems in buildings has been shown to be rather restrictive. The philosophy adopted by BFEP enables extended modelling which is an essential feature in the presence of non-standard components

The efforts for preparing the model input are kept to a minimum by supplying so called building data generation subroutines.
The finite element-based approach guarantees furthermore great flexibility in specifying other problem defining data and controls.

All users of BFEP can join in a BFEP user group, presently consisting of both government agencies and commercial firms.

Future developments will be directed towards a fully component-based version along with graphical enhancement of a preprocessing stage in which input for BFEP and SIBE is jointly composed. The related software developement will be aimed at foreground preprocessing on micro computers.

REFERENCES

[1] AUGENBROE, G.L.M., *Finite elements in Building Physics*, Report P7801, Building Physics Group, Dept. of Civil Engineering, Delft University of Technology, 1978.

[2] SEGAL, A., *AFEP User Manual*, Dept. of Mathematics and Informatics, Delft University of Technology, 1977.

[3] AUGENBROE, G.L.M., *BFEP User Manual*, Building Physics Group, Dept. of Civil Engineering, Delft University of Technology, 1982 (in Dutch).

[4] VOORDEN, M. van der, *Direct solar irradiation on given areas in the built environment*, calculated with the universal computer program SIBE1, (to be published).

[5] PRAAGMAN, N. & A. SEGAL, *The finite element method for time-dependent problems*, Delft Progress Report 2, 119-130, 1977.

[6] AUGENBROE, G.L.M. & E.H. TUMBUAN, *Performance characteristics of an air-based heating system connected to a solar collector*, Solar World Congress, Perth, 1983.

[7] HOUWEN, P.J. van der & B.P. SOMMEIJER, *On the internal stability of explicit, m-stage Runga-Kutta methods for large m-values*, ZAMM 60, 479-485, 1980.

NUMERICAL SOLUTION OF A ONE-DIMENSIONAL STEFAN PROBLEM ARISING FROM LASER-ANNEALING

M. BAKKER

1. INTRODUCTION

The problem described here was presented to the author by Frans Saris from Amolf and Wang Zhong Lie (Peking University, from September 1980 to September 1982 working at Amolf). It arose from the research of Laser-annealing of *Si* or *GaAs* whose crystalline structure had been destroyed near the surface by ion-implantation [4,5]. It appears that the crystalline structure recovers after the amorphous layer is irradiated by a laser-beam and is heated to the point of melting. We consider a small piece of *Si* which consists of a thin amorphous layer ($1.5_{10}-5$ cm) at the surface and a relatively thick crystalline layer ($1.185_{10}-3$ cm). When the surface is irradiated by a laser beam of (Gaussian) intensity

$$I(t) = I_0 e^{-(\frac{2\pi I_0(t-t_0)}{E_0})^2} \tag{1.1}$$

with

I_0 maximum intensity in Watt/cm^2
t_0 time at which intensity reaches its peak
E_0 laser energy in Joule/cm^2

the temperature of the *Si* rises and after some time (relatively spoken, the process described takes place within tens of nanoseconds), melting sets in. The melt front (see fig. 1) moves from left to right, until the heating effect due to the laser pulse $I(t)$ dies down and resolidification starts. Then the melt front (it should now be called resolidification front, but for convenience, we will always speak about melt front) moves from right to left until *Si* is solid again.

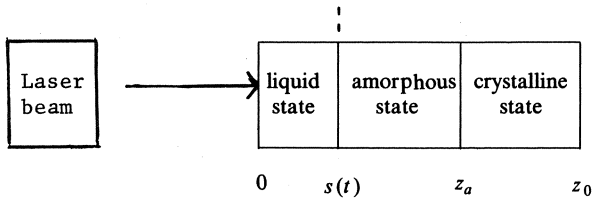


Figure 1.

1.1. Mathematical description of the process

In mathematical terms, the variation of the temperature is described by the heat equation

$$c(T)\rho\frac{\partial T}{\partial t} = \frac{\partial}{\partial z}(\kappa(T)\frac{\partial T}{\partial z}) + S_L(z,t);\ 0<z<z_0;\ t\geqslant 0;$$

$$S_L(z,t) = \alpha(z)e^{-\alpha(z)z}I(t)(1-R); \tag{1.2a}$$

where the parameters have the following meaning:

T	temperature in K;
t	time in seconds;
z	distance from irradiation surface in cm;
z_0	thickness of irradiated *Si*;
$c(T)$	heat capacity of *Si* in Joule/(gram Kelvin); c(T) $>$ 0;
ρ	mass density of *Si* in gram/cm^3 ;
$\kappa(T)$	heat conductivity of *Si* in Watt/(cm Kelvin); $\kappa(T) > 0$;
$\alpha(z)$	absorption coefficient of *Si* in cm $^{-1}$; $\alpha(z) > 0$;
$I(t)$	intensity of laser pulse, defined by (1.1) ;
R	reflectivity of surface; depends on state of surface;
	$R = R_1$,if no melting occurs;
	$R = R_2$,if melting occurs;
	$R_1 > R_2$;

The boundary and initial conditions of T are

$$\begin{aligned} &\frac{\partial T}{\partial z}(0,t) = 0; \\ &T(z_0,t) = T_0\ (= 300K); \\ &T(z,0) = T_0; \end{aligned} \tag{1.2b}$$

where T_0 is the environment temperature.

The two-phase state

If *Si* is only in solid state, equation (1.2a) is valid on $(0,z_0)$ and $\kappa(T)$ and $\partial T / \partial z$ are continuous on $(0,z_0)$, while the forcing term $S_L(z,t)$ may be discontinuous at some points, e.g. at the amorphous-crystalline interface. If melting occurs, however, these expressions are discontinuous at the melt front $s(t)$, where the Stefan interface conditions hold:

$$T(s(t),t) = T_m\ (= 1685K); \tag{1.2c}$$

$$L_0\rho\dot{s}(t) = \lim_{z\downarrow s(t)}[\kappa(T)\frac{\partial T}{\partial z}] - \lim_{z\uparrow s(t)}[\kappa(T)\frac{\partial T}{\partial z}] \tag{1.2d}$$

with

L_0	latent heat of *Si* in J/g;
$\dot{s}(t)$	$ds(t)/dt$, the speed of the melt front;
T_m	melt temperature of *Si*;

Discontinuities of κ(T) and α(z) at melt front

In the current model, the absorption coefficient α is supposed to be a piecewise constant function defined by

$$\alpha(z) = \begin{cases} \alpha_s \ (=5.0_{10}4) & , \text{if } s(t) < z \leqslant z_0; \\ \alpha_m \ (=7.0_{10}5) & , \text{if } 0 \leqslant z < s(t) \end{cases} \tag{1.3}$$

The heat conductivity is defined by

$$\kappa(T) = \begin{cases} \kappa_m(=0.51) & , \text{if } T > T_m; \\ \kappa_s(T) & , \text{if } T_0 \leqslant T < T_m; \end{cases} \tag{1.4}$$

where κ_s is positive , continuous and monotonically decreasing on $[T_0,T_m]$ with

$$\lim_{T \uparrow T_m} \kappa_s(T) \neq \kappa_m.$$

1.2. Limitations of the model

The above model of the heat conduction by laser-iradiated *Si* is a deliberate simplification, since only the mathematical aspects are now being discussed. The simplifications are:

1. Evaporation of *Si* is not taken into account, while as a matter of fact, it evaporates if E_0 is large enough.
2. α_s is assumed to be constant on $(s(t),z_0)$, while in reality α_s has a discontinuity at the crystalline-amorphous interface.
3. The melt temperature of *Si* is assumed to be the same for amorphous and crystalline *Si*, while the amorphous melt temperature is some hundreds of degrees lower than the crystalline melt temperature (1373 K vs. 1685 ° K).
4. $\kappa_s(T)$ is discontinuous at the amorphous-crystalline interface, while in this paper it is supposed to be continuous on $[s(t),z_0]$.

2. NUMERICAL SOLUTION OF THE MOVING BOUNDARY PROBLEM

There are some pitfalls for the numerical analyst, when he tries to solve this problem per computer:

1. The appearance and disappearance of the melt front.

   Before and after the melting, the Boundary Problem is standard. It is, however, impossible to establish beforehand when the melting commences and terminates. Consequently, after each timestep, the state of the system has to be checked:

   a. Is the temperature at the surface (where it is hottest) still below T_m ? (only relevant, if the temperature is rising and if there is no liquid state).

   b. Is the melt front still present?

2. The presence of a moving material interface inside $(0,z_0)$ makes it impossible to use standard semi-discretization and time-integration methods.

   The use of a variable space-grid which was successfully practiced by Bonerot & Jamet [2], fails here because of the initial and final tininess of the liquid region $(0,s(t))$.

2.1. Transformation of the space variable

A well known method for coping with MBPs is the use of transformations replacing variable domains by fixed domains. In this case, we introduce the transformations

$$z = \begin{cases} xs(t) & , \text{if } z \leqslant s(t); \\ xz_0 + (1-x)s(t) & , \text{if } z \geqslant s(t). \end{cases} \tag{2.1}$$

Furthermore, we split T in T_s and T_l by

$$T = \begin{cases} T_l & , \text{if } z \leqslant s(t); \\ T_s & , \text{if } z \geqslant s(t). \end{cases} \tag{2.2}$$

After some calculus, problem (1.2) is transformed to the following system of problems:

$$\begin{aligned} &c(T_l)\rho[\frac{\partial T_l}{\partial t} - \frac{x\dot{s}}{s}\frac{\partial T_l}{\partial x}] = \frac{\kappa_m}{s^2}\frac{\partial^2 T_l}{\partial x^2} + \\ &\quad + \alpha_m(1-R)I(t)e^{-\alpha_m xs}; \\ &\frac{\partial T_l}{\partial x}(0,t) = 0;\ T_l(1,t) = T_m; \end{aligned} \tag{2.3}$$

$$\begin{aligned} &c(T_s)\rho[\frac{\partial T_s}{\partial t} - \frac{(1-x)\dot{s}}{z_0-s}\frac{\partial T_s}{\partial x}] = \frac{1}{(z_0-s)^2}\frac{\partial}{\partial x}(\kappa_s(T_s)\frac{\partial T_s}{\partial x}) + \\ &\quad + \alpha_s(1-R)I(t)e^{-\alpha_s(xz_0+(1-x)s)}; \\ &T_s(0,t) = T_m;\ T_s(1,t) = T_0; \end{aligned} \tag{2.4}$$

$$L_0\rho\dot{s} = \frac{\kappa_s(T_m)}{z_0-s}\frac{\partial T_s}{\partial x}(0,t) - \frac{\kappa_m}{s}\frac{\partial T_l}{\partial x}(1,t). \tag{2.5}$$

We see that by the use of (2.1), we have transformed (1.2) into a system of two transient boundary problems and one *Ordinary Differential Equation* (ODE).

2.2. The discrete time Galerkin method

If s(t) were always positive, problems (2.3)-(2.5) could be semi-discretized in space by some standard method and afterwards integrated in time by some ODE integrator. However, s(t) starts and ends with a zero value which makes (2.3) singular. We therefore introduce a *discrete time method:* we approximate $\partial T_l / \partial t$ and $\partial T_s / \partial t$ by some difference method and hence solve the resulting system of ODEs.

For the time-discretization, we choose the *Backward Euler Method:*

$$\frac{\partial T_l}{\partial t}(x,t) \approx \frac{T_l(x,t) - T_l(x,t-\tau)}{\tau};\ \frac{\partial T_s}{\partial t}(x,t) \approx \frac{T_s(x,t) - T_s(x,t-\tau)}{\tau}; \tag{2.6}$$

Substitution of (2.6) in (2.3) and (2.4) leads to a system of *Two Point Boundary Problems*

$$c(T_l)\rho[\frac{T_l-T_l^*}{\tau} - \frac{x\dot{s}}{s}T_l'] =$$

$$= \frac{\kappa_m}{s^2}T_l'' + \alpha_{in}(1-R)I(t)e^{-\alpha_m xs}; \tag{2.7}$$

$$T_l'(0) = 0;\ T_l(1) = T_m;$$

$$c(T_s)\rho[\frac{T_s-T_s^*}{\tau} - \frac{(1-x)\dot{s}}{z_0-s}T_s'] =$$

$$= \frac{1}{(z_0-s)^2}(\kappa_s(T_s)T_s')' + \alpha_s(1-R)I(t)e^{-\alpha_s[xz_0+(1-x)s]}; \tag{2.8}$$

$$T_s(0) = T_m;\ T_s(1) = T_0;$$

$$L_0\rho\dot{s} = \frac{\kappa_s(T_m)}{z_0-s}T_s'(0) - \frac{\kappa_m}{s}T_l'(1). \tag{2.9}$$

In (2.7) and (2.8), T_l, T_s, T_l^* and T_s^*, denote $T_l(x,t)$, $T_s(x,t)$, $T_l(x,t-\tau)$ and $T_s(x,t-\tau)$, respectively.

The problems (2.7)-(2.8) can be solved by some space discretization method. This method can be the Finite Difference Method (FDM) or the Finite Element Method (FEM) [6] . We prefer the latter method, not solely for reasons of taste but because it enables us to approximate $\dot{s}$ more accurately, as we will show.

2.2.1. The Finite Element Method using piecewise linear functions

Let

$$\Delta = \{0=x_0 < x_1 < \cdots < x_N = 1\} \tag{2.10}$$

be a uniform partition of [0,1] in N segments with

$$x_i = hi;\ i = 0,\cdots,N;\ h = 1/N; \tag{2.11}$$

If we put

$$T_l(x_i,t) \approx U_i;\ T_l(x_i,t-\tau) \approx U_i^*;$$

$$T_s(x_i,t) \approx V_i;\ T_s(x_i,t-\tau) \approx V_i^*;\ i = 0,\cdots,N. \tag{2.12}$$

Then, for $\vec{U}$ and $\vec{V}$, the following difference scheme results from application of the C^0 Galerkin method using piecewise linear functions on Δ:

$$c(U_0)\rho\frac{h}{2}[(U_0-U_0^*)/\tau] = \frac{\kappa_m}{s^2}\frac{U_1-U_0}{h} + \frac{h}{2}\alpha_m I(t)(1-R);$$

$$c(U_i)h\rho[(U_i-U_i^*)/\tau - \frac{x_i\dot{s}}{s}\frac{U_{i+1}-U_{i-1}}{2h}] =$$

$$= \frac{\kappa_m}{s^2}\frac{U_{i+1}-2U_i+U_{i-1}}{h} + h\alpha_m I(t)(1-R)e^{-\alpha_m s x_i};\ 1 \leqslant i \leqslant N-1; \tag{2.13}$$

$$U_N = T_m;$$

$$c(V_i)\rho h\left[\frac{V_i-V_i^*}{\tau} - \frac{(1-x_i)\dot{s}}{z_0-s}\frac{V_{i+1}-V_{i-1}}{2h}\right] =$$

$$= \frac{(\kappa_s(V_{i-1})+\kappa_s(V_i))(V_{i-1}-V_i) + (\kappa_s(V_{i+1})+\kappa_s(V_i))(V_{i+1}-V_i)}{2(z_0-s)^2 h} + \tag{2.14a}$$

$$+ h(1-R)\alpha_s I(t)e^{-\alpha_s[x_i z_0+(1-x_i)s]};\ 1 \leqslant i \leqslant N-1;$$

$$V_0 = T_m;\ V_N = T_0.$$

If no melting occurs and if we apply the same discrete time Galerkin method, we obtain for $\vec{V}$ the linear system

$$c(V_0)\rho h\frac{V_0-V_0^*}{2\tau} = \frac{(\kappa_s(V_0)+\kappa_s(V_1))(V_1-V_0)}{2z_0^2 h} +$$

$$+ \tfrac{1}{2}h(1-R)\alpha_s I(t);$$

$$c(V_i)\rho h\left[\frac{V_i-V_i^*}{\tau} =\right.$$

$$= \frac{[\kappa_s(V_{i-1})+\kappa_s(V_i)](V_{i-1}-V_i) + [\kappa_s(V_{i+1})+\kappa_s(V_i)](V_{i+1}-V_i)}{2z_0^2 h} + \tag{2.14b}$$

$$+ h(1-R)\alpha_s I(t)e^{-\alpha_s x_i z_0};\ 1 \leqslant i \leqslant N-1;$$

$$V_N = T_0.$$

Note that in (2.14b) V_0 is now variable, because of the natural boundary condition (1.2b) for $z=0$.

2.2.2. Approximation of $\dot{s}$

In (2.13)-(2.14) $\dot{s}$ has yet to be approximated by some difference expression. It would be tempting to use some finite difference formula for $\dot{s}$ like

$$L_0\rho\dot{s} \approx \frac{\kappa_s(V_0)(V_1-V_0) - \kappa_m(U_N-U_{N-1})}{h}. \tag{2.15}$$

This formula, however, has the disadvantage that the approximation of $\dot{s}$ is one order less accurate than the approximation of T_s and T_l, i.e. of $O(h)$ vs. $O(h^2)$.

In order to obtain a better approximation of $\dot{s}$, we use a result of Wheeler [3] which consists of a cheap and accurate approximation of the flux at the break-points x_i, once the Galerkin solution is known (see the Appendix for a more extensive treatment). If we apply their theorem to (2.13) and (2.14), we obtain the following approximations for $T_l'(1,t)$ and $T_s'(0,t)$:

$$\frac{\kappa_m}{s}T_l'(1) \approx \frac{\kappa_m(U_N-U_{N-1})}{hs} + \tag{2.16}$$

$$+ \tfrac{1}{2}[\alpha_m hsI(t)(1-R)e^{-\alpha_m s} - \dot{s}c(U_N)\rho(U_N-U_{N-1})];$$

$$\frac{\kappa_s(T_m)}{z_0-s}T_s'(0,t) \approx \frac{[\kappa_s(V_0)+\kappa_s(V_1)](V_1-V_0)}{2h(z_0-s)} + \tag{2.17}$$

$$+ \tfrac{1}{2}[\dot{s}c(V_0)\rho(V_0-V_1) + h(z_0-s)\alpha_s I(t)(1-R)e^{-\alpha_s s}];$$

If we subtract (2.16) from (2.17), we easily find that

$$\dot{s}[L_0\rho + \tfrac{1}{2}c(T_m)(U_{N-1}-V_1)] \approx$$

$$\approx \frac{[\kappa_s(V_0)+\kappa_s(V_1)](V_1-V_0)}{2h(z_0-s)} - \frac{\kappa_m(U_N-U_{N-1})}{hs} + \tag{2.18}$$

$$+ \tfrac{1}{2}hI(t)(1-R)[\alpha_m s e^{-\alpha_m s} + \alpha_s(z_0-s)e^{-\alpha_s s}]$$

which yields an approximation of $\dot{s}$ which is not only more accurate than (2.15) but works better at the early stage of the melting state.

2.3. Iteration scheme for time-integration

As we saw in the previous section, time-integration is easy enough, if $s(t)=0$. If melting occurs, we use for the solution of (2.13)-(2.14) and (2.18) the following scheme (see also Bonerot & Jamet [2]) :

<table>
<tr><td colspan="2">Make an initial guess of s by
$s_0(t)=s(t-\tau)+\tau\dot{s}_0(t)$; $\dot{s}_0(t)=\dot{s}(t-\tau)$
Make an initial guess of $\vec{U}$ and $\vec{V}$ by putting $\vec{U}^0 = \vec{U}^*$ and $\vec{V}_0 = \vec{V}^*$</td></tr>
<tr><td rowspan="2">Repeat until some stop criterion is satisfied</td><td>Perform one Newton iteration for the solution of (2.13)-(2.14) for $s = s_i(t)$, $\dot{s} = \dot{s}_i(t)$, to obtain $\vec{U}^{i+1}$ and $\vec{V}^{i+1}$, $i = 0, \cdots,$</td></tr>
<tr><td>Compute $\dot{s}_{i+1}(t)$ from (2.18) for $\vec{U}^{i+1}$ and $\vec{V}^{i+1}$ and put $s_{i+1}(t)=s(t-\tau)+\tau\dot{s}_{i+1}(t)$</td></tr>
</table>

Table 1. Iteration scheme for time-integration

We see that the above iteration scheme reduces to a common Newton iteration process, if we delete the updating of $\vec{U}$, $s(t)$ and $\dot{s}(t)$ from it.

There is only one small problem in the iteration scheme of table 1: how to get a reasonable guess of $\dot{s}$ when, the melting starts? One could try to extrapolate formula (2.18) to $s=0$. Another method is the following: One can easily derive from (1.2) the ODE

$$\begin{aligned}
&\frac{d}{dt}[L_0\rho s(t) + \int_0^{z_0} T(z,t)dz] = \\
&= \int_0^{z_0} S_L(z,t)dz + \kappa_s(T_0)\frac{\partial T}{\partial z}(z_0,t) = \\
&= (1-R)I(t)(1-e^{-\alpha_m s}+e^{-\alpha_s s}-e^{-\alpha_s z_0}) + \kappa_s(T_0)\frac{\partial T}{\partial z}(z_0,t).
\end{aligned} \tag{2.19}$$

If we cancel $\frac{\partial T}{\partial z}(z_0,t)$ in (2.19) and cancel the variation of $\int_0^{z_0} T(z,t)dz$, we have the approximation

$$L_0\rho\dot{s}(t) \approx (1-R)I(t)(1-e^{-\alpha_s z_0}), \text{ if } s(t) = 0. \tag{2.20}$$

REMARK

Formula (2.20) is an overestimation of $\dot{s}$ but has the advantage that its order of magnitude is not wildly misguessed.

2.4. Stepsize control

In order to control the time-integration process, the following conditions were imposed to the stepsize τ :

1. The surface temperature T(0,t) is not allowed to change by more than, say, 100 K. If so, the time-step is rejected and, by interpolation, a new and smaller time-step is tried.
2. The melt front is not allowed to change by more than , say, 50 Angstroem. If so, the time-step is rejected and, by interpolation, a new and smaller time-step is tried.
3. τ has the bounds

$$1.0_{10}-12 = \tau_{\min} \leqslant \tau \leqslant \tau_{\max} = 10.0(hz_0)^2$$

4. τ is not allowed to increase by more than 50 % per timestep.
5. If the state of the system changes between $t-\tau$ and t, t is rejected and , by interpolation, replaced by a new t which is a prediction of the time at which the change takes place. For example, if

   $V_0^* < T_m < V_0$;

   where $\vec{V}$ is the solution of (2.14b), then τ is replaced by

$$\tau^* = \frac{(T_m - V_0^*)\tau}{V_0 - V_0^*}.$$

3. NUMERICAL EXAMPLE

In this chapter, problem (1.2) is numerically solved for the following input parameters:

Parameter	value or domain
T_0	300 K
T_m	1685 K
t_0	25 *ns*
I_0	$5.0_{10}7$
z_0	$1.2_{10}-3$ cm
ρ	2.33
E_0	0.5,1,1.5,2,2.5 Joule
κ_m	0.51
α_m	$7.0_{10}5$
α_s	$5.0_{10}4$
L_0	1801
κ_s	$1.38 \geqslant \kappa_s \geqslant 0.32$
	κ_s monotonically decreasing on (T_0,T_m)
c(T)	$0.95 \leqslant c(T) \leqslant 1.10$
	c monotonically increasing on $[T_0,\infty)$
R	R = 0.3, if no melting occurs;
	R = 0.6, if *Si* is melting

From the graphs of $s(t;E_0)$ (see fig. 2) and $T(0,t;E_0)$ (see fig. 3) one can see that

1. The melting starts sooner, the melt depth is larger and the melting ends later, when E_0 is larger;
2. The surface temperature has a larger peak, if E_0 is larger.

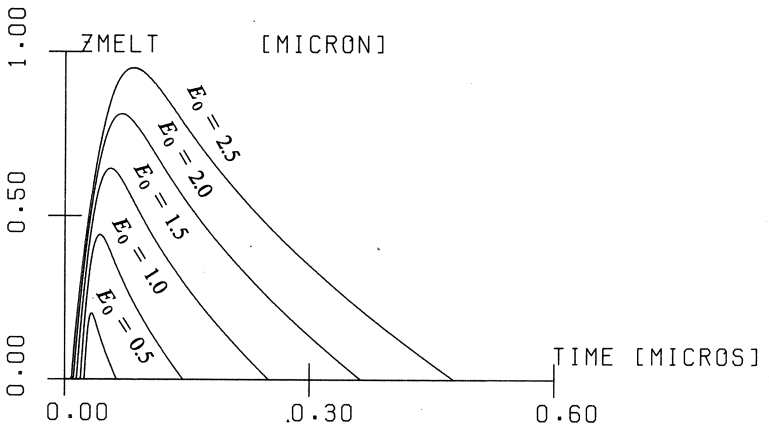


Figure 2. Graph of $s(t)$ for $E_0 = 0.5,1.0,1.5,2.0,2.5$

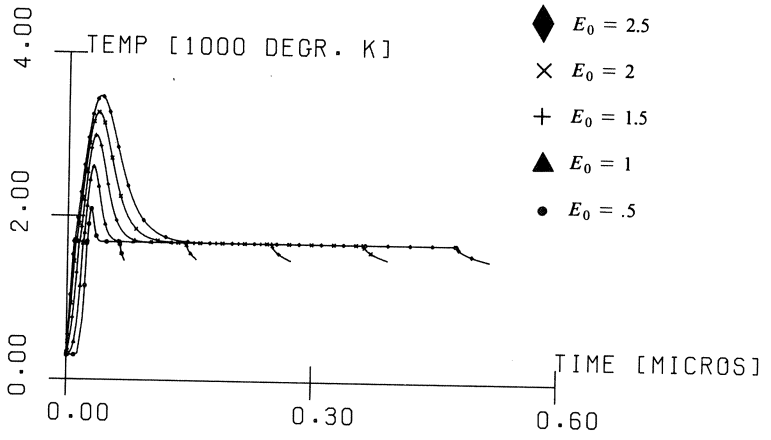


Figure 3. Graph of $T(0,t)$ for $E_0 = 0.5,1.0,1.5,1.0,1.5$

REFERENCES

[1] ABRAMOWITZ, M. & I. STEGUN, *Handbook of Mathematical Functions,* Dover Publications, 1964.

[2] BONEROT, R. & P. JAMET, *A Second Order Finite Element Method for the One-Dimensional Stefan Problem,* Internat. J. Numer. Methods Engrg. **17** (1981), 811-820;

[3] CAREY, G.F., D. HUMPHREY & M.F. WHEELER, *Galerkin and Collocation-Galerkin Methods with Superconvergence and Optimal Fluxes,* Internat. J. Numer. Methods Engrg. **8** (1974), 939-950;

[4] GILES, G.E.& R.F. WOOD, *Macroscopic Theory of Pulsed Laser Annealing,* Phys. Rev B **3** (1981), 2923-2942;

[5] DE JONG, T., WANG Z.L. & F.W. SARIS, *An experimental test of GaAs decompostion due to pulsed laser irradiation,* to appear in Physical Letters

[6] MITCHELL, A.R. & R. WAIT, *The Finite Element Method in Partial Differential Equations,* John Wiley & Sons, Chichester, New York, Brisbane, Toronto, 1977;

APPENDIX

Let

$$-(p(x)y')' + q(x)y' + r(x)y = f(x),\ x \in (0,1); \tag{A1}$$

$$y(0) = y(1) = 0; \tag{A2}$$

be some two-point boundary problem with unique solution. Then for every $\phi \in C^0(0,1)$ the relation

$$(py',\phi') + (qy',\phi) + (ry,\phi) = (f,\phi) + [p(x)y'(x)\phi(x)]_0^1 \tag{A3}$$

holds, where (.,.) denotes the usual $L^2(I)$ inner product.

Let Δ defined by (2.10)-(2.11) , be a partition of [0,1] and let ϕ_i, $i = 0, \cdots, N$ be defined by

$$\phi_0(x) = \begin{cases} 1-x/h & \text{, if } 0 \leqslant x \leqslant h; \\ 0 & \text{, } elsewhere; \end{cases} \tag{A4}$$

$$\phi_i(x) = \begin{cases} (x-x_{i-1})/h & \text{, if } x_{i-1} \leqslant x \leqslant x_i; \\ (x_{i+1}-x)/h & \text{, if } x_i \leqslant x \leqslant x_{i+1}; \\ 0 & \text{, } elsewhere; i = 1, \cdots, N-1; \end{cases} \tag{A5}$$

$$\phi_N(x) = \begin{cases} (x-_{N-1})/h & \text{, if } x_{N-1} \leqslant x \leqslant 1; \\ 0 & \text{, } elsewhere; \end{cases} \tag{A6}$$

Then it is standard [6] that y(x) can be approximated by

$$Y(x) = \sum_{i=1}^{N-1} c_i \phi_i(x); \tag{A7}$$

where $\vec{c}$ is the solution of the (tri-diagonal) linear system

$$\sum_{j=1}^{N-1} [(p\phi'_i,\phi'_j) + (q\phi_i,\phi'_j) + (r\phi_i,\phi_j)]c_j = (f,\phi_i),\ i = 1,\cdots,N-1. \tag{A8}$$

The pointwise error of Y is

$$|y(x) - Y(x)| \leqslant C(y)h^2,\ x \in [0,1]; \tag{A9}$$

For the boundary fluxes, Wheeler[3] developed the approximations

$$\begin{aligned} -p(0)y'(0) &\approx (pY',\phi'_0) + (qY',\phi_0) + (rY,\phi_0) - (f,\phi_0); \\ +p(1)y'(1) &\approx (pY',\phi'_N) + (qY',\phi_N) + (rY,\phi_N) - (f,\phi_N); \end{aligned} \tag{A10}$$

by simply applying (A3) for ϕ_0 and ϕ_N and replacing y by Y in the integrand. She could prove that (A10) has an approximation error of $0(h^2)$ instead of $O(h)$ which would have been achieved if the difference formulae

$$\begin{aligned} y'(0) &\approx (c_1 - c_0)/h; \\ y'(1) &\approx (c_N - c_{N-1})/h \end{aligned} \tag{A11}$$

were used.

In (A9-A10), integrals involving p,q,r,f are to be evaluated, which can sometimes be cumbersome. However, if in formulae (A9-A10) (.,.) is approximated by the extended trapezoid rule [1,ch. 25.4.2], the orders of accuracy remain $O(h^2)$.

If the above formulae are applied to (2.7)-(2.9) with use of the extended trapezoidal formula, equations (2.13)-(2.14) and (2.16)-(2.18) result.

ITERATIVE COMPUTATIONAL TECHNIQUES FOR SOLVING INTEGRAL EQUATIONS

P.M. van den BERG

In this paper we discuss some iterative techniques for solving integral equations, using an error criterion. Specifically, we consider the integral equations arising from direct scattering theory. We investigate the implications of the integrated square error criterion and the minimization of this error is taken as a condition for getting the best result. After discretization, the numerical implication of the least-square-error criterion leads to the numerical solution of a system of linear algebraic equations. Usually, this system is inverted by a direct method.

In the case, however, that we are dealing with large systems of equations, the problem of excessive computer time and computer storage required for direct numerical solution of systems of equations can be circumvented using a suitable iterative technique. Another argument to solve the pertinent system of equations iteratively, is the evident fact that we should not solve the relevant systems of equations with a higher degree of accuracy than the one imposed by our error criterion.

In the present paper we discuss alternative implementations of some iterative techniques, in which the intermediate step of reduction of the problems to a system of linear algebraic equations is superfluous. The integrated square error is taken as a measure of the approximate solution from the exact one. Starting with an arbitrary initial guess and a set of arbitrarily chosen correction functions, a convergent iteration scheme is to be developed. Some suitably chosen choices for the correction functions are discussed. Some numerical results to a number of representative problems illustrate the rate of convergence of the different methods.

1. INTRODUCTION

During the past several years considerable effort has been put into the development of computational techniques for handling the scattering and diffraction of waves by an obstacle. From a mathematical point of view, uniqueness of the solution of a scattering problem is guaranteed if the pertinent wave equations, the boundary conditions (for impenetrable objects) or constitutive equations (for penetrable objects) and the causality condition are satisfied. Exact satisfaction of these conditions can be arrived only at by analytical procedures (which exist for simple geometries only). The integral-equation techniques now guarantee that the wave equations and the causality condition are satisfied. This leaves the boundary conditions of the constitutive relations to be satisfied in a computational manner and calls for an error criterion that tells us to which degree of accuracy we have proceeded when we terminate after some number of steps.

2. INTEGRAL EQUATIONS AND THE ROOT MEAN SQUARE ERROR

The integral equations that arise from the application, in direct scattering theory, of the contrast-source type integral representation of the scattered field, either in the frequency domain or in the time domain, are all of the form

$$(1) \qquad g(x) = \int_{x' \in \mathcal{D}} K(x,x') f(x')\, dx', \quad \text{when } x \in \mathcal{D}.$$

In this equation, f is the unknown field quantity in the relevant, spatial or space-time, scattering domain, g is a known field related to the excitation of the scatterer by known sources, and K is the kernel of the integral equation, which is related to the field at x radiated by a contrast source at x'. In general, x and x' stand for the relevant coordinates (for example, the Cartesian coordinates x, y, z in three-dimensional space, and the time coordinate t), f and g are vector valued, K yields the proper matrix or tensor relationship, and $\mathcal{D}$ is the domain in which the equality sign in (1) holds.

In almost all situations met in practice, (1) can only be solved approximately with the aid of numerical techniques. How well an approximate solution is, can only be quantified after having defined a quantitive error.

In order to discuss this, we introduce an operator formalism and an inner product of two functions defined in $\mathcal{D}$ (with the associated norm). To write (1) in an operator form, we define the operator K acting on a function f by

$$(2) \qquad Kf = \int_{x' \in \mathcal{D}} K(x,x') f(x')\, dx'.$$

Then, (1) is equivalent to

$$(3) \qquad g = Kf \quad \text{when } x \in \mathcal{D}.$$

Further, we introduce as the inner product of two functions f and g defined on $\mathcal{D}$ the, real or complex, number

$$(4) \qquad \langle f , g \rangle = \int_{x \in \mathcal{D}} f(x) g(x)\, dx = \langle g , f \rangle$$

while the norm of a function f is defined as

$$(5) \qquad ||f|| = \langle f^* , f \rangle^{\frac{1}{2}}$$

Let now for any approximate solution f^A, the root mean square error be defined as

$$(6) \qquad ERR = \langle (g - Kf^A)^* , g - Kf^A \rangle^{\frac{1}{2}},$$

being the norm of $g - Kf^A$. We note that $ERR \geq 0$. Only if $f^A = f$, the equality sign holds. In the next section, we shall show, how in an iterative way we can minimize ERR in (6) and enforce its convergence to zero. We finally mention that in scattering problems, where g is related to the exciting field in the domain $\mathcal{D}$, we often normalise the error to the root-mean-square value of the exciting field. We then have the normalised error

$$(7) \qquad \overline{\overline{ERR}} = \frac{\langle (g - Kf^A)^* , g - Kf^A \rangle^{\frac{1}{2}}}{\langle g^* , g \rangle^{\frac{1}{2}}} .$$

3. ITERATIVE MINIMIZATION OF THE ERROR

In this section we outline an iterative minimization of the root mean square error which leads to the solution of the integral equation. We assume the existence of an iterative procedure, in which n steps have been

carried out, and that this has led to the values $f^{(n)}$. The root mean square error $ERR^{(n)}$ after n steps of iteration is

$$(8) \qquad ERR^{(n)} = \langle r^{(n)*}, r^{(n)} \rangle^{\frac{1}{2}},$$

in which the residual $r^{(n)} = r^{(n)}(x)$ is given by

$$(9) \qquad r^{(n)} = g - Kf^{(n)}.$$

In going from the $(n-1)$-st step to the n-th, we take

$$(10) \qquad f^{(n)} = f^{(n-1)} + \eta^{(n)} \phi^{(n)},$$

where $\eta^{(n)}$ is a variational parameter and $\phi^{(n)} = \phi^{(n)}(x)$ is a suitably chosen variational function (how they actually are constructed will be discussed later). Using Eqs. (9) and (10), the residual error becomes

$$(11) \qquad r^{(n)} = r^{(n-1)} - \eta^{(n)} K\phi^{(n)}.$$

The expression for $[ERR^{(n)}]^2$ can be written as

$$(12) \qquad [ERR^{(n)}]^2 = [ERR^{(n-1)}]^2 - 2\,\mathrm{Re}[\eta^{(n)} A^{(n)}] + |\eta^{(n)}|^2 B^{(n)},$$

in which

$$(13) \qquad A^{(n)} = \langle r^{(n-1)*}, K\phi^{(n)} \rangle,$$

$$(14) \qquad B^{(n)} = \langle K^* \phi^{(n)*}, K\phi^{(n)} \rangle.$$

Equation (12) can be rewritten as

$$(15) \qquad [ERR^{(n)}]^2 = [ERR^{(n-1)}]^2 - \frac{|A^{(n)}|^2}{B^{(n)}} + B^{(n)} \left| \eta^{(n)} - \frac{A^{(n)*}}{B^{(n)}} \right|^2.$$

The right-hand side of (15) has, as a function of $\eta^{(n)}$, a minimum at

$$(16) \qquad \eta^{(n)} = \frac{A^{(n)*}}{B^{(n)}}.$$

Substitution of this value in (15) leads to

$$(17) \qquad [ERR^{(n)}]^2 = [ERR^{(n-1)}]^2 - \frac{|A^{(n)}|^2}{B^{(n)}},$$

from which it follows that, if $A^{(n)} \neq 0$, an improvement in the satisfaction of the integral equation is arrived at. This latter condition puts some restriction on the choice of the variational function $\phi^{(n)}$. The complete iteration scheme for a given sequence of variational functions $\phi^{(n)}$ can be found in Table I. If the values of $K\phi^{(n)}$ in each step of iteration can be stored, a substantial reduction of computing labour has been achieved. The iteration scheme can be terminated as soon as a sufficiently small error has been achieved.

Table I. The iteration scheme for a given sequence of variational functions $\phi^{(n)}$.

$n = 0$

Initial estimate : $f^{(0)}$

Residual error : $r^{(0)} = g - Kf^{(0)}$

$ERR^{(0)} = \langle r^{(0)*} , r^{(0)} \rangle^{\frac{1}{2}}$

$n = n + 1$

Variational function: $\phi^{(n)}$

$A^{(n)} = \langle r^{(n-1)*} , K\phi^{(n)} \rangle$

$B^{(n)} = \langle K^*\phi^{(n)*} , K\phi^{(n)} \rangle$

$\eta^{(n)} = \frac{A^{(n)*}}{B^{(n)}}$

storage: $f^{(n)}$, $r^{(n)}$

Improved estimate : $f^{(n)} = f^{(n-1)} + \eta^{(n)} \phi^{(n)}$

Residual error : $r^{(n)} = r^{(n-1)} - \eta^{(n)} K\phi^{(n)}$

$ERR^{(n)} = \langle r^{(n)*} , r^{(n)} \rangle^{\frac{1}{2}}$

Substitution of Eq. (16) in (11), and using the definitions (13) and (14), it follows that our minimization procedure has led to

$$(18) \qquad \langle r^{(n)*} , K\phi^{(n)} \rangle = 0,$$

an orthogonality property on $\mathcal{D}$ that will be of later use. For later purposes we also want to bring out the dependence of $A^{(n)}$ on $\phi^{(n)}$. To this end, we interchange the integrations of the inner product definition and the operator definition. We arrive at

$$(19) \qquad A^{(n)} = \langle K^T r^{(n-1)*} , \phi^{(n)} \rangle,$$

where the transposed operator K^T acting on a function f is defined by

$$(20) \qquad K^T f = \int_{x' \in \mathcal{D}} K(x',x) f(x') \, dx'.$$

In case K is a matrix, we should also take the transposed matrix form of K under the integral sign. Similarly, Eq. (18) can be rewritten as

$$(21) \qquad \langle K^T r^{(n)*} , \phi^{(n)} \rangle = 0.$$

In subsequent sections, some particular choices of $\phi^{(n)}$, which up to now have been completely arbitrary, will be discussed.

4. A SECOND MINIMIZATION STEP

In this section we will investigate how, during the n-th iteration, we can decrease the right-hand side of Eq. (17) still further by manipulating the variational function $\phi^{(n)}$. This will be done in such a manner that $B^{(n)}$ is minimized, while keeping $A^{(n)}$ constant. In view of Eq. (21), with n replaced by $n-1$, the value of $A^{(n)}$ remains unchanged if, in the right-hand side of Eq. (19), the function $\phi^{(n)}$ is replaced by $\phi^{(n)} - \xi^{(n)} \phi^{(n-1)}$, where $\xi^{(n)}$ is a second variational parameter. Carrying this replacement in Eq. (13), a new value $\overline{B}^{(n)}$ of $B^{(n)}$ is constructed, that follows as

$$(22)\qquad \overline{B}^{(n)} = \langle K^{*}(\phi^{(n)} - \xi^{(n)}\,\phi^{(n-1)})^{*}\,,\, K(\phi^{(n)} - \xi^{(n)}\,\phi^{(n-1)})\rangle$$

$$= B^{(n)} - 2\,\mathrm{Re}[\xi^{(n)}\,C^{(n)}] + |\xi^{(n)}|^{2}\,B^{(n-1)}$$

$$= B^{(n)} - \frac{|C^{(n)}|^{2}}{B^{(n-1)}} + B^{(n-1)}\left|\xi^{(n)} - \frac{C^{(n)*}}{B^{(n-1)}}\right|^{2},$$

where

$$(23)\qquad C^{(n)} = \langle K^{*}\phi^{(n)*}\,,\, K\phi^{(n-1)}\rangle.$$

The right-hand side of Eq. (22) has, as a function of $\xi^{(n)}$, a minimum at

$$(24)\qquad \xi^{(n)} = \frac{C^{(n)*}}{B^{(n-1)}}.$$

Substitution of this value in (22) leads to

$$(25)\qquad \overline{B}^{(n)} = B^{(n)} - \frac{|C^{(n)}|^{2}}{B^{(n-1)}}.$$

First of all, this shows that $\overline{B}^{(n)} < B^{(n)}$, if $C^{(n)} \neq 0$. Further, it follows by substituting Eq. (24) in

$$(26)\qquad \overline{K\phi}^{(n)} = K\phi^{(n)} - \xi^{(n)}\,K\phi^{(n-1)},$$

that

$$(27)\qquad \langle K^{*}\overline{\phi}^{(n)*}\,,\, K\phi^{(n-1)}\rangle = 0.$$

If the original function $\phi^{(n)}$ was already such that the right-hand side of Eq. (23) vanished, then no improvement will be attained in this second minimization step. Note that this is consistent with Eq. (27). Hence, the second minimization step automatically stops after being carried out once. The resulting expression for the improved error follows as

$$(28)\qquad [\overline{ERR}^{(n)}]^{2} = [ERR^{(n-1)}]^{2} - \frac{|A^{(n)}|^{2}}{B^{(n)} - \dfrac{|C^{(n)}|^{2}}{B^{(n-1)}}}.$$

In the next section we shall discuss a procedure that leads, in each step of iteration, to the generation of a particular value of $\phi^{(n)}$. Once

$\xi^{(n)}$ has been determined, we then replace $\phi^{(n)}$ by

$$\bar{\phi}^{(n)} = \phi^{(n)} - \xi^{(n)}\,\phi^{(n-1)}, \tag{29}$$

where $\xi^{(n)}$ is given by Eq. (24). Note that the second minimization step can only be carried out from $n = 2$ onward since $\phi^{(0)}$ is not defined. The complete iteration scheme can be found in Table II.

Now, with $\bar{\phi}^{(n)}$ we carry out once more the procedure of the former section, with the result that Eq. (18) is replaced by

$$\langle \bar{r}^{(n)*}, K\bar{\phi}^{(n)} \rangle = 0, \tag{30}$$

where

$$\bar{r}^{(n)} = r^{(n-1)} - \frac{A^{(n)*}}{\bar{B}^{(n)}} K\bar{\phi}^{(n)}. \tag{31}$$

Multiplying the complex conjugate of Eq. (31) by $K\phi^{(n-1)}$, integrating over $\mathcal{D}$, using the orthogonality relations of Eq. (18), with n replaced by n-1, and Eq. (27), we obtain

$$\langle \bar{r}^{(n)*}, K\phi^{(n-1)} \rangle = 0. \tag{32}$$

By substituting Eq. (26) in (30) and using Eq. (32), we arrive at

$$\langle \bar{r}^{(n)*}, K\phi^{(n)} \rangle = 0, \tag{33}$$

or

$$\langle K^{T}\bar{r}^{(n)*}, \phi^{(n)} \rangle = 0. \tag{34}$$

Eq. (34) is the replaced form of Eq. (21) and will be of later use.

5. GENERATION OF THE VARIATIONAL FUNCTIONS

As we have seen in Section 3, an iterative improvement in the satisfaction of the integral equation is only arrived at if, in each iteration, $A^{(n)} \neq 0$. Eq. (19) shows that this can be guaranteed, if we take

Table II. The iteration scheme with second minimization step for a given sequence of variational functions $\phi^{(n)}$.

$n = 0$	Initial estimate : $f^{(0)}$
	Residual error : $r^{(0)} = g - Kf^{(0)}$
	$ERR^{(0)} = \langle r^{(0)*}, r^{(0)} \rangle^{\frac{1}{2}}$
$n = n + 1$	
	Variational function : $\phi^{(n)}$
	$A^{(n)} = \langle r^{(n-1)*}, K\phi^{(n)} \rangle$
	$B^{(n)} = \langle K^*\phi^{(n)*}, K\phi^{(n)} \rangle$
if $n > 1$	
	second minimization step : $C^{(n)} = \langle K^*\phi^{(n)*}, K\phi^{(n-1)} \rangle$
	$\xi^{(n)} = \frac{C^{(n)*}}{B^{(n-1)}}$
	$B^{(n)} := B^{(n)} - \xi^{(n)} C^{(n)}$
	$\phi^{(n)} := \phi^{(n)} - \xi^{(n)} \phi^{(n-1)}$
	$K\phi^{(n)} := K\phi^{(n)} - \xi^{(n)} K\phi^{(n-1)}$
	$\eta^{(n)} = \frac{A^{(n)*}}{B^{(n)}}$
	Improved estimate : $f^{(n)} = f^{(n-1)} + \eta^{(n)} \phi^{(n)}$
	Residual error : $r^{(n)} = r^{(n-1)} - \eta^{(n)} K\phi^{(n)}$
	$ERR^{(n)} = \langle r^{(n)*}, r^{(n)} \rangle^{\frac{1}{2}}$

storage:
$f^{(n)}$
$r^{(n)}$
$\phi^{(n)}$
$B^{(n)}$
$[K\phi^{(n)}]$

(35) $$\phi^{(n)^*} = K^T r^{(n-1)^*} \qquad \textit{(gradient method)}$$

For a similar procedure in discrete form, we refer to the method of steepest descent applied to the iterative solution of a linear algebraic equations [1]. Substituting Eq. (35) in (19) we obtain

(36) $$A^{(n)} = \langle \phi^{(n)^*} , \phi^{(n)} \rangle .$$

Using Eqs. (35) and (36), together with the results of Section 3, an iteration scheme can be developed (Table III).

Using the variational function of Eq. (35), we subsequently perform the second minimization step of Section 4. Then, in the following iteration step the variational function would be

(37) $$\phi^{(n+1)^*} = K^T \bar{r}^{(n)^*} ,$$

where $\bar{r}^{(n)}$ follows from Eq. (31). From Eqs. (34) and (37) it is observed that

(38) $$\langle \phi^{(n+1)^*} , \phi^{(n)} \rangle = 0,$$

exhibiting the orthogonality of the gradients in two subsequent iterations. As a consequence,

(39) $$\langle K^* \phi^{(n+1)^*} , r^{(n-1)} \rangle = 0,$$

where Eq. (35) and some changes of order of integration have been employed. With this property, it can be shown that the expression for $\xi^{(n+1)}$ can be simplified. In order to obtain

(40) $$\xi^{(n+1)} = \frac{C^{(n+1)^*}}{\bar{B}^{(n)}}$$

(compare Eq. (24)), we substitute in the expression

(41) $$C^{(n+1)} = \langle K^* \phi^{(n+1)^*} , K\bar{\phi}^{(n)} \rangle$$

(compare Eq. (23)), the value of $K\bar{\phi}^{(n)}$ that follows from Eq. (31). With

Table III. The iteration scheme based upon the gradient method (GR) and conjugate gradient method (CGR).

$n = 0$

Initial estimate : $f^{(0)}$

Residual error : $r^{(0)} = g - Kf^{(0)}$

$ERR^{(0)} = \langle r^{(0)*}, r^{(0)} \rangle^{\frac{1}{2}}$

$n = n + 1$

$A^{(n)} = \langle K^{T} r^{(n-1)*}, K^{T*} r^{(n-1)} \rangle$

Variational function :

gradient method	conjugate gradient method
$\phi^{(n)} = K^{T*} r^{(n-1)}$	if $n = 1$ then $\phi^{(1)} = K^{T*} r^{(0)}$
	if $n > 1$ then $\phi^{(n)} = K^{T*} r^{(n-1)} + \frac{A^{(n)}}{A^{(n-1)}} \phi^{(n-1)}$

storage:

GR	CGR
$f^{(n)}$	$f^{(n)}$
$r^{(n)}$	$r^{(n)}$
	$A^{(n)}$
	$\phi^{(n)}$

$B^{(n)} = \langle K^{*} \phi^{(n)*}, K\phi^{(n)} \rangle$

$\eta^{(n)} = \frac{A^{(n)}}{B^{(n)}}$

Initial estimate : $f^{(n)} = f^{(n-1)} + \eta^{(n)} \phi^{(n)}$

Residual error : $r^{(n)} = r^{(n-1)} - \eta^{(n)} K\phi^{(n)}$

$ERR^{(n)} = \langle r^{(n)*}, r^{(n)} \rangle^{\frac{1}{2}}$

Eq. (39) and

$$A^{(n+1)} = \langle \bar{r}^{(n)*}, K\phi^{(n+1)} \rangle \tag{42}$$

(compare Eq. (13)), we simply obtain

$$C^{(n+1)} = -\bar{B}^{(n)} \frac{A^{(n+1)*}}{A^{(n)*}} \tag{43}$$

and

$$\xi^{(n+1)} = -\frac{A^{(n+1)}}{A^{(n)}}. \tag{44}$$

Hence, in the $(n+1)$-st of iteration, $\xi^{(n+1)}$ can be computed in advance, provided the value of $A^{(n)}$ of the n-th iteration will be stored. Hence, we now observe that the variational functions are generated as

$$\begin{aligned} \phi^{(1)} &= K^{T^*} r^{(0)}, \\ \phi^{(n)} &= K^{T^*} r^{(n-1)} + \frac{A^{(n)}}{A^{(n-1)}} \phi^{(n-1)} \qquad (n > 1). \end{aligned} \tag{45}$$

(conjugate gradient method)

Using Eqs. (45) and (36), together with the results of Section 3, an iteration scheme can be developed (Table III). This iteration scheme can be regarded as a continuous version of the conjugate gradient method for solving systems of linear algebraic equations iteratively [2]. However, we have shown how to obtain optimum convergence from some first minimization principles. In the same way as in the discrete case [2], it can be shown that the sequence of functions $\{K^{T^*} r^{(n)}, n = 0,1,2,\ldots\}$ and the sequence of functions $\{K\phi^{(n)}, n = 0,1,2,\ldots\}$ are orthogonal sequences.

6. CONVOLUTION KERNELS

In this section we consider the case that $K(x,x')$ is of the convolution type

$$K(x,x') = K(x-x'). \tag{46}$$

Using the (temporal and) spatial Fourier transform technique

(47) $\quad f \leftrightarrow \tilde{f},$

where $\tilde{f}$ denotes the Fourier transform of the relevant function f, we can evaluate the expressions

(48) $\quad \phi^{(n)} = K^{T^*} r^{(n-1)} = \int_{x' \in \mathcal{D}} K^*(x'-x)\, r^{(n-1)}(x')\, dx'$

and

(49) $\quad K\phi^{(n)} = \int_{x' \in \mathcal{D}} K(x-x')\, \phi^{(n)}(x')\, dx'$

very simple. The values of $\phi^{(n)}$ and $K\phi^{(n)}$ are nothing but the inverse transforms of

(50) $\quad \tilde{\phi}^{(n)} = \tilde{K}^* \tilde{r}_{\mathcal{D}}^{(n-1)}$

and

(51) $\quad \widetilde{[K\phi^{(n)}]} = \tilde{K}\, \tilde{\phi}_{\mathcal{D}}^{(n)}.$

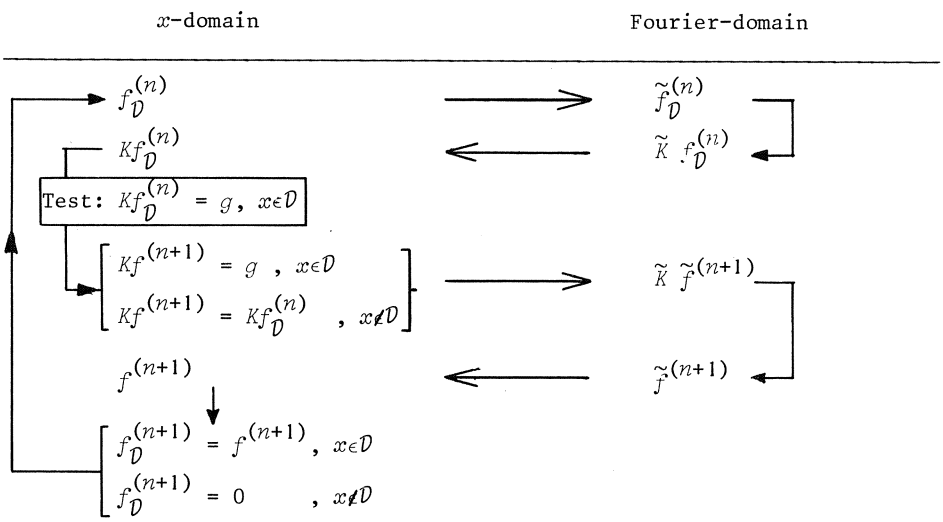
In Eqs. (50) - (51), $\tilde{r}_{\mathcal{D}}^{(n-1)}$ and $\tilde{\phi}_{\mathcal{D}}$ are the windowed Fourier transforms of $r^{(n-1)}$ ($r^{(n-1)}(x) = 0, \quad x \notin \mathcal{D}$) and $\phi^{(n)}$ ($\phi^{(n-1)}(x) = 0, \quad x \notin \mathcal{D}$). The actual use of this transform technique depends on the problem at hand and the availability of Fast Fourier Transforms.

However, the Fourier transform technique yields a tool to devise a different type of variational function that is closely related to our physical problem. The choice has been inspired by the spectral iterative techniques originated by Bojarski [3] and Ko and Mittra [4]. However, this suggested scheme (Table IV) contains no convergence criterion. In our case, however, we determine the variational function $\phi^{(n)}$ as the approximate "contrast source" at $\mathcal{D}$ that corresponds as close as possible to the "field function" $r^{(n-1)}$ at $\mathcal{D}$. This value of $\phi^{(n)}$ is obtained as the inverse transform of

(52) $\quad \tilde{\phi}^{(n)} = (\tilde{K})^{-1}\, \tilde{r}_{\mathcal{D}}^{(n-1)}.$

If $\phi^{(n)}(x) = 0, \quad x \notin \mathcal{D}$ we have the results

Table IV. The direct spectral iterative technique (SIT) suggested by Ko and Mittra.



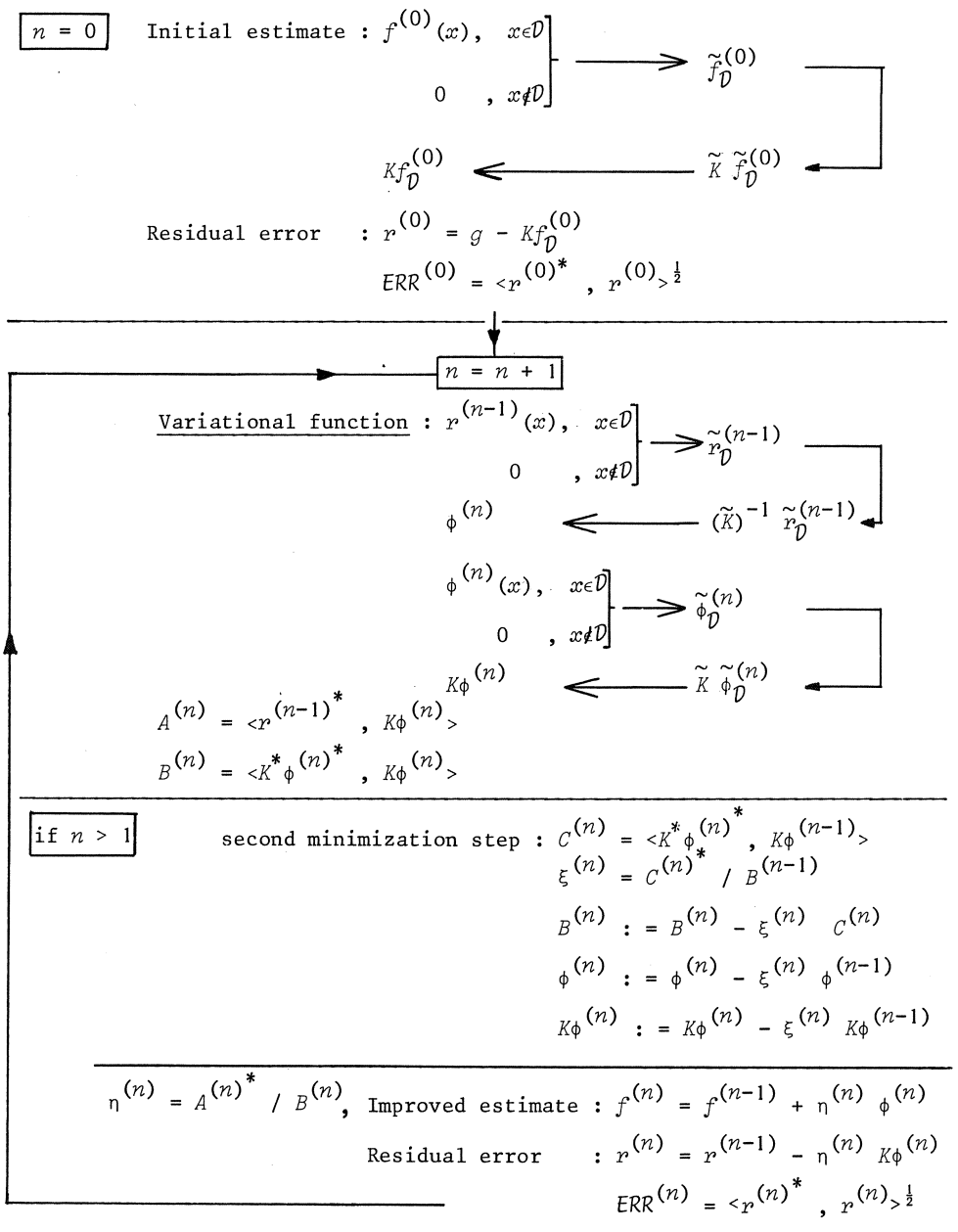
$$(53)\qquad K\phi^{(n)} = r^{(n-1)},$$

$$(54)\qquad A^{(n)} = \langle r^{(n-1)^*},\ r^{(n-1)}\rangle = B^{(n)}$$

$$(55)\qquad [ERR^{(n)}]^2 = \langle r^{(n-1)^*},\ r^{(n-1)}\rangle - \frac{\langle r^{(n-1)^*},\ r^{(n-1)}\rangle^2}{\langle r^{(n-1)^*},\ r^{(n-1)}\rangle} = 0,$$

and the iterative scheme terminates: we have arrived at the exact solution. In practice, this would not be the case. Then, we have to window the function $\phi^{(n)}$ (obtained from the inverse transform of (52)) by setting $\phi^{(n)}(x) = 0$, $x\notin\mathcal{D}$. After this contrast-source-truncation technique, the iteration scheme has to be continued (Table V). Subsequently, the minimization of the error at $\mathcal{D}$ provides a convergent iterative scheme. In this way, convergence problems arising in the direct spectral iterative techniques have been eliminated. The complete iteration scheme has been shown in Table V. The step for $n > 1$ is the second minimization step of Section 3. If one copes with limited computer storage, this step can be omitted, but the convergence can decrease dramatically.

Table V. The iteration scheme based upon the contrast-source truncation technique.



7. NUMERICAL RESULTS

The numerical convergence of the different iterative schemes for some representative scattering problems will now be presented by considering the numerical value of the normalised error $\overline{\overline{ERR}}$ (cf. Eq. (7)) as a function of the number of iterations.

Scattering of transient acoustic waves in fluids and solids

The problem of the scattering of transient elastic waves by arbitrarily shaped, three-dimensional, inhomogeneous, penetrable objects of bounded extent can be formulated in terms of a volume-integral equation over the three-dimensional domain of the scatterer [5]. As an example, we consider the scattering of a Gaussian pulsed plane wave by a cube. The incident wave propagates with speed c and has a pulse width $\tau = 3a/c$, while the cube exhibits an acoustic contrast of a factor 2 with with respect to its surroundings. The integral equation is solved iteratively using the gradient method of Table III. For the numerical calculations, the cube is subdivided

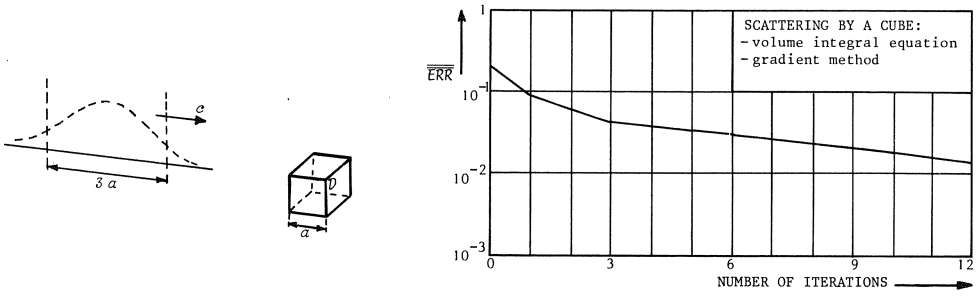


Figure 1. Scattering of a Gaussian pulsed plane wave by a cubic inhomogeneity: the normalised error as a function of the number of iterations.

into 96 identical tetrahedra, while the time step $\Delta t = 0.12\tau$. The computation time for 15 iterations is approximately 150 s (CPU) on an IBM 370/158 computer. In Fig. 1, we present the numerical value of the normalised error $\overline{\overline{ERR}}$ (Eq. (7)) versus the number of iterations. As initial guess we have taken the incident field. In view of the computer storage demands we have not experienced the conjugate gradient method for this problem. More

details can be found in [6].

Vibroseismic wave generation

The problem of the excitation of the elastic wave motion by a vibrating, rigid plate on the surface of a semi-infinite elastic solid can be formulated in terms of a boundary-integral equation over the plate [7]. As an example, we consider a rectangular plate of 1 m × 1 m, vibrating with a frequency of 100 Hz. The integral equation is solved iteratively using both the gradient method and conjugate gradient method. For the numerical calculations, the plate is subdivided in 121 subsquares. The computation time for 8 iterations amounts to about 20 s (CPU) on a UNIVAC 1100/82 computer. In Fig. 2, we present the normalised error $\overline{\overline{ERR}}$ (Eq. (7)) versus the

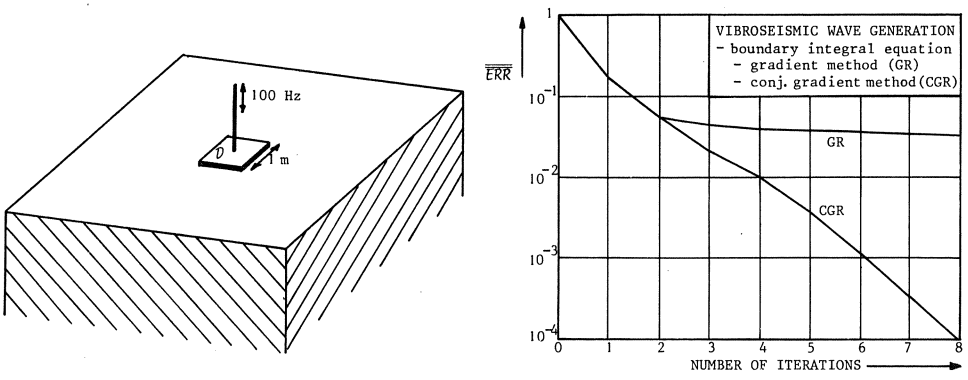


Figure 2. Vibroseismic wave generation: the normalised error as a function of the number of iterations.

number of iterations. As initial guess we have taken a rather crude approximation: all field values are zero. The convergence of the conjugate gradient method (CGR) is superior to the one of the gradient method (GR).

Microwave hyperthermia

Another problem we have studied with the present iterative techniques is the computation of the heat generated in a distribution of biological tissue (hyperthermia). The incident field is generated by a 27 MHz cooled ridged waveguide applicator. As mathematical tool for determining the field inside a two-dimensional model of a human body, we have taken a two-dimen-

sional domain-integral equation. As a typical example, the cross-section of the human pelvis is subdivided in squares of 0.005 m × 0.005 m, the number of cells now amounting to about 2500. The convergence of the iterative techniques, viz. the gradient method (GR) and conjugate gradient method (CGR) appears to be very satisfactory (Fig. 3). As initial guess we have

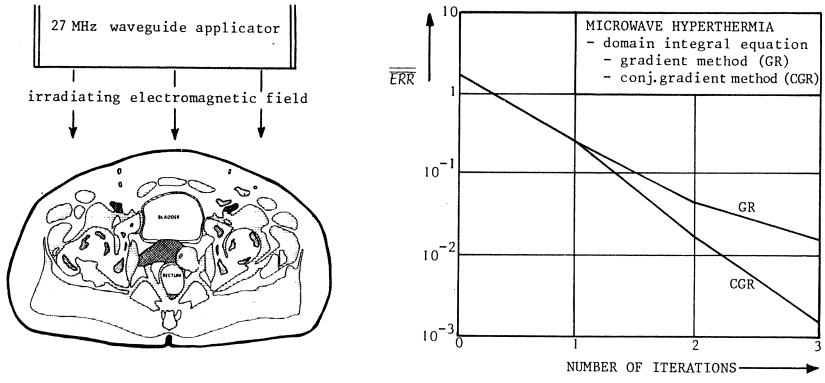


Figure 3. The microwave heating of the human pelvis: the normalised error as a function of the number of iterations.

taken the incident field. The computer time for 3 iterations is approximately 900 s (CPU) on an AMDAHL V7/B computer. More details can be found elsewhere [8]; in this paper only the gradient method has been dealt with.

Scattering by a strip

One of the canonical problems in electromagnetics and acoustics is the scattering of a time-harmonic plane wave by a strip. In electromagnetics, we consider the scattering of an E-polarized wave (the electric-field vector is parallel to the edges of the strip). This latter problem is identical to the one of acoustic scattering by a soft strip. For this problem, Ko and Mittra [4] have presented their spectral iterative technique (SIT) of Table IV. Therefore, we devote special attention to the convergence of various iterative techniques for solving the integral equation of this simple configuration. We use the spectral methods of Section 6. All convolution integrals in the iteration schemes are computed using a 1024-point FFT-routine. In order to cope with the occurrence of

the branch-point in the expression of the spatial Fourier transform of the kernel function, we have assumed a slightly lossy embedding by taking a complex wave number of $k' = k(1 + 0.01i)$, where $k = 2\pi/\lambda$, λ is wavelength. In Fig. 4, we present the numerical values of the normalised error $\overline{\overline{ERR}}$ (Eq. (7)) versus the number of iterations. The number of sample points at the strip (of width $2a$) amounts to 6 and 41, when $ka = 0.1$ and 10, respectively. We consider the following iteration schemes:

SIT : Spectral Iteration Technique (Table IV)

GR : GRadient method (Table III)

CGR : Conjugate GRadient method (Table III)

CST : Contrast-Source Truncation technique (Table V)

CCST: Conjugate Contrast-Source Truncation technique (Table V with second minimization step).

We have taken the initial estimate $f^{(0)} = cg$, where the constant c is determined in such a way that the error $\overline{\overline{ERR}}$ has a minimum value. We observe that the direct spectral iteration scheme is not convergent for small

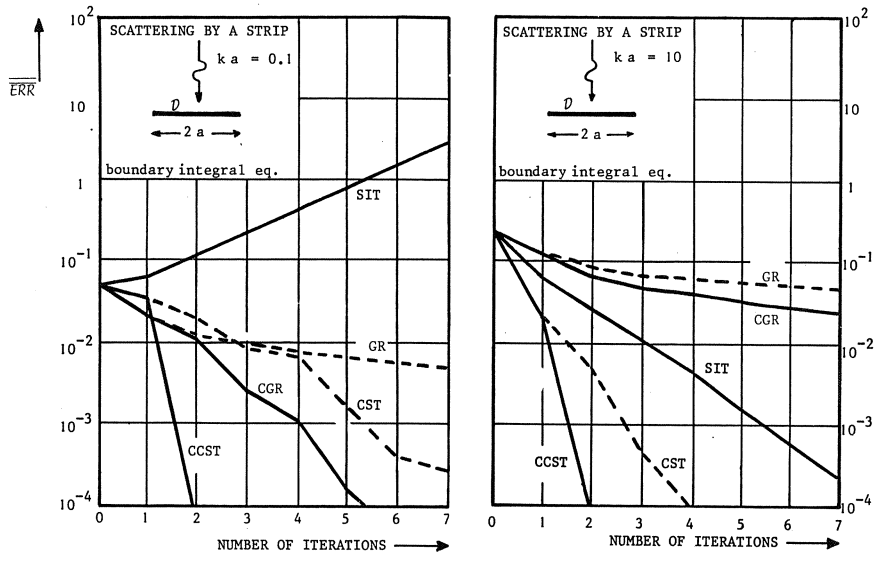


Figure 4. Scattering by a strip: the normalised error as a function of the number of iterations.

values of ka. The gradient method (GR) is not suitable for this type of problems. The conjugate gradient method (CGR) converges too slowly for large values of ka. The contrast-source truncation technique (CST) converges sufficiently in all cases considered, however the second minimization step leads to a method (CCST) that yields excellent results. Within two iterations, the error is considerably less than 1%. For all cases considered, the computation time of an iteration is roughly the same. On an AMDAHL V7/B computer, it is approximately 0.5 s (CPU). More details can be found in [9].

The electric-field problem of an interdigital transducer

The analysis of the computation of the electric field excited by an interdigital transducer in a multi-layered structure can be reduced to the solution of a boundary-integral equation over the surfaces of the transducer fingers. This integral equation is solved iteratively using the contrast-source truncation technique with second minimization step (Table V). In the spectral representation of the kernel, a transfer-matrix formu-

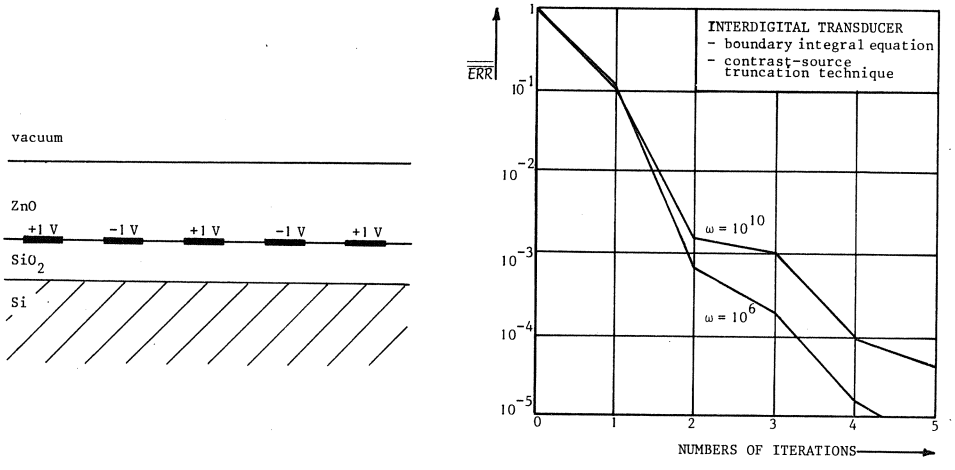


Figure 5. The excitation of the electric field by a transducer in a multi-layered structure: the normalised error as a function of the number of iterations. The radial frequency of operation is 10^6 and 10^{10}, respectively.

lation accounts for the presence of the layers [10]. In Fig. 5, we present the normalised error $\overline{\overline{ERR}}$ (Eq. (7)) versus the number of iterations. The width of a transducer finger is discretized in 64 sample points (using a 256-points FFT-routine). The computation time for 5 iterations is approximately 5 s (CPU) on an AMDAHL V7/B computer. The remarkable convergence of the present iterative technique for this problem has to be noted.

8. CONCLUDING REMARKS

A scheme has been developed by which an integral equation can be handled computationally. Experience has shown that the iterative technique is very efficient and can indeed handle configurations that are beyond the reach of a direct, i.e. non-iterative method. A point of further investigation is the question of the choice of the variational functions.

REFERENCES

[1] SARKAR, T.K., K.R. SIARKIEWICZ & R.F. STRATTON, *Survey of numerical methods for solutions of linear equations for electromagnetic field problems*, IEEE Trans. Antennas Propagat. AP-29, 847-856, 1981.

[2] HESTENES, M. & E. STIEFEL, *Method of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards 49, 409-436, 1952.

[3] BOJARSKI, N.N., *The k-space formulation of the scattering problem in the time domain*, J. Acoust. Soc. Am. 72, 570-584, 1982. (This paper includes a comprehensive review of Bojarski's work.)

[4] KO, W.L. & R. MITTRA, *A new approach based on a combination of integral equation and asymptotic techniques*, IEEE Trans. Antennas Propagat. AP-25, 187-197, 1977.

[5] HERMAN, G.C., *Scattering of transient acoustic waves in fluids and solids*, Ph.D. Thesis, Delft University of Technology, The Netherlands, 1981.

[6] HERMAN, G.C. & P.M. VAN DEN BERG, *A least-square iterative technique for solving time-domain scattering problems*, J. Acoust. Soc. Am. 72, 1947-1953, 1982.

[7] VAN ONSELEN, C., *Analysis of the elastic wave motion in a semi-infinite solid, excited by a vibrating disk on its stress-free surface ("vibroseis problem")*, M.Sc. Thesis, Laboratory of Electromagnetic Research, Delft University of Technology, The Netherlands, Report 1980-22, 1980.

[8] VAN DEN BERG, P.M., A.T. DE HOOP, A. SEGAL & N. PRAAGMAN, *A computational model of the electromagnetic heating of biological tissue with application to hyperthermia cancer therapy*, IEEE Trans. Biomed. Eng. BME-25, 797-805, 1983.

[9] VAN DEN BERG, P.M., *Iterative computational techniques in scattering, based upon the integrated square error criterion*, Laboratory of Electromagnetic Research, Delft University of Technology, The Netherlands, Report 1983-18, 1983. (To be published in IEEE Trans. Antennas Propagat.)

[10] VAN DEN BERG, P.M., W.J. GHIJSEN & A. VENEMA, *The electric field problem of an interdigital transducer in a multi-layered structure*, Laboratory of Electromagnetic Research, Delft University of Technology, The Netherlands, Report 1984-05, 1984. (Submitted for publication to IEEE Trans. Microw. Theory Techn.)

ON THE ROLE OF A MODELLING SYSTEM AS A BRIDGE BETWEEN MODEL BUILDERS AND ALGORITHM DEVELOPER

J.J. BISSCHOP

This paper is on the border line between the field of numerical mathematics and the field of information technology. It concentrates on two distinct professionals, namely a model builder and an algorithm developer. Despite the fact that their needs and requirements are distinctly different, their work cannot be meaningful in isolation of each other. In order to bridge the gap between these two worlds, the availability of a modeling system becomes a necessity. Such a system offers a modeling language for the expression of models, coupled with an automatic interface with algorithms. The result is an improved modeling technology with advantages for both the model builder and the algorithm developer. The General Algebraic Modeling System GAMS is referred to as an example of improved modeling technology for large-scale model building activities resulting in mathematical programming models.

1. INTRODUCTION

For the sake of discussion, one can distinguish two different persons that provide input into mathematical modeling exercises. They are the model builder and the algorithm developer. The model builder is the person that translates aspects of reality into mathematical relationships. His prime goal is to solve a real-life problem, and his mathematical model is the indirect route to the solution. He views the solution algorithm as a necessary tool to get to the solution. The algorithm developer, on the other hand, is not interested in the semantics of mathematical relationships, but only in the mathematical structure, for which the algorithm has been designed.

The main point of this paper is to argue that a modeling system provides a necessary communication link between a model builder and an algorithm

developer. The second main point is to argue that the cost-benefit ratio associated with both model building and algorithm development will be greatly improved with the existence of a modeling system.

2. MODELING TECHNOLOGY FOR THE MODEL BUILDER

Whenever we talk about the current modeling technology in this section, we talk about a technology in the absence of a sophisticated modeling system for the automation of the modeling process.

The current modeling technology places a heavy burden on the model builder. There are several steps in the model building process for which the model builder is responsible. They are the following.

i. The model builder must develop a mathematical representation of the model equations and the associated data components. The data will specify a particular instance of the mathematical model.

ii. The model builder must translate the mathematical representation into an input format that is required by the selected solution algorithm designed to solve the mathematical model.

iii. The model builder must set up the interface with the solution algorithm.

iv. The model builder must translate the output results of the algorithm into readable reports using the original notation for data and model.

The following comments should be made.

ad i. Experience in the field of large-scale mathematical programming has shown that the choice of mathematical notation and the final mathematical representation of the model is a personal one, and often insufficient to provide a clear statement of the mathematical model.

ad ii. Even though the input format for an algorithm usually takes on a simple and unambiguous form, it is frequently not suitable for understanding the underlying model. The reason for this is that the input format is designed around the algorithm, and therefore quite different from the rich mathematical structures that are needed for a transparant and understandable description of the original model.

ad iii. The interface is usually in the form of a control program, and in most instances does not prove a bottleneck in the overall modeling process.

ad iv. The translation of output results into readable reports is time consuming, and usually results in custom-made reports. Any change in the original model may have a strong impact on the amount of work required in this step.

In addition to the above comments on the current technology for the model builder, there are other critical comments to be made. Besides knowning how to build models, a model builder must be able to comprehend and prepare the input to (one or more of) the solution algorithms. The preparation of essentially error-free input for a solution algorithm for a large-scale model is an activity to be measured in terms of months. Any subsequent changes to the original model requires changes in the algorithmic input which is in the order of days to weeks. The entire translation process will almost certainly create translation errors, some of which will not be straightforward to detect.

It is precisely the error-prone translation step from a human readable form into a computer readable form which is costly in terms of time, manpower and money. That is why this translation step must be automated.

The ideal technology for a model builder is one in which his task is limited to model building alone. Such technology can be provided by a modeling system which offers a clear and flexible modeling language to be read by both man and computer. In this way, the model builder is guided by the syntax of a powerful language rather than being forced into inventing a new and personal notation which cannot be checked by a compiler. In addition to a modeling language, a modeling system must provide an automatic interface with solution algorithms, thereby releasing the model builder from an important burden. This automatic interface must be coupled with an automatic report generator, which is capable of translating the output of the solution algorithm into meaningful reports using a notation which resembles the notation of the modeling language.

There are several positive consequences attached to an ideal modeling technology for a model builder. First of all, no skills other than modeling skills are required. This allows the model builder not only to concentrate on his main activity, but also to complete a model development in terms of weeks rather than months. In addition, model changes can then be accomplished in terms of hours to days rather than days to weeks. Last

but not least, there will be significantly fewer mistakes using the ideal technology because of the following two reasons. First there are no errors introduced by the modeling system during the translation step from the algebraic representation into the algorithmic representation. Secondly, a good compiler will detect many types of symbolic errors and domain violations in data definitions and model equations.

From the viewpoint of the model builder we may summarize by stating that in the presence of a sophisticated modeling language and modeling system, the entire process of model building can be made more reliable and can be done more cost effective than without the presence of such a language and system.

In the field of mathematical programming, the general algebraic modeling system GAMS is an example of a sophisticated modeling language coupled with an automatic interface with a series of large-scale linear and nonlinear programming codes. The purpose of this paper is not to explore the particulars of any modeling system. Instead, we only want to point at the importance of such a system. For further details, the reader is referred to references (1) and (2).

3. MODELING TECHNOLOGY FOR THE ALGORITHM DEVELOPER

Just as in the previous section, whenever we talk about the current modeling technology in this section, we talk about a technology in the absence of a sophisticated modeling system for the automation of the modeling process.

Under the current modeling technology the algorithm developer must assume several responsibilities. They are the following.

i. The algorithm developer must provide and document an interface for each algorithm. In the absence of an already existing interface, the choice will be a personal one.

ii. The algorithm developer must provide a correctness check on the input in order to guarantee that the input is compatible with the algorithmic requirements.

iii. The algorithm developer must frequently search for or derive structural information from the input. Examples of structural information used

by nonlinear programming codes are sparsity information, derivative information, linearity information, and other functional form information.

iv. The algorithm developer must test his algorithm, and compare its performance with others.

v. The algorithm developer must market his algorithm, assuming of course, that the algorithm is of interest to a variety of algorithm users.

The following comments should be made.

ad i. In the field of nonlinear programming, for instance, one finds that as a result of different personal choices, there exist several different algorithmic input representations. Some of them are not compatible with each other in the sense that there does not exists an invertible function mapping one into the other.

ad ii. Depending on the input format expected by the solution algorithm, the development of a comprehensive correctness check can be an extremely time consuming task.

There are several other critical comments that can be made about the current technology for the algorithm developer. Not only does it take extensive development time to provide a good user's interface and appropriate documentation thereof, the verification of the correctness of the data input stream every time the algorithm is being used is also a costly stop in terms of computer time. In addition, the task of comparing a new algorithm to existing algorithms turns out to be time consuming, as each existing code requires its own input representation of the same test problem(s). If the code is developed at a university or at any other organization which is not set up for the commercial support of software, experience in the area of mathematical programming software has shown that algorithms survive in only a restricted environment for just a limited number of years.

The ideal technology for an algorithm developer is one in which his task is limited to algorithmic development alone. Such technology can be provided by a modeling system which provides not only a correct input, but also the structural information as it is required by the algorithm. In addition, the modeling system provides the infrastructure to test and compare a battery of real-life problems which can be understood by persons who might want to verify or repeat some or all of the test results.

There are several positive consequences attached to such an ideal technology for an algorithm developer. First of all, the main concentration will then be on algorithmic development alone, which can only result in improved algorithms. Secondly, the time saved from not having to provide a documented interface can now be used to make sure that the algorithm will recover from any errors and that these errors are reported back correctly to the modeling system. Thirdly, the availability of a modeling system as an umbrella for many solution codes will not only create a marketing device for many algorithm developers, it will also facilitate a healthy competition among them. Such competition will be of direct benefit to model builders since the best codes will be available to them by merely requesting their use. Finally, with the modeling system as a marketing device, it may now become worthwile to develop special codes to speed up the solution time for a large variety of structured problems. In that case, the presence of a large variety of solution codes is not a problem for the model builder anymore, since efforts on his part to acquire and maintain these codes is reduced to practically nothing.

From the viewpoint of the algorithmic developer we may summarize by stating that in the presence of a sophisticated modeling language and modeling system, the entire process of algorithm development can be made more reliable and can be done more cost effectively than without the presence of such a language and system. Again, in the field of mathematical programming, the GAMS modeling system serves as a good illustration, since it incorporates several linear and nonlinear programming codes developed at different institutions.

4. CONCLUSIONS

In this paper we have portrayed the current technology for both model builders and algorithm developers. By comparing the current technology with an ideal technology, it has become clear that a sophisticated modeling system will provide extensive benefits for both parties. In addition, since it is the same instrument that provides these benefits, a sophisticated modeling system becomes a bridge, permitting a fast, practical and cost effective communication link between both model builder and algorithm developer.

REFERENCES

[1] BISSCHOP, J. & A. MEERAUS, *On the Development of a General Algebraic Modeling System in a Strategic Planning Environment*, Mathematical Programming Study 20 (1982), 1-29.

[2] FOURER, R., *Modeling Languages versus Matrix Generators for Linear Programming*, Transactions on Mathematical Software (1983), Vol. 9, No. 2, 143-183.

ON THE INTEGRATION OF MULTIGRID RELAXATION INTO A ROBUST FAST-SOLVER FOR TRANSONIC POTENTIAL FLOWS AROUND LIFTING AIRFOILS

J.W. BOERSTOEL & A. KASSIES

A robust fast-solver for the calculation of transonic potential flows around lifting airfoils is presented. The solver is a combination of Newton iteration and CS (correction scheme) multigrid relaxation. This combination allows a simpler analysis of convergence properties than the FAS (full approximation storage) multigrid relaxation that is usually applied, and was selected for this reason. The solver has been implemented in the NLR code TRAFS and has been extensively tested numerically.

Three new salient features are required to make the fast solver really effective.

a. *The circulation in the asymptotic far-field solution is computed by a procedure based on Newton iteration applied to the Kutta condition.*

b. *In order to reduce as much as possible artificial viscosity in shocks and the associated stiffness effects (these effects hamper fast Newton-iteration convergence), upwinding based on mass-flux-vector splitting was introduced.*

c. *At certain stages of the algorithm, velocity overshoots and corresponding potential jumps at shocks are allowed. These are usually generated by large long-wavelength solution corrections. It is shown that these jumps also arise in other algorithms than that of TRAFS. The potential jumps are translated here in corresponding shock-position updates by a simple and cheap linear extrapolation technique in potential distributions. Results of numerical experiments illustrate these features.*

The TRAFS algorithm is shown to be at least competitive to existing FAS multigrid algorithms, and is at this moment equally fast as very efficient AF algorithms. It can compute subsonic and transonic potential flows around lifting airfoils in 8-25 line-relaxation work units.

*) The results presented were partially obtained under contract 1853 with the Netherlands Agency for Aerospace Research NIVR, NLR order numbers 101.813, 526.801.

1. INTRODUCTION

The purpose of the exploratory research study reported here was the design of a robust multigrid fast-solver for the calculation of transonic potential flows around lifting airfoils. Calculation speed should be as high as possible. The study should clarify problem areas that prevent multigrid fast-solvers to have maximum expected efficiency. The major problem areas turned out to be the fast calculation of the lift and of the shock waves.

In literature, many publications about the application of multigrid techniques to transonic potential flow problems may be found. Recent publications are those of Raj [1] (multigrid built into FL022), McCarthy and Reyhner [2] (axisymmetric inlets), Pelz and Steinhoff [3] (lifting airfoils), Deconinck and Hirsch [4] (nonlifting symmetrical airfoils), Shmilovich and Caughey [5] (FL030 for wing-body configurations), Brown [6] (local mesh refinement in inlet-flow calculations), Boerstoel [7] (lifting airfoils) and Van der Wees, Van der Vooren and Meelker [25]. Older publications are those of Jameson [8], Arlinger [9], Fuchs [10] and South and Brandt [11]. For background information about the multigrid technique, excellent recent surveys have been presented by Stüben and Trottenberg [12], Hackbusch [13] and Brandt [14]; see also NASA CP2202 [15]: "Multigrid Methods" and the very stimulating paper by Brandt, [17].

As far as transonic potential flow calculations are concerned, it is clear from these publications that multigrid did not yet bring what can hopefully be achieved from extrapolations of experiences with elliptic problems [17]. Compared to nonlinear successive line over-relaxation, considerable improvements could be reported [1,12]. However, the convergence rates are worse than in elliptic problems; see the convergence rate stalling and local non-convergence in e.g. [1] (page 7), [3] (fig. 12), [4] (figs. 6,12), [6] (page 164, fig. 13a), [8] (page 125), etc., and convergence rates of the order 0.85-0.95 (instead of 0.55, say) in [1] (figs. 10, 12), [2] (table 2), [3] (figs. 8, 10, 12), [4] (figs. 5, 6, 12, 13), for example. This implies that calculation times seems to be too large by a factor of the order 2-5, compared to what seems to be possible in elliptic problems.

In the study reported here, it has become clear that these convergence rate problems are due to inefficient treatment by the multigrid process of the lift and of the shock waves; this considerably slows down convergence speeds.

In order to make multigrid really effective, three major numerical

modifications to current practices were introduced.

-The circulation (lift), or rather, the far-field contribution to the overall solution is computed by a new and simple procedure outside the multigrid algorithm, using in fact a simple domain-splitting (zonal) approach.

-To achieve as much transparancy of the overall convergence process as possible, the iteration on the nonlinearity in the problem (with elliptic and hyperbolic zones separated by shock waves and sonic lines) was decoupled from the iterative solution of large linear sparse systems. Iteration on the nonlinearity was done by Newton's process, and that on the linear sparse systems by multigrid relaxation.

-Mass-flux vector splitting instead of artificial compressibility or artificial viscosity is applied for three reasons:

- -to rigorously exclude expansion shocks at sonic lines;
- -to eliminate (almost) completely velocity overshoots, undershoots, smearing, or wiggles in converged final results;
- -to allow, during multigrid iteration towards a converged result, in a precisely controlled way, preshock or postshock velocity overshoots or undershoots.

The preshock or postshock velocity overshoots or undershoots, generated by efficient multigrid relaxation cycles and allowed in each linear stage of the iterative solution process, are eliminated at completion of the linear stages by a simple shock-displacement algorithm, before they can do any harm.

These novel features are extensively discussed in the paper.

The NLR multigrid study has resulted in a new fast-solver code TRAFS (Transonic Airfoil Flow-calculation System) for the fully conservative calculation of transonic potential flow around lifting airfoils. It has been released for general use in the NLR about two years ago [18,19]. TRAFS was found to be robust, two to three times faster as Jameson's FL06 code [18] and not slower than Holst's very efficient TAIR code (based on AF) [16]. TRAFS does not require user tuning of (convergence or viscosity) parameters, and has an iterative graphical analysis package, so that it is very user-friendly.

2. CONTINUUM-MODEL EQUATIONS

The continuum-model equations representing steady transonic potential flows around given airfoils are specified as follows. It is assumed that

a regular and sufficiently smooth mapping

(1) $\quad x = x(\xi,\eta), \quad y = y(\xi,\eta)$

from a rectangular computational domain for (ξ,η) to the flow domain around the airfoil is given. The mass-conservation equation is

(2) $\quad \nabla^T F = 0 \quad$ in the flow field,

where T means transposition, and

(3) $\quad \nabla^T = [\partial/\partial\xi, \partial/\partial\eta],$ (divergence or gradient)

(4) $\quad F = \rho h U,$ (contravariant mass-flux vector)

(5) $\quad U = G\nabla\phi,$ (contravariant velocity vector)

(6) $\quad G = H^{-1}H^{-1T},$ (contravariant metric tensor)

(7) $\quad H = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix},$ (Jacobian of mapping (1))

(8) $\quad h = \det(H),$ (determinant of Jacobian)

(9) $\quad \rho = f^{1/(\gamma-1)},$ (density)

(10) $\quad f = 1 + \tfrac{1}{2}(\gamma-1)M_\infty^2(1-q^2),$

(11) $\quad q^2 = \nabla^T g G \nabla\phi = U^T\nabla\phi,$ (squared local speed)

(12) $\quad a^2 = fM_\infty^{-2}.$ (squared local sound speed)

The boundary conditions are that on the airfoil,

(13) $\quad F^T n = 0$ on the airfoil,

where n is a unit vector in (ξ,η)-coordinates normal to the airfoil image in the computational domain, and that at free-stream infinity,

(14) $\quad \phi = \bar{x} + \Gamma\bar{\theta} + o(1) \quad$ for $\quad (x,y) \to \infty,$

where $\bar{x}$, $\bar{\theta}$ are coordinates aligned to the free-stream flow according to linearized theory:

(15) $\quad \bar{x} = x\cos(\alpha_\infty) + y\sin(\alpha_\infty),$

(16) $$\bar{y} = -x \sin(\alpha_\infty) + y \cos(\alpha_\infty),$$

(17) $$\bar{\theta} = \frac{1}{2\pi} \arctan(\beta\bar{y}/\bar{x}), \quad \beta = (1-M_\infty^2)^{\frac{1}{2}}.$$

The airfoil incidence α_∞ and the free stream Mach number $M_\infty \in [0,1)$ are given numbers. The circulation Γ is determined by the Kutta condition at the trailing edge of the airfoil:

(18) $$0 \leq q_{te} < \infty.$$

This inequality constraint may be transformed into a single algebraic equation at the trailing edge, if the mapping (1) is suitably defined, see further equation (48) below where this issue is further considered.

The velocity, sound speed, and density have been scaled by the free-stream values of the velocity and density, see (14), and (9-12).

At compression shocks, $\nabla^T F = 0$ has to be replaced by shock relations. Further, expansion shocks must be excluded. These issues will be analyzed in detail in sections 3 and 4 below.

The conservation equation, boundary conditions, and Kutta condition represent a boundary-value problem, augmented by an algebraic Kutta equation, for the unknown velocity potential ϕ and the circulation Γ. Under appropriate smoothness assumptions, this equation system has solutions. The potential of solutions has period Γ when the airfoil is encircled once.

The linear first-variation equations, that may be derived from the nonlinear conservation equations and boundary conditions, are needed for two purposes.

-They will be used to design Newton-iteration algorithms on the nonlinearity in the problem and to analyze the convergence of the nonlinearity.

-They will be the point of departure for the design of stable nonlinear difference schemes based on splitting of the mass flux vector F.

The issues of stability and of the iteration on the nonlinearity by Newton's methods are closely related to each other. To see how, in the boundary-value problem (2-18), replace ϕ by $\phi + d\phi$, and Γ by $\Gamma + d\Gamma$:

(19) $$\left.\begin{aligned} \phi &\Leftarrow \phi + d\phi \\ \Gamma &\Rightarrow \Gamma + d\Gamma \end{aligned}\right\} \text{ (in 2-18).}$$

a. If (ϕ,Γ) in $(\phi+d\phi,\Gamma+d\Gamma)$ is regarded to be a known approximate solution of the nonlinear boundary-value problem, $(d\phi,d\Gamma)$ may be regarded as a solution correction to (ϕ,Γ), to be determined such that $(\phi+d\phi,\Gamma+d\Gamma)$ becomes the solution of the boundary-value problem. The linear first-variation equation system, obtained by linearizing in $(d\phi,d\Gamma)$ after substitution of (19) into the nonlinear equation system, may then be used to generate approximate solution corrections $(d\phi,d\Gamma)$ with an error of $O(\|d\phi\|^2, |d\Gamma|^2, \|d\phi\|\,|d\Gamma|)$.

b. Alternatively, if (ϕ,Γ) in $(\phi+d\phi,\Gamma+d\Gamma)$ is a solution, a necessary condition for this solution to be unique (stable) is that the first-variation problem for $(d\phi,d\Gamma)$ is well-posed and has $d\phi = d\Gamma = 0$ as a unique solution. The analysis of first-variation equation systems allows us to draw conclusions about necessary stability conditions to be met.

The linear first-variation equations for $d\phi$, obtained after substituting (19) in (2-18), may be summarized as follows.

(20) $\quad \nabla^T dF = -\nabla^T F$ in flow field,

(first-variation mass-conservation euqation)

(21) $\quad dF = P\nabla d\phi,$

(first-variation mass-flux vector)

(22) $\quad P = \rho h(G-UU^T/a^2),$

(2*2 coefficient matrix)

(23) $\quad dF^T n = -F^T n$ on the airfoil,

(24) $\quad d\phi = d\Gamma\,\bar{\theta}$ for $(x,y) \to \infty,$

where $d\Gamma$ is a correction circulation corresponding to $d\phi$. All second-order terms in $d\phi$ (these are products of $d\phi_\xi$, $d\phi_\eta$, $d\phi_{\xi\xi}$, $d\phi_{\xi\eta}$, $d\phi_{\eta\eta}$ with each other) have been neglected.

This first-variation equation system has to be augmented by a first-variation equation for the Kutta condition (see equation (58) below). Moreover, when compression shocks are present, first-variation relations for the shock relations and for shock displacements have to be formulated, see sections 3 and 4 below.

Interesting conclusions may be drawn when the coefficient matrix P of the first-variation mass-flux vector dF is split by an eigenvalue-eigen-vector decomposition. The result is

$$P = \rho\, hH^{-1}V\begin{bmatrix}1-M^2 & 0\\ 0 & 1\end{bmatrix}V^T H^{-1T}, \tag{25}$$

where $M = q/a$ is the local Mach number and V is an orthonormal matrix mapping vectors in the orthogonal streamline coordinates (s,n) to vectors in the Cartesian (x,y) coordinates (s arc length along streamlines, n arc length normal to streamlines):

$$V = \begin{bmatrix}x_s & x_n\\ y_s & y_n\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix},\quad VV^T = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}, \tag{26}$$

where θ is the flow angle with respect to the x-axis. The derivation is based on a decomposition of the exterior product UU^T of the contravariant velocity vector U with its transpose:

$$\begin{aligned} UU^T/a^2 &= a^{-2}G\nabla\phi\nabla^T\phi G = \\ &= a^{-2}H^{-1}VV^T H^{-1T}\nabla\phi\nabla^T\phi H^{-1}VV^T H^{-1T} = \\ &= a^{-2}H^{-1}V\begin{bmatrix}\phi_s\\ \phi_n\end{bmatrix}[\phi_s\phi_n]V^T H^{-1T} = \\ &= H^{-1}V\begin{bmatrix}\phi_s^2/a^2 & \phi_s\phi_n/a^2\\ \phi_s\phi_n/a^2 & \phi_n^2/a^2\end{bmatrix}V^T H^{-1T} = \\ &= H^{-1}V\begin{bmatrix}M^2 & 0\\ 0 & 0\end{bmatrix}V^T H^{-1T}, \end{aligned} \tag{27}$$

where we have used that

$$\left.\begin{aligned}\phi_n &= 0\\ \phi_s^2/a^2 &= q^2/a^2 = M^2\end{aligned}\right\}\ \begin{cases}\text{in streamline}\\ \text{coordinates.}\end{cases} \tag{28}$$

It follows that, in subsonic flow, P is positive definite while, in supersonic flow, the eigenvalue $(1-M^2)$ is negative, when the mapping is regular (H nonsingular), and $h > 0$.

The eigenvalue-eigenvector decomposition (25) will be seen to play a

role in the design of stable finite-difference equations.

3. FINITE-DIFFERENCE DISCRETIZATION

The fully-conservative finite-difference approximations of the mass-conservation equation in arbitrary curvilinear coordinates are derived with mass-flux vector splitting. This replaces the usual artificial viscosity or artificial density concepts because of its improved nonlinear stability properties required in fast nonlinear iteration processes.

The discretization of the mass-conservation equation is based on the following list of requirements.

-The mass-conservation equation should be approximated in a fully conservative way, to prevent unacceptable numerical-error accumulation in the discrete conservation equation system.

-Shocks will be captured and not fitted. This keeps shock treatment relatively simple.

-Because the physical shocks thickness is always orders of magnitude smaller than any reasonable fine mesh length, the thickness of numerical shocks should be minimal, i.e. one mesh size at most, independent of the shock strength and its orientation with respect to a grid.

-The finite-difference approximations of the mass-conservation equation over shocks should be made as independent as possible from finite differences in potentials over the shock discontinuity. For the latter differences have 0(1) approximation errors and introduce in general smearing or wiggles, and stiffness of the discrete conservation equation system impeding near shocks fast nonlinear convergence.

-Over sonic lines, expansion shocks have to be rigorously suppressed.

-In smooth supersonic zones and over sonic lines, the discrete flow should be "smooth". This will be achieved by introducing implicit artificial viscosity through upwinding.

The finite-difference expressions are formulated on a rectangular grid. Around each grid point (i,j) of the grid, a square cell and a discrete mass-conservation equation are defined. Introducing the notation

$$f_{i+\alpha,j+\beta} = f_{\alpha\beta} \tag{29}$$

for a grid-point value near a grid point (i,j), the mass-conservation equation for a cell (i,j) is defined as

$$\delta^{\xi}_{00} F^{d1} + \delta^{\eta}_{00} F^{d2} = 0, \tag{30}$$

where $F^d = [F^{d1}, F^{d2}]^T$ is a discrete mass-flux-vector grid function defined below, while δ^{ξ}_{00} is a central-difference operator in ξ-direction,

$$\delta^{\xi}_{00} F^{d1} = \delta^{\xi}_{i,j} F^{d1} = F^{d1}_{i+\frac{1}{2},j} - F^{d1}_{i-\frac{1}{2},j}. \tag{31}$$

The central-difference operator in η-direction δ^{η}_{00}, is defined analogously. The components F^{d1} and F^{d2} of the mass-flux vector in (30) have thus to be defined at the four cell-face midpoints $(\pm\frac{1}{2}0) = (i\pm\frac{1}{2},j)$ and $(0\pm\frac{1}{2})$.

The mass-flux vector $F^d_{\frac{1}{2}0}$ and its ξ-component $F^{d1}_{\frac{1}{2}0}$ at the cell face $(\frac{1}{2}0)$ are defined in three steps.

a. Evaluation of flow condition at cell face $(\frac{1}{2}0)$. The discrete potential gradient is computed using central-difference formulas centred at the cell-face midpoint:

$$(\nabla\phi)_{\frac{1}{2}0} = [\phi_{10}-\phi_{00}, \tfrac{1}{4}(\phi_{11}+\phi_{01}-\phi_{1-1}-\phi_{0-1})]^T, \tag{32}$$

and the contravariant velocity vector is discretized by

$$U_{\frac{1}{2}0} = G_{\frac{1}{2}0}(\nabla\phi)_{\frac{1}{2}0}. \tag{33}$$

The velocity q, density ρ and Mach number $M = q/a$ at the cell-face midpoint are evaluated from these two expressions using the algebraic relations (9-12).

b. Evaluation of two auxiliary mass-flux vectors at cell face $(\frac{1}{2}0)$:

$$F^c_{\frac{1}{2}0} = \rho^* q^* h_{\frac{1}{2}0}(U/q)_{\frac{1}{2}0}, \tag{34}$$

$$F^a_{\frac{1}{2}0} = \{(\rho q)_{\frac{1}{2}0} - \rho^* q^*\} h_{\frac{1}{2}0}(U/q)_{\frac{1}{2}0}, \tag{35}$$

where $\rho^* q^*$ is the maximum of the physical mass flux per unit area, ρq; this maximum occurs at sonic conditions. Observe that F^c and F^a may be summed to the usual contravariant mass-flux vector (4).

c. Definition of the discrete mass-flux vector $F^d_{\frac{1}{2}0}$ at the cell-face $(\frac{1}{2}0)$. Assume that the ξ-component of $U_{\frac{1}{2}0}$ is positive,

(36) $\quad U^1_{\frac{1}{2}0} > 0,$

so that the upstream direction is in $-\xi$ direction. A distinction is now made between four flow conditions in order to realize in each condition certain numerical and physical effects. The distinction is made using tests on the Mach number at both the cell face ($\frac{1}{2}0$) and the upstream (see test (36)) cell face ($-\frac{1}{2}0$). In each of the four conditions, the discrete mass flux F^d is put equal to a suitable combination of F^c and F^a:

subsonic cell face: $M_{\frac{1}{2}0} < 1$ and $M_{-\frac{1}{2}0} < 1$:

(37a) $\quad F^d_{\frac{1}{2}0} = F^c_{\frac{1}{2}0} + F^a_{\frac{1}{2}0}.$

sonic cell face: $\quad M_{\frac{1}{2}0} \geq 1$ and $M_{-\frac{1}{2}0} < 1$:

(37b) $\quad F^d_{\frac{1}{2}0} = F^c_{\frac{1}{2}0},$

supersonic cell face: $M_{\frac{1}{2}0} \geq 1$ and $M_{-\frac{1}{2}0} \geq 1$:

(37c) $\quad F^d_{\frac{1}{2}0} = F^c_{\frac{1}{2}0} + F^a_{-\frac{1}{2}0},$ (NB upstream bias)

shock cell face: $\quad M_{\frac{1}{2}0} < 1$ and $M_{-\frac{1}{2}0} \geq 1$:

(37d) $\quad F^d_{\frac{1}{2}0} = F^c_{\frac{1}{2}0} + \{\underline{\text{if}}\ (\rho q)_{\frac{1}{2}0} > (\rho q)_{-\frac{1}{2}0}\ \underline{\text{then}}\ F^a_{-\frac{1}{2}0}\ \underline{\text{else}}\ F^a_{\frac{1}{2}0}\}.$

The tests on the Mach numbers at the cell face ($\frac{1}{2}0$) and the upstream cell face ($-\frac{1}{2}0$) allow a rigorous distinction between subsonic flow, supersonic flow, acceleration through sonic conditions (expansion shocks have to be suppressed here) and deceleration through sonic conditions (shocks should be allowed and extend over one mesh in streamline direction). The common discretizations using artificial viscosity or artificial density concepts make only a distinction between subsonic and supersonic flow.

The upstream bias in the Mach number tests and in F^d at supersonic and shock cell faces is changed when the flow direction at face ($\frac{1}{2}0$) is in negative direction, $U^1_{\frac{1}{2}0} < 0$. At cell-face midpoints $(0\frac{1}{2}) = (i,j+\frac{1}{2})$, analogous definitions hold.

The difference schemes for the mass-conservation equation have the following properties.

a. The schemes may easily be verified to be fully conservative and to capture shcoks.

b. At sonic lines, or more precisely, at the first supersonic cell face downstream of sonic lines, F^d is forced to have sonic magnitude $|F^c|$; its direction is solution dependent, c.f. (37b). Note that, at sonic lines, $F^a = 0$, while F^c is the sonic value of the mass-flows vector F. This rigoriously suppresses expansion shocks, because for expansion shocks $|F^d| < |F^c|$ must hold.

c. In subsonic flow regions, the schemes reduce to standard second-order accurate expansions, with densities evaluated at cell-face midpoints.

d. In supersonic flow regions, the upstream shifting of the flux vector F^a in (37c) lowers the accuracy of the scheme to first order and introduces implicit AV (artificial viscosity). This AV may be analyzed by deriving the modified equation, using Taylor series expansions to replace finite differences by partial derivatives. Elegant results follow if streamline coordinates (s,n) and the eigenvalue-eigenvector decompositions (25-28) are introduced. It is then found that (h_m = mesh size)

$$h_m^{-2}(\delta^{\xi}_{00}F^{d1} + \delta^{\eta}_{00}F^{d2}) = \nabla^T F + h_m AVT, \tag{38}$$

$$AVT = (c^{11}\phi_{\xi\xi} + c^{12}\phi_{\xi\eta})_{\xi} + (c^{12}\phi_{\xi\eta}+c^{22}\phi_{\eta\eta})_{\eta} +$$

$$+ \text{lower-order derivatives of } \phi +$$

$$+ \text{higher-order terms in } h_m, \tag{39}$$

where the real coefficients may be arranged in the symmetric negative-definite matrix

$$\begin{bmatrix} c^{11} & c^{12} \\ c^{12} & c^{22} \end{bmatrix} = hH^{-1}V\begin{bmatrix} \rho(1-M^2) & 0 \\ 0 & (\rho q-\rho^* q^*)/q \end{bmatrix} V^T H^{-1} T. \tag{40}$$

(Without restriction, $h > 0$ may be assumed here. Note that $\rho q-\rho^* q^* < 0$.) The formula shows that, for small supersonic Mach numbers, the AVT is of the same kind as that of Jameson [20], because $\rho q - \rho^* q^* = 0(1-M^2)^2$ is then negligable.

e. At shock cell faces, there is usually only one velocity point in the shock in streamline direction. (See for example the figure 5 to be discussed.) To understand why, first observe that, at shocks, $F^c_{\frac{1}{2}0}$ cannot be contaminated by large discretization errors due to the differencing in

$(\nabla\phi)_{\frac{1}{2}0}$ across shock discontinuities; for

$$\frac{U}{q} = \frac{1}{q} G\nabla\phi = \frac{1}{q} H^{-1}VV^{T}H^{-1}T\nabla\phi$$

$$= H^{-1}V\begin{bmatrix}\phi_s/q\\ \phi_n/q\end{bmatrix} = H^{-1}V\begin{bmatrix}1\\ 0\end{bmatrix}; \tag{41}$$

hence, $F^{c}_{\frac{1}{2}0}$ depends only on the direction of the flow and on the mapping (1), but not on the magnitude of velocities. Note also, that in shocks F^{a} in general will have 0(1) approximation errors due to 0(1) errors in $\nabla\phi$ at the single point in the shock. In equation (37d), this point is either the last supersonic upstream cell face $(-\frac{1}{2}0)$, or the first subsonic cell face $(\frac{1}{2}0)$. The tests on ρq in (37d) reject the flux F^{a} that corresponds to the point in the shock: of the two fluxes $F^{a}_{\pm\frac{1}{2}0}$ in (37d), the erroneous flux is characterized by the largest ρq, because the corresponding velocity is closest to M = 1 and ρq has a maximum $\rho^{*}q^{*}$ at M = 1. This thus explaines why the tests in (37d) are structured as shown. Because erroneous fluxes F^{a} at shocks are not used in the nonlinear system of discrete conservation equations this system is free to generate very sharp shocks: there is no need to fit erroneous fluxes into the numerical solution and to balance their errors by viscosity terms.

f. The discrete mass-flux operators (37d) at shocks have as small AV as possible. To see, how AV is avoided, evaluate the term $\delta^{\xi}_{00}F^{d1}$ in the discrete mass-flux equation across the shock:

$$\delta^{\xi}_{00}F^{d1} = (F^{c1}_{\frac{1}{2}0} - F^{c1}_{-\frac{1}{2}0}) + (F^{a1}_{\frac{1}{2}0} - F^{a1}_{-\frac{3}{2}0}). \tag{42}$$

(It has been assumed here that $F^{a}_{-\frac{1}{2}0}$ has been rejected in the test (37d) at the shock.) The term $(F^{c1}_{\frac{1}{2}0} - F^{c1}_{-\frac{1}{2}0})$ gives an $0(h_m^2)$ contribution to h_m*AVT of (39) and gives thus negligable artificial viscosity. The second term $(F^{a1}_{\frac{1}{2}0} - F^{a1}_{-\frac{3}{2}0})$ cannot give rise to second-order differences because the stencils of $F^{a1}_{\frac{1}{2}0}$ and $F^{a1}_{-\frac{3}{2}0}$ do not contain common grid points; it may be shown that, due to this, the analysis for the AVT in the modified equation fails to be valid.

It may be checked that the AVT is not precisely zero over shocks, because the stencils of the upstream and downstream F^{a}-fluxes may still overlap, or have common grid points. From numerical experiments it was found that this remnant AV is not harmful up till now. It may perhaps be completely

eliminated by suitably reducing the stencil sizes of the discrete gradients $(\nabla\phi)_{\frac{1}{2}0}$ in (32) and $(\nabla\phi)_{0\frac{1}{2}}$ and/or changing the upwind-bias rules in (37); this has not been investigated, however.

g. In fact, at shocks the discrete mass-conservation equations reduce to discrete shock relations; e.g., in the case of flow in $+\xi$-direction,

$$F^{a1}_{\frac{1}{2}0} \cong F^{a1}_{-\frac{3}{2}0} + O(h_m) \qquad \text{(discrete shock relation)}$$

$$F^{a1}_{\frac{1}{2}0} = F^{a1}_{\frac{1}{2}0}(M_{\frac{1}{2}0}, (U/q)_{\frac{1}{2}0}),$$

$$F^{a1}_{-\frac{3}{2}0} = F^{a1}_{-\frac{3}{2}0}(M_{-\frac{3}{2}0}, (U/q)_{-\frac{3}{2}0}), \tag{43}$$

$$M_{-\frac{3}{2}0} - M_{\frac{1}{2}0} = O(1) \text{ at shocks.} \tag{44}$$

h. The linear first-variation expressions, corresponding to the nonlinear discrete conservation equations and mass fluxes, have a number of useful properties discussed in section 4; these are used in the iterative solution of the nonlinear equations.

In connection with fast nonlinear iteration, the most important properties of the mass-flux vector-splitting difference schemes are the rigorous suppression of expansion shocks (equation (37b), point b) and the elimination of the stiffness of the discrete conservation equation systems that would result when the erroneous fluxes F^a at shocks would have been used (equation (37d), points d,e). An example of such an undesirable flux is the Engquist-Osher-like shock flux

$$F^d_{\frac{1}{2}0} = F^c_{\frac{1}{2}0} + F^a_{-\frac{1}{2}0} + F^a_{\frac{1}{2}0}. \tag{37d'}$$

The boundary condition (13) on the airfoil image (the grid line $J-\frac{1}{2}$ = constant) is discretized to

$$F^{d2}_{i,J-\frac{1}{2}} = 0, \tag{45}$$

$$F^{2}_{i,J-\frac{1}{2}} = F^{c2}_{i,J-\frac{1}{2}} + F^{a2}_{i,J-\frac{1}{2}} = 0. \tag{46}$$

These two equations overrule at the airfoil image the definitions (37) of the discrete mass-flux vector F^d (modified to account for the changeover

from cell faces ($\frac{1}{2}$0) to cell faces (0$\frac{1}{2}$)). (45) is directly substituted in the discrete mass-conservation equations (i,J-$\frac{1}{2}$) along the airfoil image, c.f. (30); (46) is a second-order central difference equation relating potential values at grid points (i,J) half a mesh distance inside the airfoil to those at grid points (i,J-1) half a mesh distance outside the airfoil.

It may be verified that the discrete boundary conditions are such that only one row of grid points inside the airfoil is required, because these boundary conditions cannot upwind too far into the airfoil in the direction of the η-coordinate.

The free-stream boundary condition (14) is applied at the grid points (i,$\frac{3}{2}$) on a numerical boundary at 4-8 chords from the airfoil:

$$\phi_{i,\frac{3}{2}} = \bar{x}_{i,\frac{3}{2}} + \Gamma\, \bar{\theta}_{i,\frac{3}{2}}. \tag{47}$$

The Kutta condition (18) may be reduced to the condition that, at the image of the trailing edge, the ξ-derivative of the potential is zero,

$$(\phi_{\xi})_{te} = 0, \tag{48}$$

because the mapping (1) will be required to give smooth airfoil images in (ξ,η) coordinates, at the trailing edge, too. This algebraic condition is discretized to

$$\{\tfrac{1}{2}(\phi_{2,J} + \phi_{2,J-1}) - \tfrac{1}{2}(\phi_{1,J} + \phi_{1,J-1})\} = 0, \tag{49}$$

where ($\frac{3}{2}$,J-$\frac{1}{2}$) is the trailing-edge-image point. This discretization is a second-order accurate central difference approximation of Kutta condition (48).

Equations (30-37), (45-49) define a nonlinear finite-difference equation system, to be solved for the unknown grid function $\phi_{i,j}$ and the unknown circulation in (47).

4. SOLUTION ALGORITHM

When a fast-solver algorithm based on multigrid relaxation has to be developed, it is possible to follow Brandt's recommendation [14,17] and use

the elegant FAS (full approximation storage) multigrid strategy. This approach was adopted by nearly every author applying multigrid relaxation in transonic potential-flow calculations, see [1-6,8,10,11,24], for example.

For various reasons, the algorithm presented below is not based on the FAS approach, but on a combination of the linear CS (correction scheme) multigrid relaxation [17,12,14] with Newton iteration. As mentioned in the introduction, this approach allows a separation of the iteration on the non-linearity (by Newton iteration) and the multigrid relaxation on linear equation systems. This iteration involves non-standard issues like shock displacements, and varying Dirichlet boundary conditions by iterative updates of the circulation Γ.

To see in more detail why Newton iteration is useful, the effect of shocks iterating to final converged positions have to be analyzed. It is well-known [12,14] that FAS and CS multigrid relaxation are very fast iteration procedures as long as the basic assumption underlying multigrid processes is valid: on each grid of a grid sequence arranged from fine to coarse, the short-wavelength content of a solution error (short with respect to the mesh size on each individual grid) should be effeciently reduced by a relaxation process that smoothes residuals locally ([14], sect. 13; [12], sect. 1.1). Here local means that, on each grid, large local residuals should give only large local solution corrections; strong (nonlinear) coupling between long- and short-wavelength residuals and solution corrections must thus be absent.

However, the problem in transonic potential-flow calculations is that such strong coupling effects do arise when shocks are iterated over grids to final positions. It was found that shock movements over more than one fine-grid mesh length are usually due to smooth solution corrections (in multigrid procedures efficiently calculated on coarse grids), but at shocks such smooth solution corrections give rise to large residuals on fine grids, creating strong coupling between long-wavelength solution corrections, short-wavelength residuals and short-wavelength solution corrections near shocks. Hence, a basic assumption underlying the multigrid philosophy becomes violated.

The shock movements may be easily seen to lead to inconsistencies in standard FAS algorithms, when the FAS algorithm is not suitably adapted. For example, the derivations of the 2-level versions of the FAS algorithm from the CS algorithm (Brandt, [14], sect. 8.1: from CS to FAS), Stüben

and Trottenberg, [12], sect. 5) are not valid without modifications, when shocks move over the coarser grid, because near shocks this leads to inconsistent calculation rules for the residuals on coarse grids of fine-grid solution-approximations. (More precisely, in the notation of Brandt in [14], sect. 8.1: near shocks, the coarse-grid residual $L^H(I_h^H \tilde{u}_h)$ of a fine-grid solution-approximation $\tilde{u}^h$ is, at the beginning of a coarse-grid correction, different from that at the completion of the coarse-grid correction, if the shock has moved over the coarse grid. This movement changes L^H.)

On the other hand, in the approach followed here, in each Newton linearization step the shock positions are frozen to those corresponding to the current solution estimate ϕ of the potential. Because, in a continuum model, the solution potentials ϕ are continuous over shocks, this means that freezing shocks at erroneous positions in general will lead to (a discretization of) jumps in correction potentials at the frozen shocks, see figure 1 for an illustration of this phenomenon.

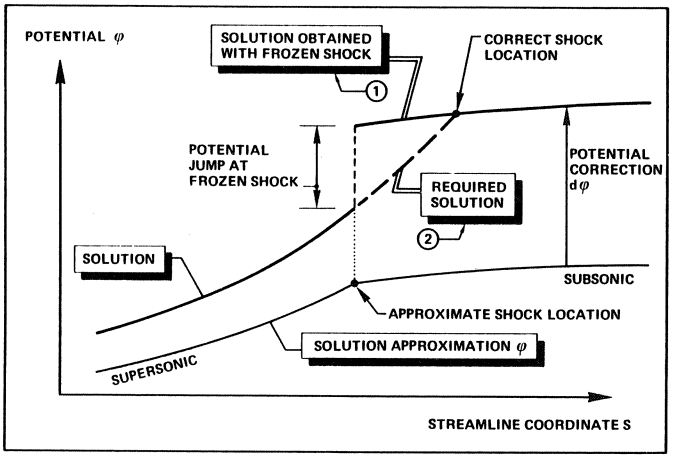


Fig. 1 Effect of shock-freezing in iteration processes

The potential jumps give rise to very large velocity overshoots on fine grids, because velocities become of order $(\text{mesh size})^{-1}$ at frozen shocks; the densities of such velocities may be found to become "negative".

In order to find an efficient and proper way of treating these

potential jumps, it is useful to start with the continuum-model equations and analyze the first-variation formulation of the non-linear equations. In such a classical first-variation formulation, it is natural to introduce first-variation models of shock displacements. When it is necessary to introduce a downstream shock-position displacement to keep the sum (potential + its first variation) continuous at the varied shock (see figure 2), we need the analytic continuation in downstream direction of the smooth potential in the supersonic zone ahead of the shock (see figure 2).

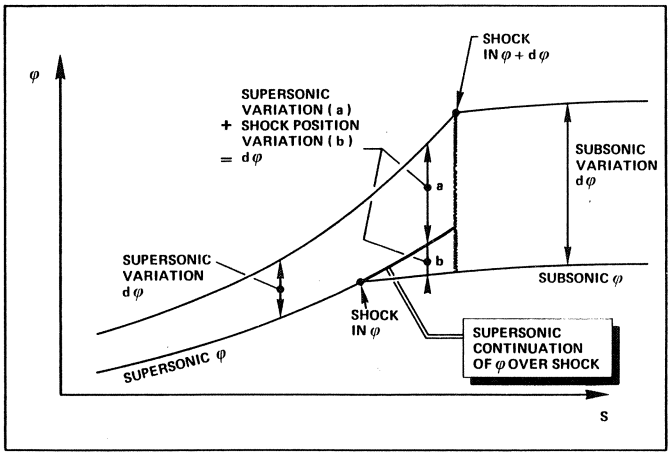


Fig. 2 First-variation analysis for solution approximation φ:
Modelling of shock movements

When a shock-position variation in upstream direction is required, the subsonic potential has to be analytically continued in upstream direction.

The translation of these continuum-model considerations to discrete-model considerations introduces the fine-grid mesh length h_m. When the potential jumps are of $O(h_m)$, the jumps are small; this may be expected to occur in classical line over-relaxation algorithms. However, when the potential jumps are much larger than $O(h_m)$, they can better be interpreted as potential jumps. Such jumps do also arise at certain stages in approximate-factorisation algorithms, see the corresponding velocity overshoots in Ballhauss, [21], figure 2. Severe jumps arise in multigrid processes, because these are able to generate efficiently long-wavelength corrections, see e.g. Boerstoel [7,22] and Jespersen [23] (Euler equations).

To correctly treat these potential jumps, the analytic continuation processes of the continuum-model first-variation analysis have to be simulated numerically. This may be done in various ways. Within the Newton-iteration approach adopted here, two measures have been taken:

a. Each linearization of the discrete nonlinear equations is broken up into two stages:

- -linearization at fixed shock positions and calculation of solution corrections with in general potential jumps at shocks (figure 1, stage 1);
- -simulation of the analytic continuation process by displacing shocks in such a way that discontinuities in (potential + solution correction) at shocks become $\leq O(h_m)$. This continuation is realized by simple linear extrapolation of potentials along the grid lines that are closest to the normals to shocks, according to the dashed line of figure 1, stage 2.

b. The potential jumps in the first-variation equations are explicitly allowed at shocks. Artificial viscosity that would damp these jumps is reduced as much as possible, because this would falsify the jumps: the jumps contain the information how much the shock has to be displaced. The technical means allowing these jumps are contained in the definition (37d) of the nonlinear discrete mass-flux vector at shocks and in the corresponding first-variation discrete mass-flux vector (61d) to be presented below.

When the potential jumps are $O(h_m)$ at most, it is not possible and not necessary to update shock positions by extrapolation, because in this case the shock displacements are about one mesh length at most, and such displacements can be accounted for by linearizations of the shock fluxes (37d).

It is possible that in FAS multigrid algorithms and in approximate-factorisation algorithms similar measures will improve convergence rates. In FAS algorithms, the inconsistences discussed above, should also be eliminated.

A second deviation of the present study from common practice in literature concerns the iteration on the circulation. Usually the circulation is iterated upon during the multigrid relaxation. Here, the iteration on the circulation is brought outside the multigrid relaxation. In fact, the multigrid process has to deal with a Dirichlet boundary condition containing the circulation as an unknown parameter. Standard multigrid processes deal with known boundary conditions, however. So a Newton iteration process for the circulation was designed, with in each Newton iteration step a fixed

Dirichlet boundary condition that can be efficiently dealt with by standard multigrid processes.

The Newton/multigrid iteration process may be considered to consist of three nested loops:

1. outer loop: quasi-Newton iteration on the circulation using the Kutta condition.
2. second loop: Newton iteration on nonlinear flow equations at fixed circulation values.
3. third loop: in each Newton iteration step, solution of a linear first-variation equation system for a correction potential, using multigrid iteration.

1. The outer loop is a quasi-Newton iteration loop to determine the unknown circulation Γ in the far-field approximations (14-17), (47) of the flow. The nonlinear difference equations are rearranged in the form

$$
\begin{aligned}
&\{\phi_\xi(\Gamma)\}_{te} = 0: && \text{discrete Kutta condition,} \\
&L\{\phi(\Gamma)\} = 0: && \text{discrete flow equations and boundary conditions.}
\end{aligned}
\tag{50}
$$

If Γ is a given approximate circulation, a correction $d\Gamma$ may be determined by replacing Γ by $\Gamma + d\Gamma$, linearizing the discrete Kutta condition in $d\Gamma$, and solving for $d\Gamma$ the equation system

$$\{\phi_\xi(\Gamma)\}_{te} + \partial\{\phi_\xi(\Gamma)\}_{te}/\partial\Gamma \, d\Gamma = 0, \tag{51}$$

$$L\{\phi(\Gamma)\} = 0. \tag{52}$$

Equation (51) is used to update Γ. The derivative in (51) is computed with a finite-difference approximation having the form

$$\partial\{\phi_\xi(\Gamma)\}_{te}/\partial\Gamma \cong \frac{\{\phi_\xi(\Gamma_2)\}_{te} - \{\phi_\xi(\Gamma_1)\}_{te}}{\Gamma_2 - \Gamma_1}. \tag{53}$$

In order to keep the quasi-Newton iteration stable, it is necessary to calculate $\{\phi_\xi(\Gamma_i)\}_{te}$ in (53) accurately. So the update is only done when $\{\phi_\xi(\Gamma)\}_{te}$ has been computed to suficient precision using (52). Moreover, (53) is only applied for $|\Gamma_2 - \Gamma_1|$ exceeding a lower limit. In practice, the Newton iteration on Γ converges in two-four steps, starting from an initial

guess precomputed on a coarser grid.

2. The second loop is a Newton iteration to determine, for a fixed given circulation Γ, the solution of the nonlinear equation (52). This iteration process computes $\phi(\Gamma)$ and thus the derivative $\{\phi_\xi(\Gamma)\}_{te}$ needed in (51) to update the circulation Γ.

The linear equation system to be solved in each Newton iteration step is derived by replacing ϕ by $(\phi+d\phi)$, where ϕ is a known solution estimate, and $d\phi$ is a solution correction to be computed, and linearizing $L\{\phi(\Gamma)+d\phi\}=0$ in $d\phi$. The result is formally (the dependence on Γ is not shown):

$$C(\phi)d\phi = -L(\phi). \tag{54}$$

The right-hand side is the residue of ϕ in the nonlinear discrete conservation equations and boundary conditions, and the left-hand side represents a linear equation system for the correction potential $d\phi$. The system may be written out as follows:
discrete first-variation mass-flux equations:

$$\delta^{\xi}_{00}dF^{d1} + \delta^{\eta}_{00}dF^{d2} = -\text{l.h.s. of (30)}, \tag{55}$$

discrete first-variation potential gradients:

$$\left.\begin{aligned}\{\nabla d\phi)_{\frac{1}{2}0} = [&d\phi_{10}-d\phi_{00},\\ &\tfrac{1}{4}(d\phi_{11}+d\phi_{01}-d\phi_{1-1}-d\phi_{0-1})]\end{aligned}\right\}\left\{\begin{aligned}&\text{same stencil}\\ &\text{as for } (\nabla\phi)_{\frac{1}{2}0},\end{aligned}\right. \tag{56}$$

discrete first-variation mass-flux vector:

$$dF^{c}_{\frac{1}{2}0} = P_{\frac{1}{2}0}(\nabla d\phi)_{\frac{1}{2}0}, \tag{57}$$

$$dF^{a}_{\frac{1}{2}0} = C_{\frac{1}{2}0}(\nabla d\phi)_{\frac{1}{2}0}, \tag{58}$$

$$P_{\frac{1}{2}0} = [+(\rho^{*}q^{*}/q)h(G-UU^{T}/q^{2})]_{\frac{1}{2}0}, \tag{59}$$

$$C_{\frac{1}{2}0} = [((\rho q-\rho^{*}q^{*})/q)h(G-UU^{T}/q^{2}) + \rho h(1-M^{2})UU^{T}/q^{2}]_{\frac{1}{2}0}; \tag{60}$$

when $U^{1}_{\frac{1}{2}0} > 0$ (test (36), upstream direction $-\xi$):

subsonic cell face: $M_{\frac{1}{2}0} < 1$ and $M_{-\frac{1}{2}0} < 1$:

(61a) $$dF^{d}_{\frac{1}{2}0} = dF^{c}_{\frac{1}{2}0} + dF^{a}_{\frac{1}{2}0},$$

sonic cell face: $M_{\frac{1}{2}0} \geq 1$ and $M_{-\frac{1}{2}0} < 1$:

(61b) $$dF^{d}_{\frac{1}{2}0} = dF^{c}_{\frac{1}{2}0},$$

supersonic cell face: $M_{\frac{1}{2}0} \geq 1$ and $M_{-\frac{1}{2}0} \geq 1$:

(61c) $$dF^{d}_{\frac{1}{2}0} = dF^{c}_{\frac{1}{2}0} + dF^{a}_{-\frac{1}{2}0},$$ (NB upstream bias)

shock cell face: $M_{\frac{1}{2}0} < 1$ and $M_{-\frac{1}{2}0} \geq 1$:

(61d) $$dF^{d}_{\frac{1}{2}0} = dF^{c}_{\frac{1}{2}0} + \{\underline{\text{if}}\ (\rho q)_{\frac{1}{2}0} > (\rho q)_{-\frac{1}{2}0}\ \underline{\text{then}}\ dF^{a}_{-\frac{1}{2}0}\ \underline{\text{else}}\ dF^{a}_{\frac{1}{2}0}\};$$

discrete first-variation boundary conditions:

at airfoil:

(62) $$dF^{d2}_{i,J-\frac{1}{2}} = -\ \text{l.h.s. of (45)} = 0,$$

(63) $$dF^{2}_{i,J-\frac{1}{2}} = dF^{c2}_{i,J-\frac{1}{2}} + dF^{a2}_{i,J-\frac{1}{2}} = -\ \text{l.h.s. of (46)},$$

at numerical far-field boundary:

(64) $$d\phi_{i,1} = d\Gamma\ \bar{\theta}.$$

It may be observed that all tests selecting the upstream direction and determining the vectors dF^{d} are the same as those for the nonlinear equation system at the current solution estimate ϕ. The coefficient matrices P and C are evaluated at cell-face midpoints, with ρ, q, U evaluated from ϕ as in the nonlinear case, c.f. (32,33). It may be verified that the discrete first-variation fluxes dF^{c}, dF^{a}, dF^{d} are exact linearizations of the corresponding nonlinear discrete fluxes F^{c}, F^{a}, F^{d}. As a consequence, the entire discrete first-variation equation sytem is an exact linearization of the corresponding nonlinear discrete equation system.

The fact that the first-variation difference-equation system is an exact

linearization of the nonlinear discrete system has several pleasant consequences:

a. The Newton iteration has the quadratic termination property.

b. The equation system is fully-conservative, just as the nonlinear system.

c. The equation system is the one needed to verify necessary conditions for stability of the nonlinear difference equation system.

d. The design of relaxation techniques may start from the linear first-variation equations. In particular, at shocks and sonic lines, simplified but still accurate enough relaxation difference molecules may be readily derived from the first-variation conservation equations.

An important technical aid in such detailed studies is the fact that the coefficient matrices P and C used in the first-variation mass-conservation equations (see (55,59,60)) allow the eigenvalue-eigenvector decompositions

$$P = hH^{-1}V\begin{bmatrix}0 & 0\\ 0 & +\rho^{*}q^{*}/q\end{bmatrix}V^{T}H^{-1}T, \tag{65}$$

$$C = hH^{-1}V\begin{bmatrix}\rho(1-M^{2}) & 0\\ 0 & (\rho q-\rho^{*}q^{*})/q\end{bmatrix}V^{T}H^{-1}T, \tag{66}$$

and these show, that (P+C) is positive definite in subsonic flow and that P and -C are positive definite in supersonic flow. For the construction of relaxation schemes, it is useful to know that these matrices are positive definite, because the diagonal elements are then positive and this may be used to construct diagonally dominant iteration matrices for line relaxation.

e. It may be easily verified from (65) and (61d) that the first-variation mass-flux vector dF^{c} is insensitive to large jumps in correction potentials $d\phi$ over shocks. The first-variation mass-flux vectors dF^{a} at shocks, that are rejected in the test of (61d), are not used in the first-variation equation system for the correction potential; jumps in these correction potentials are possible at the corresponding cell faces.

The Newton iteration is terminated as soon as the trailing-edge derivative $\{\phi_{\xi}(\Gamma)\}_{te}$, needed in the circulation-update formula (51), is accurate enough. Moreover, the residues in the discrete nonlinear mass-conservation equations and boundary conditions are also reduced to the order of the truncation errors, when the circulation Γ has become sufficiently accurate.

The Newton iteration is usually terminated in 2-5 steps starting from a solution precomputed on a coarser mesh. Hence, the linear equation system $C(\phi)d\phi = -L(\phi)$ has to be solved usually only 2-5 times.

It is possible to intermix the Newton iterations on Γ and on ϕ and this was actually done; this speeds up convergence. See figure 3 for a scheme of the resulting algorithm.

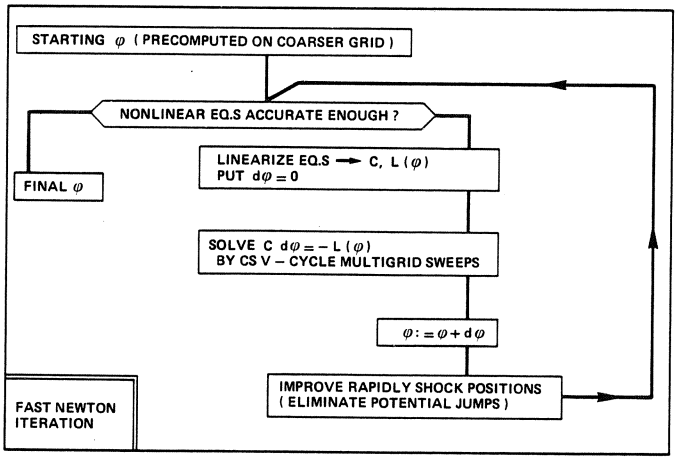


Fig. 3 Newton iteration including shock-position updating using linear extrapolations to eliminate potential jumps

Shock positions are updated by linear extrapolations after completion of each Newton iteration step and before linearizing again. Observe that this procedure prevents premature breakdown of iteration processes because of "negative densities" at shocks.

3. The third iteration loop is the linear CS multigrid relaxation to solve a first-variation equation system $C(\phi)d\phi = -L(\phi)$. This multigrid relaxation involves the three well-known main operations of relaxation smoothing, restriction and prolongation. The general form of the CS algorithm used here follows standard issues of the multigrid literature [12,14,17] and is not repeated here. A few important details should be mentioned, however.

The restriction process does not only involve residuals. It is rather considered to be a process of defining linear fully-conservative coarse-grid difference equations from given linearized fully-conservative fine-grid difference equations. For stability reasons, in supersonic zones, the

coarse-grid equations are required to have the same kind of implicit artificial viscosity as the linearized fine-grid equations have due to the upstream bias generated by equations (61c,d). Consequently, the restriction process involves restriction of residuals and of 2*2 coefficient matrices P and C in the linearized equation system $C(\phi)d\phi = -L(\phi)$, see (54-64). It is then natural to introduce staggered grids, so that four cells of a fine grid coincide with one cell of a coarse grid, see figure 4. This permits simple restriction rules for both residuals and coefficients.

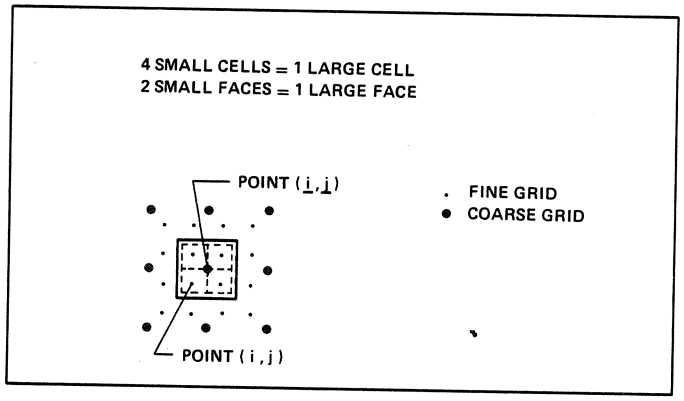


Fig. 4 Staggered grids for simple restriction rules

The restriction rules are, typically (for the relation between $(\underline{i},\underline{j})$ and (i,j), see figure 4; h: fine grid; H: next-coarser grid)

residues of mass-conservation equations in (55) at cell centres $(\underline{i},\underline{j})$:

$$(67) \qquad dR^H_{\underline{i}\,\underline{j}} = \tfrac{1}{4}\{dR^h_{ij} + dR^h_{i+1j} + dR^h_{ij+1} + dR^h_{i+1j+1}\},$$

residues of boundary conditions in (63)
at airfoil point $(\underline{i},\underline{J}-\tfrac{1}{2})$:

$$(68) \qquad dr^H_{\underline{i}} = \tfrac{1}{2}\{dr^h_i + dr^h_{i+1}\},$$

coefficients in mass-flux vectors (57-68)
at cell-face midpoints $(\underline{i}+\tfrac{1}{2},\underline{j})$:

$$(69) \qquad c^{H}_{\underline{i+\frac{1}{2}j}} = \tfrac{1}{2}\{c^{h}_{i+\frac{1}{2}j} + c^{h}_{i+\frac{3}{2}j}\},$$

where c is an element of the coefficient matrices P and C, of the fluxes ρq sign(U^1), or M (needed in the tests in (61) for the first variations dF^d of the discrete mass-flux vectors F^d). The finite-difference equations for correction potentials on the coarser grids are subsequently constructed in the same way as those for the linear first-variation equation system on the finest grid, following (55-64); this construction includes the upstream-bias rules in supersonic zones of equations (61c,d) creating a desired implicit artificial viscosity.

The prolongation of coarse-grid correction potentials for $d\phi$ is based on bilinear interpolation of correction potentials, as usual.

The relaxation is downstream line relaxation, without under- or overrelaxation and with lines along the grid lines of the O-type grid that are roughly normal to the airfoil. Each line relaxation is just a step in block-Gauss-Seidel relaxation of the full (restricted) linear first-variation equations; however, in the coefficient matrix of each line relaxation equation, the off-diagonal elements in P and C have been zeroed; this eliminates the cross-derivatives from the relaxation equations. It is possible that this zeroing is not necessary. It was found that the relaxation equations derived in this way give better convergence, at shocks in particular, than those that are derived in the usual fashion by a pseudo-time dependent analogy with the wave of heat equation.

The line-relaxation equations may be shown to have diagonally dominant coefficient matrices, because of the definiteness properties of the coefficient matrices P and C discussed after equations (65,66).

The multigrid relaxation is based on a fixed V-cycle strategy [12]: starting on a fine grid with a smooth residual distribution, the residuals and coefficients of the first-variation equations are first restricted until the coarsest grid of a grid sequence has been reached. Next, the coarsest-grid equations are solved for a correction solution (in calculations, 4 relaxation sweeps over this grid) and this correction solution is successively prolonged and relaxed (on each grid, by one sweep) until the finest grid has been swept.

The total number of V-cycles needed to solve the linear first-variation equations is of the order of 2 to 6. The iteration is terminated as soon as the maximum norm of the residuals of the linearized equations has been

reduced to the squared maximum norm of the nonlinear equations (use of quadratic termination property). In practice, the quadratic termination property holds only at the end of the calculation, because the algebraic nonlinear equation system changes its form until shocks and sonic lines do not move any more on the finest grid. Due to this effect, it is better to damp the Newton iteration and to terminate V-cycling when the maximum norm of the residuals has been reduced by an order of magnitude (in practice: a factor 20).

5. NUMERICAL EXPERIMENTS

The results of the numerical experiments presented here illustrate the main issues of this paper:

-the fast calculation of transonic potential flows by the Newton/CS-multigrid fast solver,

-the effect of mass-flux-vector splitting on the numerical shock structure,

-the fast calculation of the circulation,

-the fast update of shock positions by linear extrapolations.

Only typical examples of numerical experiments are shown. For a large user-oriented presentation of numerical tests (5 airfoils, among which 3 shock-free NLR airfoils; 14 flow conditions), see Boerstoel-Kassies, [18].

A typical calculation result is the well-known test case NACA0012, M=.75, $\alpha=2^{o}$, figures 5 and 6.

The corresponding 130*42 grid used here is presented in figures 7-10; these figures allow an appreciation of the location of the numerical boundary and of the resolution at the leading and trailing edges.

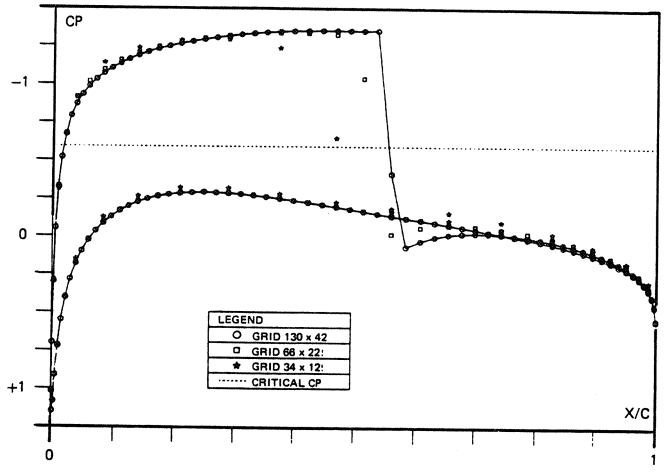


Fig. 5 Pressure distributions on coarse, medium and fine grid on NACA0012 airfoil, M=0.75, $\alpha=2^{o}$

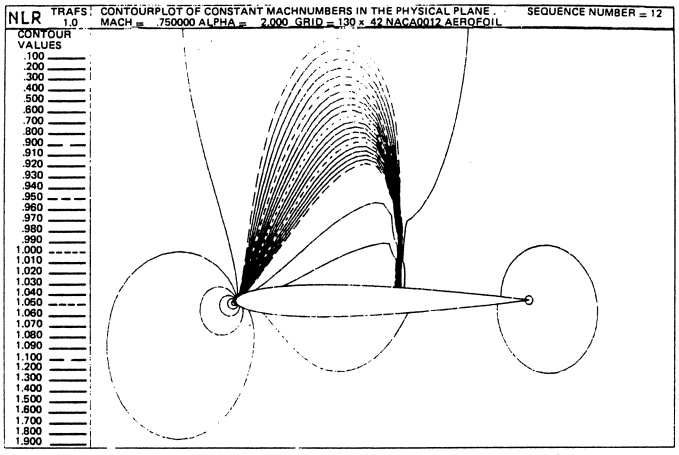


Fig. 6 Enhanced resolution around Mach=1 in M-line isobars plot (NACA0012, M=0.75, $\alpha=2^{\circ}$)

Figure 5 allows an appreciation of the convergence of the pressure distribution with mesh size reduction: the pressure distributions precomputed on a coarse 34*12 grid and on a medium 66*22 grid are also shown. Convergence with mesh size is excellent at the lower subsonic side and ahead of the shock at the upper side.

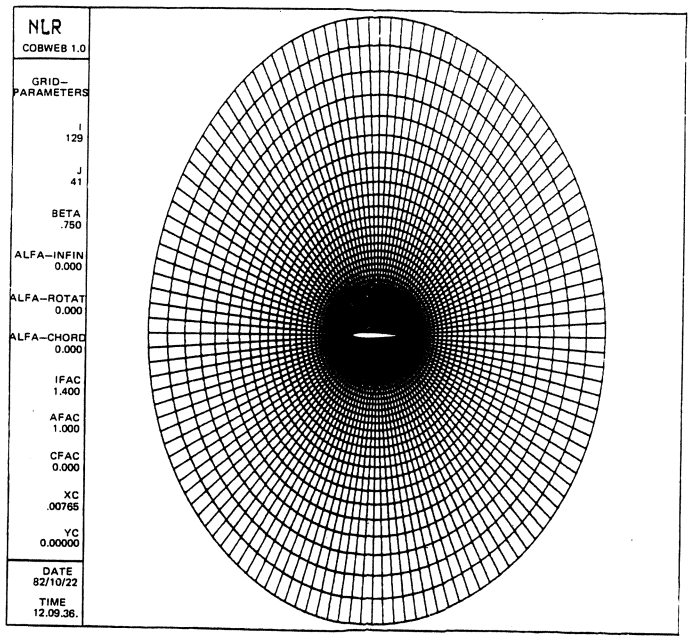


Fig. 7 Far-field part of 130 * 42 grid around NACA0012

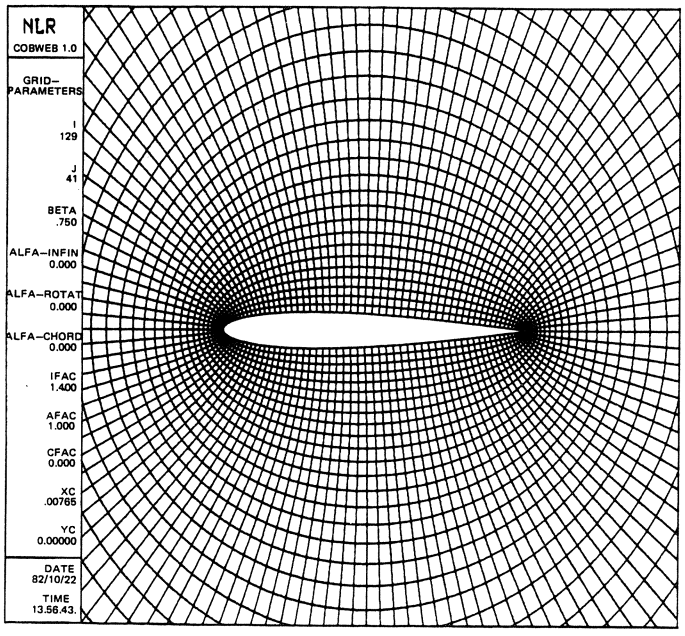


Fig. 8 130 * 42 grid around NACA0012

Convergence at the downstream side of the shock is probably influenced by the well-known Zierep singularity at the downstream foot of the shock.

Figure 5 shows also that, on each grid, there is only one point in the shock and that there are no upstream smearing or wiggle effects due to numerical viscosity or dispersion. Comparison of the shock in the Mach-isobars plots of figure 6 (line density $\Delta M = 10^{-2}$ between M=.9 and M=1.1) with the sizes of the grid cells in figure 8 shows that the numerical shock thickness is one mesh over quite a distance into the flow field.

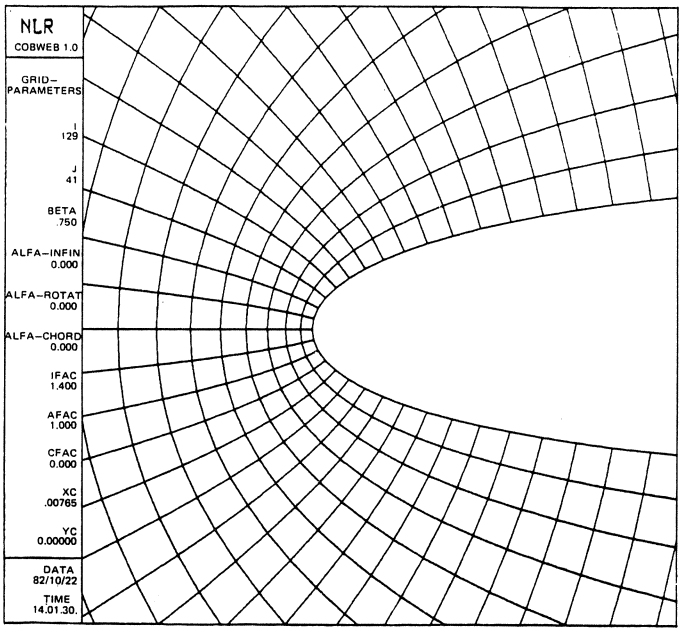


Fig. 9 130 * 42 grid near l.e. of NACA0012

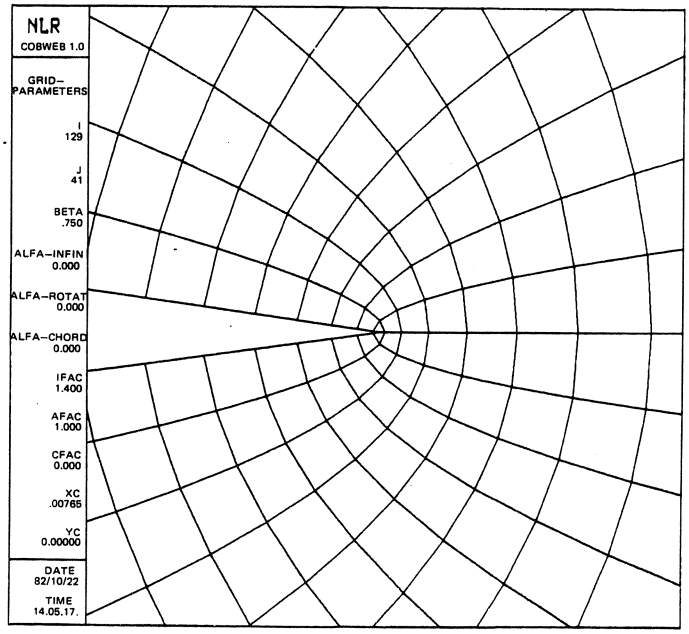


Fig. 10 130 * 42 grid near t.e. of NACA0012

A hard test from the point of view of accuracy was found to be the calculation of sonic lines of the shock-free NLR airfoils in the design condition: the slight kink in the sonic line of the supersonic zone in figure 12 should be absent. Vast numerical experimentation has revealed that this kink is probably due to errors in the velocity field of the order of ±0.003 caused by long-wavelength errors.

A global impression of the amount of computation, required to compute a flow to engineering accuracy, may be obtained from table 1. We first define the various computational-work measures in this table.

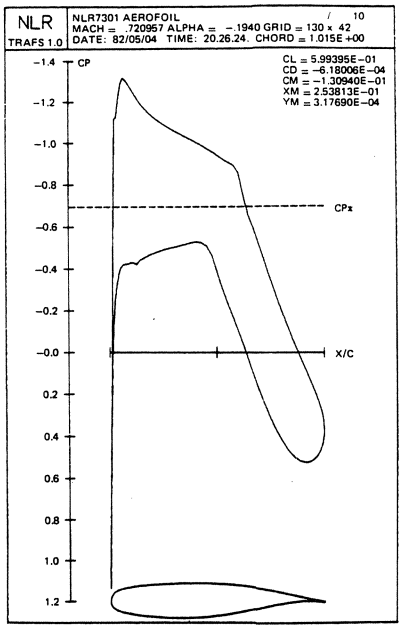


Fig. 11 Pressure distribution on NLR7301 airfoil, M=.721, α=-0.19°(des.cond.)

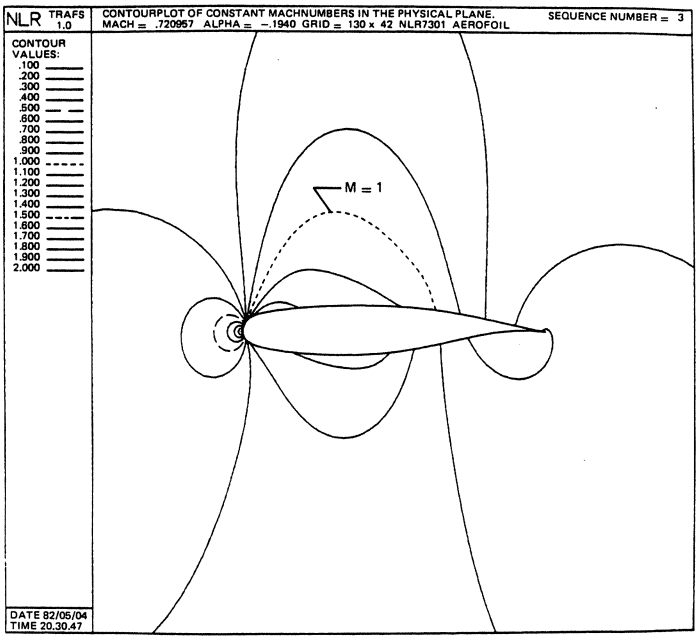


Fig. 12 Sonic-line shape in supercritical shock-free flow around NLR7301 airfoil

-The table is based on the calculation of the 14 flow conditions of [18]. The flow conditions have been divided into three classes: 4 flows without shocks and lift (including one supercritical shock-free NLR airfoil); 5 flows without shocks but with lift (including two supercritical shock-free NLR airfoils) and 5 flows with (moderate to) strong shocks.
-CB in table 1 is the average amount of CP-seconds on the NLR Cyber 170-855 computer required to complete a calculation for the type of flow mentioned. This machine is about two times slower as a CDC7600 computer. Acceptable engineering accuracy is achieved by reducing maximum norms of residuals by a factor of about 10^3. Variations around the average computational-effort values quoted may be up to 30% of the average values.
-WU is the total effort that is needed to complete a calculation in terms of number of work units. Here, one work unit is the amount of work required to perform a nonlinear over-relaxation sweep over the finest 130*42 grid. The amount of CP-time required for such a sweep may be estimated by summing up CP-times for two operations (note that actually nonlinear over-relaxation sweeps are not performed) and is 1.95 CP-sec:
-1.15 CP-sec for one linearization of the non-linear flow equations on the finest grid. This linearization provides the nonlinear residuals and the elements of the relaxation matrix.
-0.8 CP-sec for one line relaxation sweep over the finest grid.
This work unit is comparable to that employed by researchers using nonlinear relaxation.

FLOW CONDITION	n	CP - SEC NLR CYBER 170-855	CP - SEC CDC 7600	WU
NO SHOCKS , NO LIFT	4	15.5	~ 8	8
NO SHOCKS , WITH LIFT	5	24.9	~ 12½	13
WITH SHOCKS , NO / WITH LIFT	5	46.1	~ 23	24

Table 1. Average calculation efforts required to compute flows to engineering accuracy (max. norm of residuals reduced by factor 10^3)

Table 1 leads to various conclusions. Flows without lift and without shocks require only 8 WU. The calculation of lift requires some 50% additional calculation effort; this is quite acceptable. Strong shocks further increase the required amount of computational work to an average of 24 WU. Because in such flows, about 30% of the calculation effort goes into fine-tuning of the shocks, there is room for improvement; this requires redesign of the algorithm near shocks, however.

Comparison of the WU figures with those published in literature shows, that TRAFS outperforms at this moment many other algorithms, compare table 1 to, for example, Raj[1], figs. 10, 12; McCarthy-Reyhner 2, table 2; Pelz-Steinhoff 3, figs. 8, 10, 12; Deconinck-Hirsch 4, figs. 5, 6; Smilovich-Caughey 5, fig. 8a; Jameson 8, figs. 1f, 2b. TRAFS is about 2 to 3 times faster [18] as Jameson's FL06 (based on the use of a fast Poisson solver). Comparison of Holst's very efficient TAIR code (based on an AF fast solver) with TRAFS gives the impression that, in terms of WU, the TRAFS multigrid algorithm is about five times faster as the AF algorithms, compare table 1 with figure 4: for a reduction of the maximum residuals by a factor 10^3, TAIR requires about 33, 70, 83 and 133 iterations = WU, about 5 times more as TRAFS. However, in CP seconds, a TRAFS-WU costs about 10 times more than a TAIR-WU, so that TAIR is about twice as fast in CP seconds. This large difference in CP costs per WU is due to large differences in numerical techniques, and to different coding policies: TAIR is an in-core code, optimized in CP-speed for the CDC-7600 computer by very careful FORTRAN coding. TRAFS is a highly modularizd small-core code, coded for good readiblity and changeability, and it makes vast use of disk storage in inner loops so that central memory cannot limit grid sizes, and thus desired accuracy levels.

In order to give an idea of the convergence speed of the Newton iteration for the circulation Γ by eq.s (51-52), the iteration on Γ of the standard flow of figure 5 and 6 has been listed in table 2: the calculation Γ involves thus very few steps. We found precalculation of Γ on coarser grids necesary for a good efficiency.

GRID LEVEL	$\Gamma = \frac{1}{2} C_\ell$
COARSE	0.1369
	0.1789
	0.2443
	0.2231
	0.2322
MEDIUM	0.2322
	0.2457
	0.2623
	0.2749
FINE	0.2749
	0.2879

Table 2. Convergence history of circulation for NACA0012, M=0.75, $\alpha=2^o$

The effect of removing potential jumps at frozen shocks in a Newton iteration step is illustrated in figures 13-20. These figures show C_p-plots on the airfoil and Mach-isobars plots, at the end of two Newton iteration steps during the calculation of the flow around NACA0012, M=0.85, $\alpha=0$; the fine grid is that of figures 7-10. Figures 13 and 14 show the approximate solution at the end of a Newton iteration step on the medium (66*22) grid, just before shock displacement: the peak Mach number in the velocity overshoot generated by the potential jump exceeds M=1.8 (the desired value is about 1.4). The corresponding figures 15 and 16 show the effect of the subsequent removal of the potential jump by linear extrapolation of supersonic potential values along the grid lines in downstream direction according to the conception of figure 1: the overshoot has disappeared along the entire shock in the flow field. A second fine-grid (130*42) example is shown in figures 17-20. In both cases, the residuals were found to be significant only near the shocks; the rest of the flow field is practically converged.

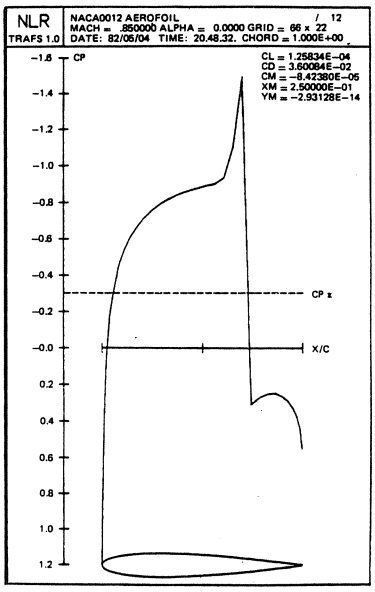


Fig. 13 Velocity overshoot at shock on medium grid encountered during calculation

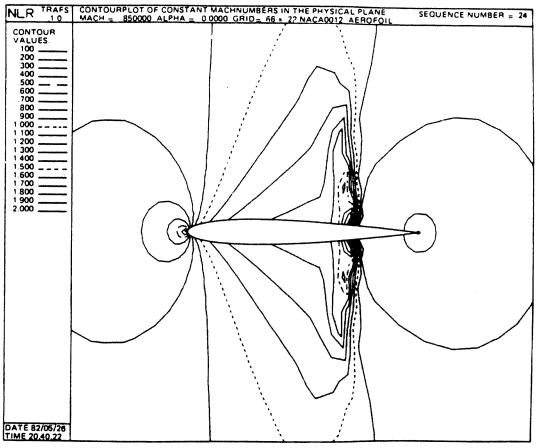


Fig. 14 Velocity overshoots at shock in flow field, encountered during calculation (medium grid)

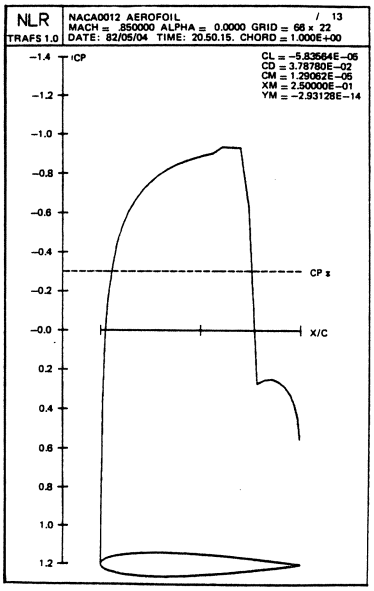


Fig. 15 Effect of shock displacement on velocity overshoot of figure 13 (medium grid)

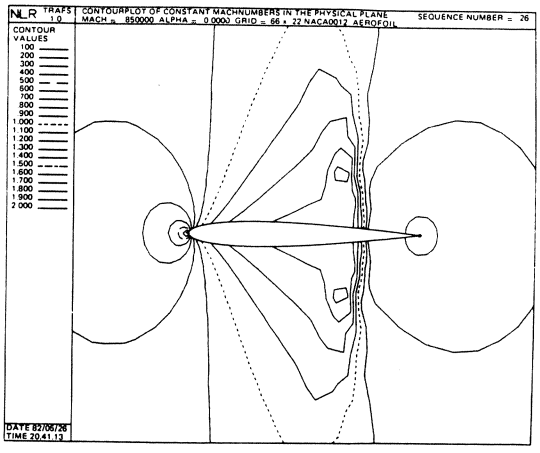


Fig. 16 Effect of shock displacement in flow field on velocity overshoots of figure 14 (medium grid)

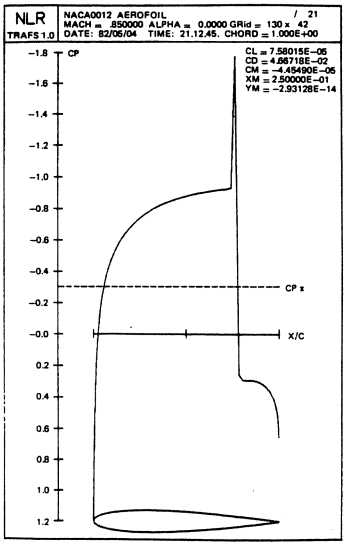


Fig. 17 Velocity overshoot at shock on fine grid encountered during calculation

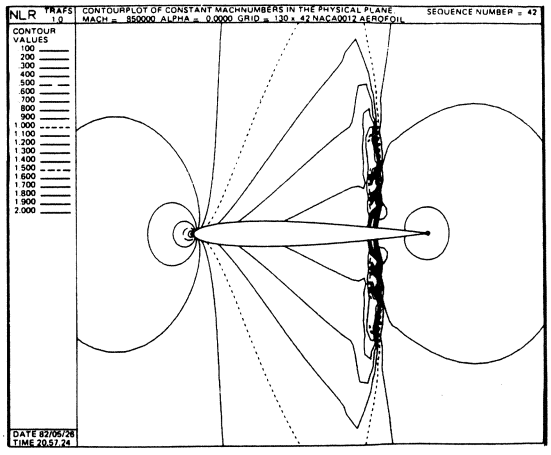


Fig. 18 Velocity overshoots at shock in flow field encountered during calculation (fine grid)

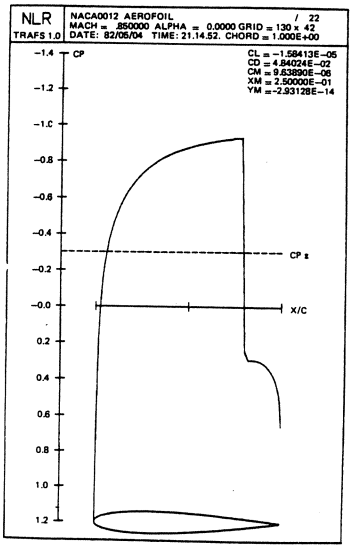


Fig. 19 Effect of shock displacement on velocity overshoot of figure 17 (fine grid)

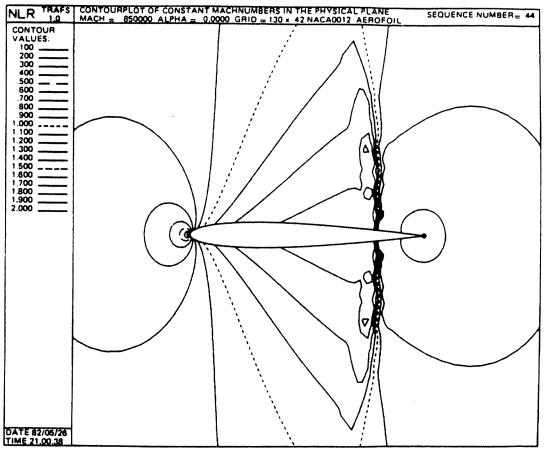


Fig. 20 Effect of shock displacement in flow field on velocity overshoots of figure 18 (fine grid)

6. CONCLUSIONS, FUTURE RESEARCH

The major conclusion from the study is that Newton iteration and CS (correction scheme) multi-grid relaxation may be combined to a robust fast solver for transonic potential flows around lifting airfoils, and that this solver is competitive to alternative fast-solver algorithms (sect. 5, table 1). However, to achieve this, new numerical concepts are required.
The calculation of the circulation in the asymptotic far-field solution was brought outside the multigrid process; the circulation is computed by Newton iteration applied to the Kutta condition (sect. 4, 1, (51-52)). This is a very fast procedure (sect. 5, table 2).
Because the nonlinear equations have to be solved in as few iteration steps as possible, much better nonlinear stability properties of difference schemes are required as when standard nonlinear over-relaxation is applied. In particular, expansion shocks should be rigorously excluded. This was achieved by applying mass-flux-vector splitting (sect. 3, (38b), point b). The fast-solver calculation of shocks requires several new ideas.

-Stiffness of the nonlinear and linearized discrete equation schemes with a length scale of O(shock thickness^{-1}) was avoided by making mass-flux vectors as independent of erroneous numerical potential gradients in shocks as possible (sect. 3, (37d), points e,f; sect. 4, (61d)).

-Fast solvers give rise to potential jumps at shocks during linear stages (here: Newton iteration steps) of iteration processes (sect. 5, figures 13-20; sect. 4, fig. 1). These jumps are due to large long-wavelength solution corrections, which show up as velocity overshoots (fig. 13-20). The potential jumps are a direct measure of shock-position errors (sect. 4, fig. 1).

-At the end of each linear stage of the overall iteration process, the potential jumps are translated into corresponding shock-position corrections by a linear, cheap extrapolation technique (sect. 4, points a,b, fig. 1 stage 2).

-The stiffness just mentioned has to be avoided in order to properly model the potential jumps (sect. 3, points f,d; sect. 4, point e) and the corresponding shock-position corrections.

-Numerical shocks have a thickness of one mesh size (sect. 5, fig. 5-6,8; sect. 3, (37d), points e,f). Numerical viscosity at shocks is deliberately avoided as this would introduce also shock smearing (sect. 3, point f).

Newton iteration was used in this study as a fruitful approach to the design of numerical approximations. The reasons are presented in the first paragraphs of section 4.

The algorithm presented can still be improved in speed. The following posibilities could be explored.

Fine-tuning of moderate to strong shocks requires still too much calculation time (sect. 5, table 1). In order to remedy this, various modifications of the algorithm may be necessary.

a. Replacement of the linear extrapolation technique to remove potential jumps by a quadratic or cubic extrapolation technique.

b. The stencil size of $\nabla\phi$ in (32) and the corresponding $\nabla\phi$ in (56) may be reduced in order to allow complete elimination of erroneous differences at shocks from the discrete equation system, c.f. the discussion in section 3, point f, after (44).

c. The V-cycle convergence-rate should be improved when potential jumps are strong. Possible ways to be investigated: point b just mentioned, improvement of the restriction rules (69) near shocks (at shocks, the averagings (69) may happen to occur in disontinuous functions), and improvement of the prolongation operation near shocks (now, bilinear interpolation in correction potentials; this causes potential-jump smearing).

When performing line relaxation sweeps, TRAFS does not use the fact that, within each Newton iteration step, the coefficient matrix of the linear equation system is fixed, so that the tridiagonal coefficient matrices of the line relaxation sweeps need to be decomposed only once per Newton step. A further reduction in computation effort is possible by applying on the coarsest 16*7 grid Gaussian elimination instead of (four) line-relaxation sweeps.

Another improvement would be the replacement of the line-relaxation process by the LU factorization technique of Van der Wees e.a. [24].

It is estimated that these improvements (or a suitable combination of some of them) may reduce the calculation-effort figure of 24 WU quoted in table 1 for flows with strong shocks to about 15 WU. This expectation has to be confirmed by numerical experiments, however.

7. REFERENCES

[1] RAJ, P., *A multigrid method for transonic wing analysis and design*, AIAA-83-0262 (1983).

[2] McCARTHY, D.R. & T.A. REYHNER, *Multigrid code for three-dimensional transonic potential flow about inlets*, AIAA Journ. 29,1 (1982) 45-50.

[3] PELZ, R.B. & J.S. STEINHOFF, *Multigrid-ADI solution of the transonic full potential equation for airfoils mapped to slits*, *Computers in Flow Predictions and Fluid Dynamic Experiments*, Am. Soc. Mech. Eng.s, Ed. Ghia, K.N. e.a. (Nov. 1981) 27-33.

[4] DECONINCK, H. & C. HIRSCH, *A multigrid method for the transonic full potential equation discretized with finite elements on an arbitrary body fitted mesh*, NASA CP-2202 (1981) pp. 61-81.

[5] SHMILOVICH, A. & D.A. CAUGHEY, *Application of the multigrid method to calculations of transonic potential flow about wing-fuselage combinations*, NASA CP-2202 (1981) pp. 101-130.

[6] BROWN, J.J., *A multigrid mesh-embedding technique for three-dimensional transonic potential flow analysis*, NASA CP-2202 (1981) pp. 131-149.

[7] BOERSTOEL, J.W., *A multigrid algorithm for steady transonic potential flows around aerofoils using Newton iteration*, NASA CP-2202 (1981) pp. 151-172; NLR MP 81050U (1981); Journ. Comp. Phys., 48, 3 (1982) 314-343.

[8] JAMESON, A., *Acceleration of transonic potential flow calculations on arbitrary meshes by the multiple grid method*, AIAA Paper 79-1458 (1979).

[9] ARLINGER, B., *Multigrid technique applied to lifting transonic flow using full potential equation*, SAAB Rep. L-0-1 B439 (1978).

[10] FUCHS, L.J., *Finite difference methods for plane steady inviscid transonic flows*, Rep. TRITA-GAD-2 (1977).

[11] SOUTH, J.C. & A. BRANDT, *Application of a multi-level grid method to transonic flow calculations*, ICASE Rep. 76-8 (1976).

[12] STÜBEN, K. & U. TROTTENBERG, *Multigrid methods: fundamental algorithms, model problem analysis and applications*, in: *Multigrid methods*, Proceedings, Köln-Porz, 1981, ed. W. Hackbush and U. Trottenberg, Springer Verlag (1982) 1-176.

[13] HACKBUSH, W., *Multigrid convergence theory*, in: *Multigrid methods*, Proceedings, Köln-Porz 1981, ed. W. Hackbush and U. Trottenberg, Springer Verlag (1982) 177-219.

[14] BRANDT, A., *Guide to multigrid development*, in: *Multigrid methods*, Proceedings, Köln-Porz 1981, ed. W. Hackbush and U. Trottenberg, Springer Verlag (1982) 220-312.

[15] ———————— , *Multigrid methods*, NASA CP2202 (1981).

[16] HOLST, T.L., *Implicit algorithm for the conservative transonic full-potential equation using an arbitrary mesh*, AIAA Journ. 17, 10 (1979) 1038-1045.

[17] BRANDT, A., *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31, 138 (1977) 333-390.

[18] BOERSTOEL, J.W. & A. KASSIES, *A user-oriented introduction of the calculation and analysis with* TRAFS *of two-dimensional transonic potential flows around aerofoils*, NLR IW-82-017 U (1982).

[19] KASSIES, A., *User's guide of program* TRAFS: *Transonic aerofoil flow-calculation system*, NLR IN-82-008 (1982).

[20] JAMESON, A., *Transonic potential flow calculations using conservation form*, Proc. AIAA 2nd Comp. Fl. Dyn. Conf. (1975) 148-174.

[21] BALLHAUS, W.F., *A fast implicit solution procedure for transonic flows.*

[22] BOERSTOEL, J.W., *Numerical modelling and fast-solver calculation of approximately normal shocks*, NLR MP 82026 U (1982).

[23] JESPERSEN, D.C., *A multigrid method for the Euler equations*, AIAA-83-0124 (1983).

[24] WEES, v.d. A.J., J. v.d. VOOREN & J.H. MEELKER, *Robust calculation of 3D transonic potential flow based on the nonlinear FAS multigrid method and incomplete LU decomposition*, AIAA 83-1950 (1983).

MODELLING OF A TRANSPORT PROBLEM IN PLASMA PHYSICS

B.J. BRAAMS

A two-dimensional, two-fluid model for transport processes in the edge plasma in a tokamak device is presented, and the numerical methods employed for its solution are described. Emphasis is given to those aspects of the numerical treatment which are generally relevant to the solution of coupled systems of convection–conduction equations.

INTRODUCTION

An important problem in plasma physics for nuclear fusion is how to extract from the magnetic confinement device the large amounts of energy and alpha particles which are produced in the inner plasma. In the interior of the toroidal confinement region the main transport process is diffusion across magnetic surfaces, while near the edge it is flow along those field lines that intersect a material boundary [**1**]. Untill recently these two processes were only modelled separately, with sophisticated codes for the one-dimensional radial transport problem, and fairly crude models for the one-dimensional flow along the field lines.

The subject of this presentation is the two-dimensional two-fluid modelling of the edge plasma region [**2**], [**3**], where both the radial diffusion of particles and energy and the convection and conduction along field lines are important processes. A two-fluid model is employed because the electrons and the ions have separate (but coupled) energy balances. The code that has been developed for these studies is based on a finite-volume discretization of the conservation equations on a topologically rectangular mesh, using methods of D.B. Spalding's school [**4**]. The discretization is fully implicit in time with the aid of an elliptic pressure correction procedure. The discrete coefficients are a continuous function of the local cell Péclet number, with central differencing and pure convective upwind differencing as the appropriate limits.

The equations are solved by a method based on incomplete L*U decomposition.

The paper is organized as follows. Chapter 1 presents a simplified version of the governing differential equations of our model. The simplified set exhibits all the essential features which are relevant to the choice of the numerical methods employed for the solution of the full set, while omitting most of the details. The full set of equations is given in appendix A. In chapter 2 the numerical methods are presented. We review the semi-implicit procedure of Patankar and Spalding, in which the continuity equation is replaced by an elliptic pressure correction equation, and their hybrid (central–upwind) discretization scheme for convection–conduction equations. Also discussed is our approach to the special problem of the coupling between the electron and the ion energy equations. In chapter 3 some calculations are shown.

The need for numerical solutions to coupled systems of convection–conduction equations arises in many branches of physics and engineering. The present paper is meant to be self-contained for an audience with an interest in computational fluid dynamics, but not neccesarily interested in the plasma physics context of our work. The governing equations are presented with a minimum of physical justification, and the generally relevant aspects of the numerical treatment are emphasized. Those who have a specific interest in the edge plasma modelling may consider this paper as a companion to [2] and [3].

1. A Two-Dimensional Model of the Edge Plasma

Physical description. In a magnetized plasma the charged particles are constrained, in lowest approximation, to follow helical orbits 'tied' to a magnetic field line. Parallel and perpendicular[1] transport processes therefore differ in character. Parallel transport of particles is governed by a momentum balance equation, and perpendicular transport by a diffusion equation. Energy transport is described by a convection–conduction equation in both directions, but the parallel conductivities are much larger than the perpendicular ones. In a magnetic confinement configuration of the tokamak type each magnetic field line is generally constrained to lie on a toroidal 'flux' surface, and it may be assumed that the density, the electron temperature and the ion temperature are each constant over a flux surface. The transport equations may then be reduced to a one-dimensional (perpendicular to the flux surfaces) set.

[1] The adjective 'parallel' refers to a component along the magnetic field, and 'perpendicular' refers to a component which is transverse to the toroidal flux surfaces.

The assumption of uniformity along field lines breaks down in the edge region of the plasma, where the field lines intersect material boundaries. In an axisymmetric toroidal device the transport in this region requires a two-dimensional description.

Simplified mathematical model. An appropriate set of equations is made up of a continuity equation for the ion density n_i, a momentum balance equation governing the parallel velocity u, a diffusion equation for the perpendicular velocity v, and convection–conduction equations for the electron- and ion temperatures, T_e and T_i (expressed in energy units). The full set as solved in our code is given in Appendix A. Below, a simplified version is given, which will serve in the next chapter as a basis for the discussion of the numerical methods. Auxiliary physical quantities appearing in the equations are the electron density, $n_e = Z_i n_i$, the mass density, $\rho = m_i n_i$, and the pressure, $p = n_e T_e + n_i T_i$. The coordinates x and y correspond to the parallel and perpendicular directions respectively. The simplified equations are:

$$\text{(1)} \qquad \frac{\partial}{\partial t} n_i + \frac{\partial}{\partial x}(n_i u) + \frac{\partial}{\partial y}(n_i v) = S_n$$

$$\text{(2)} \qquad \frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}\left(\rho u^2 - \eta_x^i \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\rho v u - \eta_y^i \frac{\partial u}{\partial y}\right) = S_{mu} - \frac{\partial p}{\partial x}$$

$$\text{(3)} \qquad v = -D \frac{\partial}{\partial y}(\ln n_i)$$

$$\text{(4)} \qquad \frac{\partial}{\partial t}\left(\frac{3}{2} n_e T_e\right) + \frac{\partial}{\partial x}\left(\frac{5}{2} n_e u T_e - \kappa_x^e \frac{\partial T_e}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{5}{2} n_e v T_e - \kappa_y^e \frac{\partial T_e}{\partial y}\right) = S_E^e - K_{ei}(T_e - T_i)$$

$$\text{(5)} \qquad \frac{\partial}{\partial t}\left(\frac{3}{2} n_i T_i + \frac{1}{2}\rho u^2\right) + \frac{\partial}{\partial x}\left(\frac{5}{2} n_i u T_i + \frac{1}{2}\rho u u^2 - \kappa_x^i \frac{\partial T_i}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{5}{2} n_i v T_i + \frac{1}{2}\rho v u^2 - \kappa_y^i \frac{\partial T_i}{\partial y}\right) = S_E^i + K_{ei}(T_e - T_i)$$

S_n, S_{mu}, S_E^e and S_E^i are volume sources of ions, momentum, electron and ion energy. η_x^i and η_y^i are the parallel and perpendicular ion viscosities, $\kappa_x^{e,i}$ and $\kappa_y^{e,i}$ are thermal conductivities, D is the diffusivity, and K_{ei} is the energy equipartition coefficient. The source terms are complicated non-local functions of the sought solution, involving models for particle ionization and recombination, and for radiation energy loss. The transport coefficients are nonlinear local functions of the solution.

In this simplified set of equations the distinction between the parallel direction and the poloidal direction has been ignored, the equations have been written in a cartesian coordinate system instead of in a curvilinear orthogonal system, and some contributions have been left out of the energy equations. These aspects do not influence the numerical procedure in a significant way.

Boundary conditions. Counting derivatives, one sees that a total of seven conditions is required on the boundaries perpendicular to the x-coordinate; eight conditions are required on boundaries perpendicular to the y-coordinate. On each segment of the boundary there will be two conditions related to the energy equations, (4) and (5), specifying either an energy flux, or a temperature, or more generally specifying the energy fluxes in terms of the temperatures and density. For the parallel momentum equation we usually have a sonic flow specification on one face, and zero flow or zero shear elsewhere. For the continuity equation and the diffusion equation together, on one face perpendicular to the x-direction and on both faces perpendicular to the y-direction either the density, or the particle flux, or some combination of the two may be specified.

2. Numerical Treatment

Outline. Following [4] it was decided to employ a finite-volume spatial discretization on a staggered mesh, and a fully implicit discretization in time. Interest is in fact restricted to steady state solutions. The continuity equation is treated by the Patankar–Spalding implicit method [5], through which it is replaced by a pressure correction equation of standard convection–conduction form. The discrete coefficients for each of the convection–conduction equations are computed using the power law scheme of Patankar [4]; this is formally second-order accurate, and is stable at all values of the cell Péclet (or Reynolds) number. The resulting five-point equations are relaxed separately in a cyclic order; the Strongly Implicit Procedure (SIP) of Stone [6] is employed. The strong coupling between the two energy equations is eliminated by relaxing in turn the total energy and the ion energy balance. The following subsections present more details on the numerical methods.

Pressure correction procedure. The need for a special treatment of the continuity equation may be seen most clearly by consideration of incompressible flow. If one would consider the momentum equation (or, in our case, one component of the momentum equation, supplemented by a diffusion equation) to govern the velocity field, and the energy equation(s) to govern the temperature(s), then the continuity equation would have to govern the pressure. But the pressure does not even appear in that equation.

For compressible flow, the pressure is a derived quantity, and the density is one of the primary variables. The continuity equation as it stands then appears suitable for relaxation of the density field, but severe problems appear with low Mach number flows, when the fluid is effectively incompressible. The traditional prescription (e.g. [7]), is to employ an explicit discretization in time for the continuity equation, regardless of the treatment of the other equations in the system, with a time step governed by the CFL condition based on the velocity of sound.

Patankar and Spalding's method is to satisfy the continuity equation through simultaneous changes to the density, pressure, and velocity fields. We present the method here with reference to a steady-state equation of the form $\partial(nu)/\partial x + \partial(nv)/\partial y = S_n$, noting that it is equally well applicable to an implicit treatment of a time-dependent equation. At each iteration on the continuity equation the following coupled adjustments are made:

$$\text{(6)} \qquad \begin{cases} p := p + \xi \\ n := n + \kappa\xi \\ u := u - c_x \dfrac{\partial \xi}{\partial x} \\ v := v - c_y \dfrac{\partial \xi}{\partial y} \end{cases}$$

Plugging these changes into the continuity equation, and neglecting terms quadratic in ξ, one sees that ξ is to be obtained as solution to a standard convection–conduction equation:

$$\text{(7)} \qquad \frac{\partial}{\partial x}(\kappa u\xi - nc_x \frac{\partial \xi}{\partial x}) + \frac{\partial}{\partial y}(\kappa v\xi - nc_y \frac{\partial \xi}{\partial y}) = r$$

where r is the current residual, $r = S_n - \mathbf{div}\,(n\mathbf{u})$.

The coefficients κ, c_x, and c_y are choosen so as to minimize the damage that Eq. (6) does to the equation of state and to the equations governing the velocities. Obviously $\kappa := (\partial n/\partial p)_T$ is appropriate. The choice of the coefficients c_x and c_y is more difficult, and requires consideration of the discretized momentum equations. The discrete equation for the x-velocity u has the form

$$A \cdot u = S_{mu} - \frac{\partial p}{\partial x} \tag{8}$$

We assume that A is a diagonally dominant operator of five-point form, and that everyting which does not fit into A has been moved into the right hand side. The prescription of Patankar and Spalding for the coefficient c_x is now to set $c_x := 1/\alpha$ at each point, where α is the diagonal coefficient in the matrix A. In this way the adjustment $u := u - c_x \partial\xi/\partial x$ approximately cancels the effect of $p := p + \xi$ in the x-component of the momentum equation. With the usual fluid flow problems the prescription for c_y is similar, but in the system (1)–(5) the y-velocity is governed by a diffusion equation, leading immediately to the assignment $c_y := \kappa D/n$.

This implicit pressure correction procedure is also discussed by Brandt, e.g. [8], in a more general setting. There the method is referred to as 'distributive relaxation', and is recommended generally as a relaxation procedure for systems of equations which are not separately elliptic: the Cauchy–Riemann system, compressible and incompressible Navier–Stokes, and the Euler equations.

Spatial discretization. This section deals with the discretization scheme for a convection–conduction equation in conservation form:

$$\mathcal{L}\phi \equiv \mathbf{div}\,(\rho \mathbf{u}\phi - \Gamma \cdot \mathbf{grad}\,\phi) = S \tag{9}$$

The differential operator is expanded on coordinates, and it is assumed that Γ is then diagonal, so that

$$\mathcal{L}\phi = \frac{1}{\sqrt{g}} \sum_{\alpha} \frac{\partial}{\partial x_\alpha}\left(\frac{\sqrt{g}}{h_\alpha}\left(\rho u_\alpha \phi - \frac{\gamma_\alpha}{h_\alpha}\frac{\partial \phi}{\partial x_\alpha}\right)\right) \tag{10}$$

Let us deal with the three-dimensional case. The region is divided into rectangular cells (control volumes), with ϕ discretized at cell centers. Consider an interior mesh point P, and integrate the differential equation over the control volume surrounding P. Denote the neighbours of P by E, W, N, S, T, B (east, west, north, south, top, bottom), and the corresponding cell faces by subscripts e, w, n, s, t, b. The volume integral is expressed as a sum of six surface integrals, e.g. for the 'east' face:

$$J_e = \iint (\rho u_1 \phi - \frac{\gamma_1}{h_1} \frac{\partial \phi}{\partial x_1}) h_2 h_3 \, dx_2 dx_3 \tag{11}$$

This expression will be approximated by

$$J_e \simeq \beta \phi_E - \alpha \phi_P \tag{12}$$

where the coefficients α and β depend on the strength of flow F_e and the conductance D_e:

$$\begin{aligned} F_e &= \iint \rho u_1 h_2 h_3 \, dx_2 dx_3 \\ D_e &= \frac{1}{d_e} \iint \gamma_1 h_2 h_3 \, dx_2 dx_3 \end{aligned} \tag{13}$$

where d_e is the distance between points P and E.

In the context of a discretization scheme one requires only approximate values of d_e, F_e, and D_e. So let A_e denote an approximate area of the cell face, and let h_1, ρu_1, and γ_1 stand for some local average of the corresponding continuous quantity, then $d_e \simeq h_1(x_1(E) - x_1(P))$, $F_e \simeq \rho u_1 A_e$, and $D_e \simeq \gamma_1 A_e / d_e$.

Two often used discretization schemes are the central difference scheme, which employs

$$\alpha = -D_e - F_e/2, \qquad \beta = -D_e + F_e/2 \tag{14}$$

and the upwind scheme, for which

$$\alpha = -D_e - \max(F_e, 0), \qquad \beta = -D_e + \min(F_e, 0) \tag{15}$$

The central difference scheme is second-order accurate, but unstable at high cell Péclet number, $P_e = F_e/D_e$, whereas the upwind scheme is always stable, but only first-order accurate.

Through consideration of the exact solution to the one–dimensional convection–conduction equation with constant coefficients, Patankar [4] is led to define two intermediate schemes, both of which approximate central differencing at low cell Péclet number, and upwind differencing with zero diffusion at high cell P. These are the piecewise linear scheme,

$$\alpha = \left\{ \begin{array}{l} 0 \\ -D_e - F_e/2 \\ -F_e \end{array} \right\} \quad \text{if} \quad \left\{ \begin{array}{l} F_e/2 \le -D_e \\ |F_e/2| < D_e \\ F_e/2 \ge D_e \end{array} \right\} \tag{16}$$

and the power law scheme,

$$\alpha = \left\{ \begin{array}{l} 0 \\ -D'_e - \max(F_e, 0) \\ -F_e \end{array} \right\} \quad \text{if} \quad \left\{ \begin{array}{l} F_e/10 \le -D_e \\ |F_e/10| < D_e \\ F_e/10 \ge D_e \end{array} \right\} \tag{17}$$

where $D'_e = D_e\,(1 - |P_e|/10)^5$. For all schemes, $\beta = \alpha + F_e$ and $\beta(D_e, F_e) = \alpha(D_e, -F_e)$

The power law scheme is employed in our code, although the piecewise linear scheme would serve just as well.

Relaxation procedure. In order to obtain a steady solution to the system of equations (1)–(5), a procedure is employed in which each equation is relaxed in turn, in a cyclic order until convergence is achieved. (For reasons to be discussed later, instead of (5) we employ the total energy equation, (4) + (5).) Time-stepping is employed, but only to obtain some under-relaxation; the discretization is fully implicit, and within any single timestep the equations are not relaxed to convergence. Each cycle consists of the following actions:

(a) The source terms S_n, S_{mu}, and $S_E^{e,i}$ are computed.

(b) The momentum balance equation (2) is relaxed by changes to the field u. Also the coefficient c_x is computed for each point in the mesh.

(c) v is adjusted to satisfy the diffusion equation (3), and the coefficients c_y are computed.

(d) The continuity equation (1) is relaxed and the equation of state is satisfied through simultaneous changes to n_i, u, v, and p.

(e) The ion energy equation (5) is relaxed by changes to the field T_i; the pressure is adjusted accordingly.

(f) The total energy equation, (4) + (5), is relaxed by changes to the field T_e; the pressure is adjusted accordingly.

Each of the five-point equations is relaxed by means of one or two iterations of the Strongly Implicit Procedure of Stone [6], as implemented in the NAG library code D03UAF [9]. The residuals of all equations are monitored in order to decide whether a converged solution has been achieved.

A special complication in the system (1)–(5) is the presence of two energy equations, which, due to the term $K_{ei}(T_e - T_i)$, may be strongly coupled, at least over part of the domain. To relax these equations separately would lead to very slow convergence. (Analogous problems occur in the modelling of chemically reacting flow). Our treatment, employing the sum equation and the ion-energy equation, eliminates this problem. An alternative is to employ the sum equation and the difference, but that is less satisfactory in the common case where the electron energy transport is dominant; then the sum and difference equations become strongly coupled.

3. An Example Calculation

Some early calculations with the code were presented in [3]. More recently the code was used, in collaboration with C.E. Singer, to provide predictions for the edge plasma in the proposed TFCX experiment at Princeton. Those studies are summarized in [10]. Shown here are the results of one particular calculation; the 'baseline case' of [10]. The present discussion refers to the equations as given in the appendix.

Geometry and boundary conditions. Fig. 1 shows a poloidal cross-section of the TFCX design, with the domain of the calculation indicated. This region is mapped to the rectangular mesh as shown in Fig. 2. The size of this region is 4.0 m × 0.2 m, divided into 20 × 16 cells. It is to be noted that, although the parallel length scale is much larger than the perpendicular scale, also parallel transport coefficients are much larger than perpendicular ones, so the problem really is two-dimensional. The metric coefficients $\sqrt{g}$, h_x and h_y are constants in this calculation; $B_\theta/B = 0.2$.

The boundary conditions are the following:

– On the 'south' edge (the interface with the bulk plasma), the ion density and the parallel flow velocity were prescribed, $n_i = 10^{20}/\mathrm{m}^3$ and $u_\parallel = 0$. The conditions for the energy equations were $T_e = T_i = T_{bulk}$, but T_{bulk} was to be found as part of the overall solution process in order to obtain a prescribed average energy flux into the boundary plasma of 135 kW/m^2.

– On the 'north' edge (the outer wall), we prescribed zero transverse particle flux, zero shear, and low values (2 eV) for T_e and T_i. (Note: in [10] actually, instead of zero particle flux, a fixed low value for the density was specified. Either condition is arbitrary to some extent).

– On the 'west' edge (the midplane), symmetry conditions were specified: zero parallel flow and zero parallel gradients of n_i, T_e and T_i.

– On the 'southern' half of the 'east' edge, again symmetry conditions were specified (but the density gradient was not fixed; we are only allowed three conditions here).

– On the 'northern' half of the 'east' edge (the limiter), sonic flow is required: $u_\parallel = \sqrt{p/\rho}$. Furthermore the energy fluxes are specified in the following form:

$$
\begin{aligned}
Q_e &= \delta_e n_e u T_e \\
Q_i &= \delta_i n_i u T_i + \frac{1}{2}\rho u u_\parallel^2
\end{aligned}
\tag{18}
$$

with $\delta_e = 4.0$ and $\delta_i = 2.5$; essentially this condition specifies the ratio between conducted and convected energy transport.

Model for atomic processes. The physical picture here [1] is that plasma striking the limiter is almost all released as neutral particles and re-ionized somewhere in the plasma volume, mainly very close to the limiter. (There is therefore almost no net plasma flow into the boundary region). An energy loss is associated with this re-ionization.

A proper treatment of these processes is beyond the scope of the present code; a Monte Carlo neutrals routine would be required. Instead we employ a crude but reasonable model for the recycling process, which conserves particles and approximates the proper energy loss terms. It is not neccessary to present the details here.

Transport coefficients. The coefficients $\eta_\parallel^i$, $\kappa_\parallel^e$, $\kappa_\parallel^i$ and K_{ei} are assumed classical. The radial transport coefficients were assigned anomalous values: $D = 2.5\,\mathrm{m}^2/\mathrm{s}$, $\eta_y^i/\rho = 0.2\,\mathrm{m}^2/\mathrm{s}$, $\kappa_y^e/n_e = 5\,\mathrm{m}^2/\mathrm{s}$ and $\kappa_y^i/n_i = 0.2\,\mathrm{m}^2/\mathrm{s}$.

Results. The outcome of this calculation is displayed in the contour plots, figs. 3–5.

Fig. 3 shows the density field n_i. In the narrow recycling zone in front of the limiter the density rises by a factor of six over the prescribed value at the interface with the bulk plasma, to a value $6 \times 10^{20}/\mathrm{m}^3$.

Fig. 4 shows the electron temperature T_e. Outside the recycling zone the strong parallel conductivity causes the temperature to be nearly constant along the fieldlines, but in front of the limiter T_e falls off rapidly. The expected temperature near material surfaces is a major engineering concern for the design of this type of experiment.

In fig. 5 the mach number of the parallel flow velocity is displayed. The direction of flow is to the left, away from the limiter, in the lower part of the domain, and towards the limiter in the upper part. The interest of this result is to bring out that the very localized recycling process does have a global influence on the flow field.

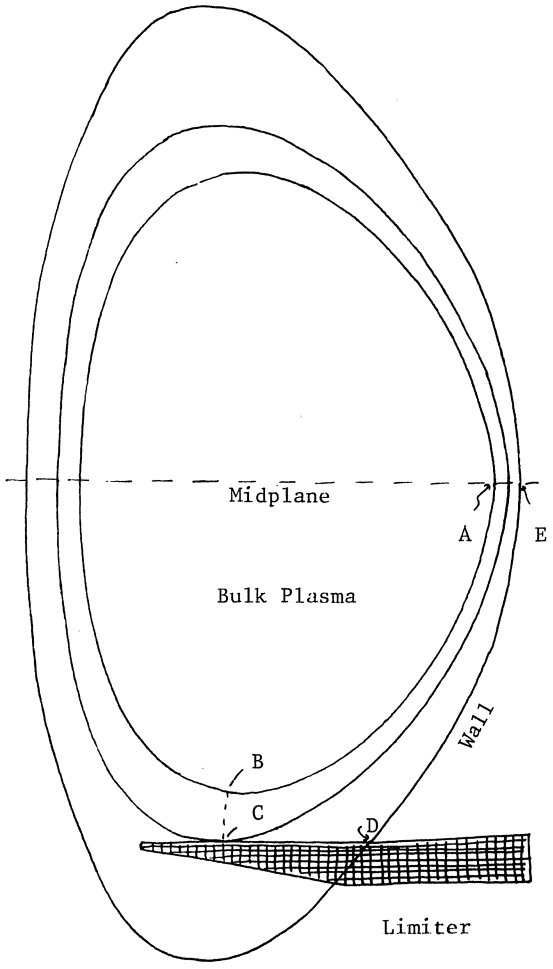


Fig. 1.

Poloidal cross-section of the proposed TFCX experiment.
Points A-E show the boundary of the computational region.

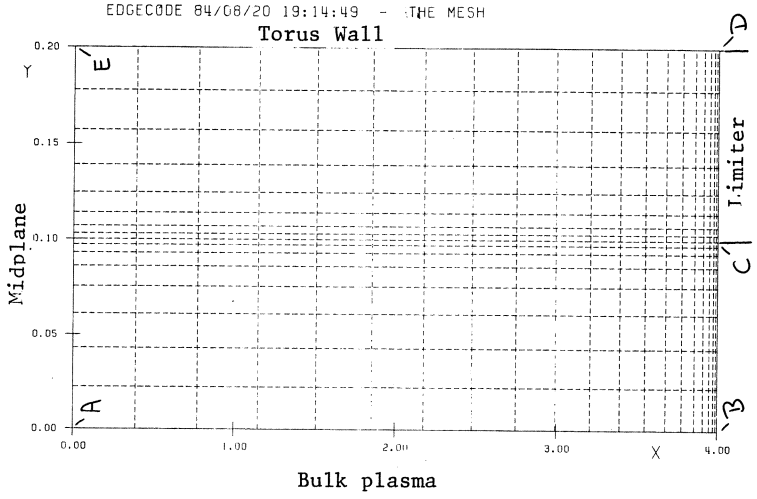


Fig. 2.

The computational mesh.

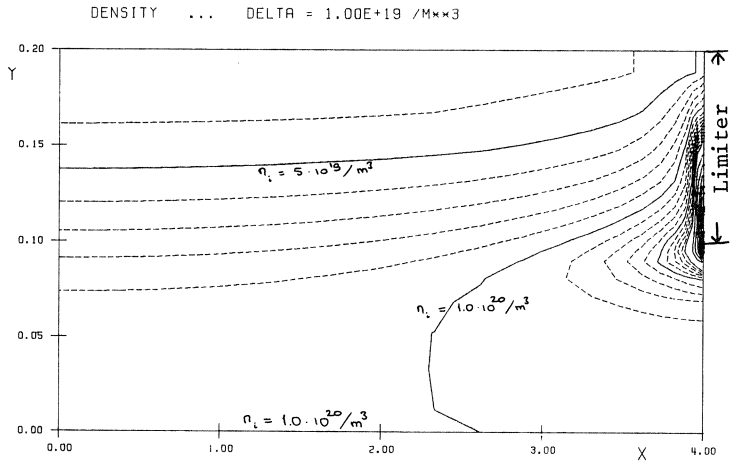


Fig. 3.

Contour plot of the ion density. The increment between the dotted contours is $1.10^{19}/m^3$; between the solid contours it is $5.10^{19}/m^3$.

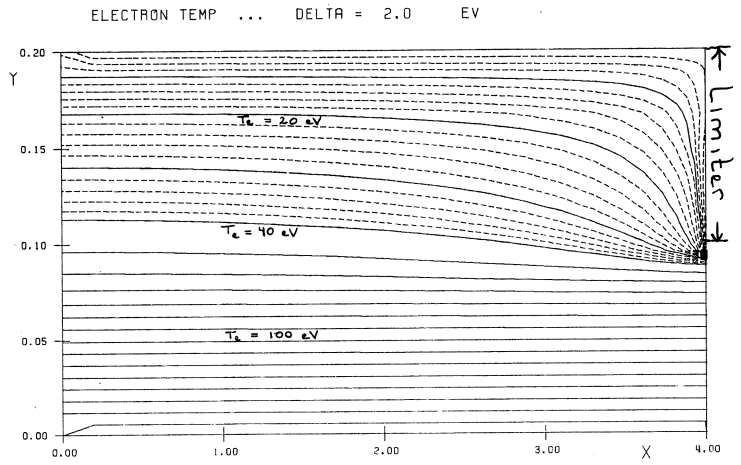


Fig. 4.

Contour plot of the electron temperature. The increments are 2 eV between the dotted contours and 10 eV between the solid contours.

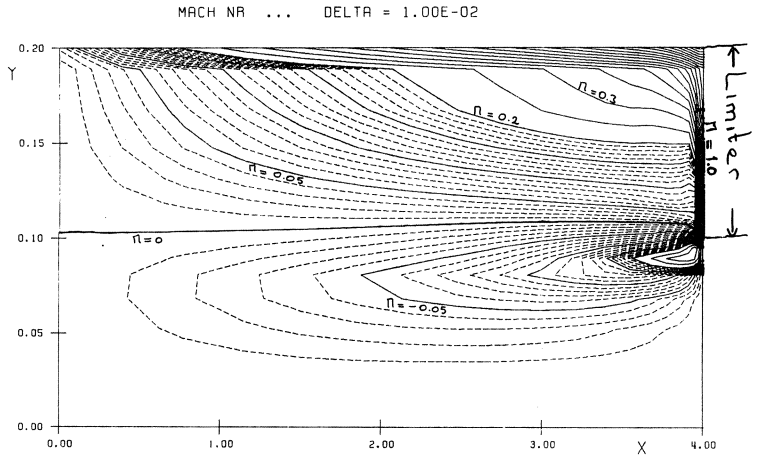


Fig. 5.

Contour plot of the mach numbers. The increments are 0.01 between the dottet contours and 0.05 between the solid contours.

APPENDIX A. THE COMPLETE SET OF EQUATIONS

We employ a system of equations governing the ion density n_i, the parallel flow velocity $u_\parallel$, the radial diffusion velocity v, and electron- and ion temperatures T_e and T_i. Auxiliary physical quantities are the electron density, $n_e = Z_i n_i$, the mass density, $\rho = m_i n_i$, the total and partial pressures, $p = p_e + p_i = n_e T_e + n_i T_i$, and the poloidal flow velocity, $u = (B_\theta/B)u_\parallel$. The coordinates x and y correspond to the poloidal and radial directions respectively. $\sqrt{g}$, h_x and h_y are metric coefficients; the coordinate system may be curvilinear, although it must be orthogonal. The equations are:

$$\text{(A.1)}\quad \frac{\partial}{\partial t}n_i + \frac{1}{\sqrt{g}}\frac{\partial}{\partial x}\left(\frac{\sqrt{g}}{h_x}n_i u\right) + \frac{1}{\sqrt{g}}\frac{\partial}{\partial y}\left(\frac{\sqrt{g}}{h_y}n_i v\right) = S_n$$

$$\text{(A.2)}\quad \frac{\partial}{\partial t}(\rho u_\parallel) + \frac{1}{\sqrt{g}}\frac{\partial}{\partial x}\left(\frac{\sqrt{g}}{h_x}\rho u u_\parallel - \frac{\sqrt{g}}{h_x^2}\eta_x^i\frac{\partial u_\parallel}{\partial x}\right) + \frac{1}{\sqrt{g}}\frac{\partial}{\partial y}\left(\frac{\sqrt{g}}{h_y}\rho v u_\parallel - \frac{\sqrt{g}}{h_y^2}\eta_y^i\frac{\partial u_\parallel}{\partial y}\right) = S_{mu_\parallel} - \frac{B_\theta}{B}\frac{1}{h_x}\frac{\partial p}{\partial x}$$

$$\text{(A.3)}\quad v = -\frac{D}{h_y}\frac{\partial}{\partial y}(\ln n_i)$$

$$\text{(A.4)}\quad \frac{\partial}{\partial t}\left(\frac{3}{2}n_e T_e\right) + \frac{1}{\sqrt{g}}\frac{\partial}{\partial x}\left(\frac{\sqrt{g}}{h_x}\frac{5}{2}n_e u T_e - \frac{\sqrt{g}}{h_x^2}\kappa_x^e\frac{\partial T_e}{\partial x}\right) + \frac{1}{\sqrt{g}}\frac{\partial}{\partial y}\left(\frac{\sqrt{g}}{h_y}\frac{5}{2}n_e v T_e - \frac{\sqrt{g}}{h_y^2}\kappa_y^e\frac{\partial T_e}{\partial y}\right) = S_E^e - K_{ei}(T_e - T_i) + \frac{u}{h_x}\frac{\partial p_e}{\partial x} + \frac{v}{h_y}\frac{\partial p_e}{\partial y}$$

$$\text{(A.5)}\quad \frac{\partial}{\partial t}\left(\frac{3}{2}n_i T_i + \frac{1}{2}\rho u_\parallel^2\right) + \frac{1}{\sqrt{g}}\frac{\partial}{\partial x}\left(\frac{\sqrt{g}}{h_x}\left(\frac{5}{2}n_i u T_i + \frac{1}{2}\rho u u_\parallel^2\right) - \frac{\sqrt{g}}{h_x^2}\left(\kappa_x^i\frac{\partial T_i}{\partial x} + \frac{1}{2}\eta_x^i\frac{\partial u_\parallel^2}{\partial x}\right)\right) + \frac{1}{\sqrt{g}}\frac{\partial}{\partial y}\left(\frac{\sqrt{g}}{h_y}\left(\frac{5}{2}n_i v T_i + \frac{1}{2}\rho v u_\parallel^2\right) - \frac{\sqrt{g}}{h_y^2}\left(\kappa_y^i\frac{\partial T_i}{\partial y} + \frac{1}{2}\eta_y^i\frac{\partial u_\parallel^2}{\partial y}\right)\right) = S_E^i + K_{ei}(T_e - T_i) - \frac{u}{h_x}\frac{\partial p_e}{\partial x} - \frac{v}{h_y}\frac{\partial p_e}{\partial y}$$

S_n, $S_{mu_\parallel}$, S_E^e and S_E^i are volume sources of ions, momentum, electron and

ion energy. η_x^i and η_y^i are the poloidal and radial ion viscosity coefficients; $\kappa_x^{e,i}$ and $\kappa_y^{e,i}$ are thermal conductivities. The poloidal coefficients are related to classical parallel coefficients according to $\eta_x^i = (B_\theta^2/B^2)\,\eta_\parallel^i$, and similarly for $\kappa_x^{e,i}$. The radial coefficients, including D, are anomalous. $K_{ei}\,(T_e - T_i)$ is the electron-ion energy equilibration term, and the $\mathbf{u} \cdot \mathbf{grad}\, p_e$ term on the right hand side of the energy equations represents work done by the electric field.

ACKNOWLEDGEMENTS

This work would not have been possible without the Euratom mobility support, which enabled me to work first at UKAEA Culham laboratory and subsequently at IPP Garching. I am indebted to Prof. F. Engelmann, to Dr. M.F.A. Harrison and the Exhaust Physics Group at Culham, and to Dr. K. Lackner and the Tokamak Physics Group at IPP for many fruitful discussions. This work was performed under the Euratom–FOM association agreement with financial support from ZWO and Euratom.

REFERENCES

1. M.F.A. Harrison, *Boundary Plasma*, in "Applied Atomic Collision Physics", Edited by H.S.W. Massey, B. Bederson and E.W. McDaniel, Academic Press, New York, 1983.
2. B.J. Braams, *Numerical Studies of the Two-Dimensional Scrapeoff Plasma*, Presented at the 11th European Conference on Controlled Fusion and Plasma Physics, Aachen, 1983, Europhysics Conference Abstracts **7D–II** (1983), 431–434.
3. B.J. Braams, P.J. Harbour, M.F.A. Harrison, E.S. Hotston and J.G. Morgan, *Modelling of the Boundary Plasma of Large Tokamaks*, J. Nucl. Mater. **121** (1984), 75–81.
4. S.V. Patankar, "Numerical Heat Transfer and Fluid Flow", Hemisphere, New York, 1980.
5. S.V. Patankar and D.B. Spalding, *A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flows*, Int.J.Heat Mass Transfer **15** (1972), 1787–1806.
6. H.J. Stone, *Iterative Solution of Implicit Approximations of Multi-Dimensional Partial Differential Equations*, SIAM J.Numer.Anal. **5** (1968), 530–558.
7. P.J. Roache, "Computational Fluid Dynamics", 2nd ed., Hermosa, 1976.
8. A. Brandt, *Guide to Multigrid Development*, in "Multigrid Methods", Proceedings of the Conference held at Köln–Porz, Nov. 1981 (W. Hackbusch, U. Trottenberg, eds.). Lecture Notes in Mathematics., Springer, Berlin, 1982.
9. Numerical Algorithms Group, "NAG Fortran Manual for Mark 10", NAG, Oxford, 1983.
10. C.E. Singer and B.J. Braams, *Low Temperature Plasma near a Limiter — A Solution for the Tokamak Fusion Core Experiment*, Applied Physics Division Report No. 30, Princeton Plasma Physics Laboratory, 1984. (to be published).

LEAST SQUARES NUMERICAL ANALYSIS OF THE STEADY STATE AND TRANSIENT THERMAL HYDRAULIC BEHAVIOUR OF L.M.F.B.R. HEAT EXCHANGERS

H. de BRUIN

The numerical method, developed in this paper, has been applied successfully to analyse three-dimensional, cylindrically symmetric flow and heat transfer in a heat exchanger. To describe the complex transport phenomena in the flow around the tubes, use has been made of the well known "liquid-structure continuum" approach, with continuously distributed flow resistance and heat transfer mechanism. The analysis leads to several sets of first order partial differential equations. These are solved numerically with a least squares programming package.

1. INTRODUCTION

Least squares numerical methods have been proposed by several authors, as a generalization of the so-called penalty methods, well known in finite element (F.E.) theory. Most serious applications of finite elements in the field of heat and mass transfer, however, seem still to be based on techniques of the Bubnov-Galerkin kind. One can establish objectively that, up to now, least squares finite element methods (L.S.FEM) never became that popular. As it becomes clear now how L.S.FEM has to be practised, when dealing with heat exchanger problems, this lack of popularity seems to be justified, indeed, by a number of misunderstandings.

Firstly, strictly speaking, L.S.FEM is not a finite element method at all. There is no classical variational principle; one has to find an

approximation for the differential equations *as they stand*. For this reason, the technique would be conceived probably much better as a kind of finite difference (F.D.) method.

Taking this view, it will be understandable, for example, that procedures of so-called "reduced integration" are merely standard for least squares finite elements. As this turns out to be the same as saying that there corresponds only *one* approximation to each equivalent finite difference cluster. Furthermore, F.D. techniques like "upwind differencing", which is the same as moving the integration point along a streamline, can be used to assure *stability* of least squares elements.

When dealing with a great number of different kinds of F.E. clusters in one mesh, as is the case with heat exchanger temperature calculations, another difficulty arises. In the classical finite element approach, each approximation, after squaring, is integrated over the element. Each approximation is thus weighted with the area of the element on which it is taken. This, as a rule, results in a very *bad conditioning* of the global "stiffness" matrix. By discarding for a moment the finite element character of L.S.FEM, this phenomenon can be cured, however, easily and effectively: namely by weighting each approximation with its "length" in vector space.

On the other hand, least squares methods become rather clumsy, if not completely unworkable, if one is not allowed to make use of some achievements of finite element theory. The useful "matrix method", for example, is, in fact, nothing more than a modification of the classical "shape function method", a technique which is standard in current F.E. applications.

After first squaring the approximations, and then differentiating with respect to the unknowns, one obtains positive symmetric systems of equations. These in turn can be handled and solved by standard finite element (e.g. wave front) procedures.

We conclude that, apparently, it seems to be essential for the succes of the least squares numerical method that finite differences and finite elements are treated at a uniform base.

2. FINITE DIFFERENCE ELEMENTS

As has been stated before (§1), the least squares finite element method is not a finite element method in the proper sense of the word. On the contrary there is much resemblance to classical finite differences. Therefore, it appears to be essential for the success of L.S.FEM that finite elements and finite differences be treated at a uniform mathematical basis.

2.1. Finite cluster spaces

Although finite differences and finite elements are distinct in origin, theory, and practice, they do have something in common. Both methods use small clusters of nodal points, as their units for discretization. Within such a *finite cluster* - called finite difference star or finite element respectively - function behaviour is approximated by polynominal expansion, whereby the function values are interpolated at the nodes.
As an example, let us take the four-node computational molecule of figure (1), defined by:

$$(1) = (-1,-1) \quad ; \quad (2) = (+1,-1)$$
$$(3) = (-1,+1) \quad ; \quad (4) = (+1,+1). \tag{1}$$

A vectorspace of dimension 4, spanned by a set of 4 independent polynomials, is associated with this molecule.
By definition, the *Finite Difference* (F.D.) base of the four-node cluster is given by:

$$\ell_i = (1,\xi,\eta,\xi\eta) \quad ; \quad i = 1,\ldots,4.$$

An arbitrary approximation ϕ is defined upon this base as:

$$\phi = a_1 + a_2\xi + a_3\eta + a_4\xi\eta \quad \text{or:}$$

$$\phi = \ell_i a_i. \tag{2}$$

The well-known summation convention is employed throughout this paper. The polynomial approximation (2) is fully equivalent to a Taylor series expansion around some origin 0:

$$a_1 = \phi(0) \quad ; \quad a_2 = \frac{\partial\phi}{\partial\xi}(0) \quad ; \quad a_3 = \frac{\partial\phi}{\partial\eta}(0) \quad ; \quad a_4 = \frac{\partial^2\phi}{\partial\xi\partial\eta}(0).$$

By definition, the *Finite Element* (F.E.) base of the four node cluster is given by the set of shape functions:

$$N_i = \{\tfrac{1}{4}(1\pm\xi)(1\pm\eta)\} \quad ; \quad i = 1,\ldots,4.$$

An arbitrary approximation ϕ is defined upon this base as:

$$(3) \qquad \phi = \tfrac{1}{4}(1-\xi)(1-\eta)\phi_1 + \tfrac{1}{4}(1+\xi)(1-\eta)\phi_2 + \tfrac{1}{4}(1-\xi)(1+\eta)\phi_3 + \tfrac{1}{4}(1+\xi)(1+\eta)\phi_4.$$

The *transition matrix*, transforming the F.D.-base into the F.E.-base, is defined by $[c_{ij}]$, where:

$$(4) \qquad N_i = \ell_i c_{ij} \quad \text{and} \quad a_i = c_{ij}\phi_j.$$

For many current molecule-forms, such as the cluster (1), the transition matrix proves to have properties of *orthogonality*.

In our case:

$$[c_{ij}] = \tfrac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

So that:

$$[c_{ij}]^T = \tfrac{1}{4}[c_{ij}]^{-1}.$$

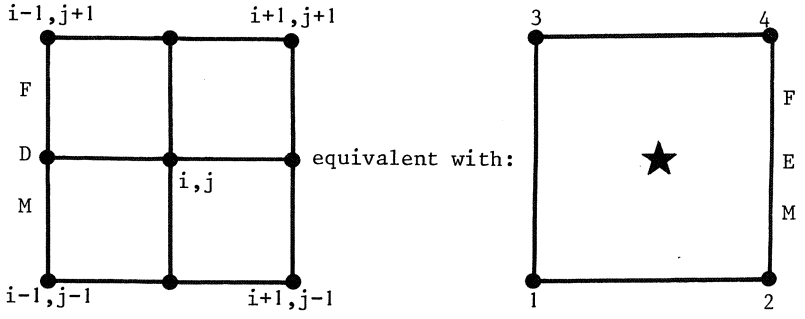


Figure 1. Four-node cluster.

Let us once more consider the finite *difference* star, depicted in figure 1. Associated with this star are the following F.D. schemes:

$$\frac{\partial\phi}{\partial x}\Big|_{i,j} = \frac{1}{4h}(\phi_{i+1,j+1}+\phi_{i+1,j-1}-\phi_{i-1,j+1}-\phi_{i-1,j-1}) + 0(h^2)$$

$$\frac{\partial\phi}{\partial y}\Big|_{i,j} = \frac{1}{4h}(\phi_{i+1,j+1}+\phi_{i-1,j+1}-\phi_{i+1,j-1}-\phi_{i-1,j-1}) + 0(h^2)$$

$$\frac{\partial^2\phi}{\partial x\partial y}\Big|_{i,j} = \frac{1}{4h^2}(\phi_{i+1,j+1}+\phi_{i-1,j-1}-\phi_{i+1,j-1}-\phi_{i-1,j+1}) + 0(h^2)$$

and, last but not least:

$$\phi|_{i,j} = \frac{1}{4}(\phi_{i-1,j-1}+\phi_{i-1,j+1}+\phi_{i+1,j-1}+\phi_{i+1,j+1}) + 0(h^2).$$

The associated quadrilateral *element*, with *reduced* integration, is depicted also in figure 1.

Identify:

$$x/h = \xi, \quad y/h = \eta, \quad (i,j) \leftrightarrow (*)$$

$$(i-1,j-1) \leftrightarrow (1), \quad (i-1,j+1) \leftrightarrow (3)$$
$$(i+1,j-1) \leftrightarrow (2), \quad (i+1,j+1) \leftrightarrow (4).$$

The element is second order $0(h^2)$ accurate; which is obvious now.
Transition matrix between the element and the difference star is the orthogonal matrix (4).

2.2. N-dimensional cube.

Generalization of the results above seems to be possible, and, in fact, proves to be appropriate.

In N-dimensional space, a unit cube has 2^N nodal points. Accordingly, there are 2^N elementary polynomials in its F.D. base. Let the k-th coordinate in N-space be named $\xi(k)$.
The F.D. base of the unit cube can then be conceived as a direct product:

$$\{\ell_i\}_{i=1,2^N} = \bigoplus_{k=1}^{N} \{1,\xi(k)\}.$$

The coefficients a_i in the Taylor-series expansion (2) are, accordingly:

$$\{a_i\}_{i=1,2^N} = \bigoplus_{k=1}^{N} \{1,\partial/\partial\xi(k)\}\phi(0).$$

The finite element (F.E.) base of the unit cube is given by the set of shape functions:

$$\{N_i\}_{i=1,2^N} = 1/2^N \bigoplus_{k=1}^{N} (1\pm\xi(k)).$$

The transition-matrix, transforming the F.D. into the F.E. base, is orthogonal for all members of the N-cube family.

To be more specific, let n be the index of the n-th coordinate of the nodal point named (i). The coordinates themselves are then defined locally by the FORTRAN inline-function [14]:

```
kube (i,n) = 2*mod((i-1)/2**(n-1),2)-1
```

where $i = 1,\ldots,2^N$ and $n = 1,\ldots,N$; mod(k,ℓ) is the remainder of k divided by ℓ. The value of the i-th shape-function at an arbitrary point, with (local) coordinates p(n), is then calculated by the following algorithm:

```
      shp = 1./2**N
      DO 1 n = 1,N
    1 shp = shp*(1.+kube(i,n)*p(n)).
```

At last, the value of the approximation $\phi = N_i\phi_i$ at p can be calculated, taking the inner product of $(shp)_{i=1,N}$ and $(\phi_i)_{i=1,N}$.

It is clear that the bilinear element (1) can be considered as a special case of the N-dimensional cube, for N = 2.

In two-dimensional transient heat transfer analysis, the 3-cube can be used as an element for approximation. The third dimension is then interpreted as a time coordinate.

Other numbers of the N-cube family, of course, include the one-dimensional linear element, used in both space and time. And, last but not least, the zero-dimensional element; that is a single point in space-time, used to describe known function values at the boundary.

A uniform treatment like the one given above can be seen to be advantageous now, since all those elements needed can be coded *at once;* thus saving a lot of FORTRAN writing, and ... errors.

3. NORMED APPROXIMATIONS

At this stage, the four node cluster could be completed with four so-called *integration points*, the coordinates of which, in general, are:

$$(\xi_k,\eta_k) = (\pm p,\pm q),$$

where $k = 1,\ldots,4$ and $p = q = 1/\sqrt{3}$ (:Gauss). In this way, the finite cluster is turned into a conventional, numerically integrated, finite element. See e.g. Zienkiewicz [1].

This case is worked out further in the paper of De Bruijn/Zijl [2]. Applying the least squares method, however, elements with only *one* integration point (reduced), as a rule, turn out to be far more appropriate. And the "matrix method" presented in [2,3] can be simplified accordingly, without loss of generality.

3.1. Isoparametrics

The four-node cluster (1) is conceived as a parent-element for the general quadrilateral, which is derived from it by isoparametric mapping, see [1] chapter 8:

$$(5) \qquad x = N_i x_i, \quad y = N_i y_i.$$

The mapping is characterized by a Jacobian determinant:

$$J = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}$$

which must be nonzero everywhere "inside" the element.

Applying a piece of standard finite element [1] theory we find:

$$(6) \qquad \begin{aligned} \frac{\partial N_i}{\partial x} &= \left(\frac{\partial y}{\partial \eta}\frac{\partial N_i}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial N_i}{\partial \eta}\right)/J \\ \frac{\partial N_i}{\partial y} &= \left(-\frac{\partial x}{\partial \eta}\frac{\partial N_i}{\partial \xi} + \frac{\partial x}{\partial \xi}\frac{\partial N_i}{\partial \eta}\right)/J. \end{aligned}$$

It should be observed that the functions $\partial(x,y)/\partial(\xi,\eta)$ and J are all linear in ξ,η. Using the "shape function method", as described in [1] pg. 708-709 and §9.2., the expressions (6) must be evaluated at an integration point (*).

By doing this, we construct, in fact, the following matrices:

$$DDX_j = \frac{\partial N_j}{\partial x}(*); \qquad DDY_j = \frac{\partial N_j}{\partial Y}(*).$$

Now $[DDX_j]$ and $[DDY_j]$ can be interpreted as matrix equivalents corresponding to the differentiations $\partial/\partial x$ and $\partial/\partial y$. They are called "differentiation-matrices", according to Csendes [4,5], or "derivation matrices", according to Boisserie [6].

This set of elementary approximations is completed by adding the case of differentiation to the zero'th coordinate, previously [2,3] called the projection matrix:

$$P_i = N_i(*). \tag{7}$$

Projection matrix (7) and differentiation matrices (6) may also be called the fundamental or universal matrices of the element. Nothing more seems to be needed, for consistent and effective application of finite element and finite difference methods.
For the family of N-cube elements, the fundamental approximations can be coded in a short-hand way, leaving free the number of dimensions actually implemented. This topic, however, will not be pursued further at the moment.

The so-called "matrix-method", which is associated in a natural way with the use of $[DDX_i]$, $[DDY_i]$ and $[P_i]$ was demonstrated in earlier publications like [3]. In the sequel, it will be referred to simply as the "approximation method".

3.2. An example: ideal flow

Given an appropriate set of elementary approximations, it is very easy to write down the approximations for the partial differential *equations*, governing the problem at hand. This will be demonstrated now for the problem of shell-side flow in the SNR-300 heat exchanger (described in chapter 7).

It was proven in [13] that this flow field can be assumed to be approximately irrotational. Furthermore, it is assumed that the problem is, as a whole, cylindrically symmetric. The differential equations for incompressible flow, then, simplify to:

$$\frac{\partial(rv)}{\partial r} + r\frac{\partial u}{\partial z} = 0$$
$$\frac{\partial u}{\partial r} - \frac{\partial v}{\partial z} = 0.$$

where r, z = radial/axial coordinate; v, u = radial/axial velocity.

To apply now the approximation method conveniently, these equations must first be presented in so-called operator-form, that is:

$$(8) \qquad \begin{bmatrix} \partial/\partial r.r & r.\partial/\partial z \\ -\partial/\partial z & -\partial/\partial r \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix} = 0.$$

As a unit for discretization of the equations (8) the *reduced* quadrilateral (see figure 1) proves to perform well. Triangles do not! If the flow domain is assumed to be rectangular, built up from elements with sizes Δr and Δz, the elementary approximations can be constructed easily:

$$[P_i] = \tfrac{1}{4}[1, 1, 1, 1]$$
$$[DDR_i] = \tfrac{1}{2}[-1, 1, -1, 1]/\Delta r$$
$$[DDZ_i] = \tfrac{1}{2}[-1, -1, 1, 1]/\Delta z.$$

The numerical equivalents of (8) are then FORTRAN - coded [14] according to the following algorithms:

```
      DO 1 I = 1,4
      A(2*I-1) = DDR(I)*R(I)
    1 A(2*I) = R(I)*DDZ(I)
      - - - - - - - - - - - -
      DO 2 I = 1,4
      A(2*I-1) = -DDZ(I)
    2 A(2*I) = DDR(I).
```

Where the radial velocities are associated with uneven, and the axial velocities with even indices, which is simply a matter of bookkeeping. R(I), I = 1,...,4 are the radial positions of the nodes of the element. Note that this kind of making approximations is, so to speak, *structure preserving:* one can "read back" the original form of the P.D.E's from the FORTRAN text. This facilitates things like documentation, updating and debugging of the computer program a great deal. In general, given an appropriate set of universal matrices, it proves to be very easy to write down approximations for the partial differential equations, or variational principle, governing the problem. The matrix method allways results in expressions of the form:

(9) $\quad r = A_i \phi_i$

where r = residual of the equations, ϕ = vector of unknowns, A = so-called *approximation* matrix [2,3] of the problem, specified for the element under consideration.

3.3. Scaling problem

After constructing the cluster approximation matrices, some method must be devised to obtain a system of equations which is representative for the problem as a whole. So far, nothing has been said about the numerical method to be employed for that purpose. Assembling of the local approximations, like (8), could be accomplished, in a finite element way, by replacing all local indices I by global numbers, associated with the mesh as a whole. After that, simply *summing up* all the approximations, together with the boundary conditions, will result in a global set of F.D. equations.

Dealing with a great number of elements, of different sizes and kinds, this global system, however, could hardly be expected to behave like a well conditioned one. In the worst case, some element contributions will be far too pronounced, while others, so to speak, will be almost invisible to the computer. Therefore, a kind of weighing procedure has to be devised, to assure that all element approximations contribute with the same order of magnitude to the global scheme.

In so far as the problem itself is not ill-conditioned, conditioning of the global matrix can be assured simply and effectively by the following procedure. Divide each approximation by its length in vector space. In other words: use *normed* approximations.

Let $[A_i]$ be a given approximation-matrix. The *norm* of $[A_i]$ is given by its length in vector space:

(10) $\quad Sp(A) = \sqrt{\left(\sum_{i=1}^{N} A_i^2\right)}.$

Divide each approximation by its norm:

$$A_i = A_i / Sp(A).$$

After that, Sp(A) = 1 for all approximations A.

In the sequel, all approximations will be assumed to be normed.

4. THE L.S.FEM PACKAGE

The *residual* r_E of the approximating equations, specified for a finite cluster E, is given by (9):

$$r_E = A_i^E \phi_i \tag{11}$$

where $[A^E_i]$ is the normed approximation at E.

4.1. Least squares method

We shall consider, once again [2,3], the general form of discretized equations, resulting from the *least squares* finite element principle of approximation; see [1] pg. 87-89. Applying the least squares criterion, the (weighted) sum of the squares of the residuals of the differential equations at all the integration points in the mesh should be a minimum:

$$\sum_E r_E^2 = \sum_E A_i^E \phi_i . A_j^E \phi_j = \min. \tag{12}$$

Here it is implicitly assumed that an element with several integration points can always be conceived as a superposition of reduced elements.

By analogy with the conventional finite element method [1], we can see that the following definition makes sense:

$$E_{ij} = A_i^E A_j^E \tag{13}$$

$[E_{ij}]$ is called *element matrix* or "local stiffness matrix". The algorithm (13) can be coded once and for all.

The least squares procedure of assembly (12) is completed by adding the element-matrices $[E_{ij}]$ to the *global* or *system matrix* $[S_{ij}]$ in the usual finite element way:

$$S_{ij} = \sum_E E_{ij} . \tag{14}$$

Boundary conditions need not to be handled separately; they can be conceived as elements (= nodes) of *zero* dimension, which is quite as simple.

By differentiating to the unknowns ϕ_n, the minimum problem (12) is transformed into a system of linear equations:

(15) $\quad S_{nj}\phi_j = B_n.$

where the *load vector* B is constructed from known contributions. Note that the S-matrix is symmetric, positive definite and, in many cases, banded. The global system of equations can be solved by straightforward Gaussian elimination, using the wave front method.

By application of Schwartz-inequality at (13) and (14) it can be shown that the following relations hold:

$$E_{ii}.E_{jj} \geq E_{ij}^2 \quad \text{and} \quad S_{ii}.S_{jj} \geq S_{ij}^2$$

The local and global matrices are diagonally dominant, in a mean square sense. Thus, for example, pivoting procedures will not be needed. Furthermore, due to norming of the approximations A_i, the following relationships hold:

$$\sum_i E_{ii} = 1 \quad \text{and} \quad \sum_i S_{ii} = \text{total number of elements.}$$

4.2. Discussion

It was pointed out already that least squares finite element methods are strongly related to conventional finite differences.
Let us make now this statement more precise. Let $A_i^E\phi_i = 0$ with $i = 1,\ldots,N$ and $E = 1,\ldots,N$ be a finite difference system of equations. Obviously, a series of real numbers will be zero if and only if the sum of the squares of all those numbers is zero:

$$\sum_E (A_i^E\phi_i)^2 = \sum_E \phi_i A_i^E A_j^E \phi_j = \text{minimum} = 0.$$

It is clear now that this is nothing more than a special case of the least squares procedure (12).
For the minimum to be exactly zero, it is required that the number of independent F.D. equations (M) equals the number of unknowns (N), so that M=N. A "finite difference element" must therefore be chosen in such a way that the resulting number of discretized equations (= elements) exactly equals the number of unknowns (= nodes) in the grid.
This sounds trivial from a finite difference point of view. In our opinion, it is, instead, a substantial problem, inherent to the F.D. method as such.

Using a conventional finite element procedure, on the contrary, no such problem arises. This is because the r_E are here, in fact, residuals of improper "equations", the square of which namely constitute the integrand of the variational principle. And these residuals, of course, don't have to be zero.
In fact, the number of integration points (= reduced elements) is always *greater* than the number of unknowns: M > N. So that most F.E. methods give rise to *over*determined systems of F.D. equations, which are solved in a least squares sense.
In this respect, L.S.FEM shows some resemblance to conventional finite element methods. Indeed, for the least squares numerical method to work, it is *not* demanded that its residuals are *exactly* zero. It must be assured, however, that they do *converge* to zero. Otherwise, no realistic solution can be expected. This enlarges somewhat the possibilities of least squares with respect to conventional finite differences.

At the same time, it is not clear at first sight why one should square working finite difference schemes, differentiate to the unknowns, and manipulate them as if they were finite elements. What should be the advantage of spending more computation time and computer memory?
The answer probably lies in the fact that, once the F.D. schemes are squared, they can be handled as if they were conventional finite elements. No further research is needed, therefore, to put the method at work. Moreover, F.E. modular concepts like "connectivity" and "isoparametrics" help to avoid the overhead, still common to all F.D. programs [15], when attempting to get rid of inhomogeneous parts of the problem domain.
On the other hand, some of the more robust F.D. concepts, such as "unconditional stability", have been entered for good in L.S.FEM.

4.3. Programming structure [14]

The core of the package consists of a library of subroutines, invisible to the user. This library is "opened" by writing a user-defined MAIN program. MAIN has the following global structure:

```
PROGRAM MAIN
Bookkeeping (TAPES, DIMENSION, COMMON)
Prepare data (using the LSFEM-library)
CALL SOLVER (..., FIELD, ...)
Evaluation of results (also using the library)
STOP & END
```

In fact, the MAIN program is built completely around the function SOLVER, with FIELD as an external in its parameter list. SOLVER is a library routine which solves a large set of positive definite symmetric equations, using mass storage. FIELD is a user supplied procedure, with input parameter IG, being the nodal number for which a row of matrix-coefficients ROW (NB1) is asked, as output, by SOLVER. FIELD is called somewhere in the body of SOLVER accoring to: CALL FIELD (IG, ROW, NB1); NB1 = bandwidth + 1.

Subroutine FIELD has the following global structure:

```
      SUBROUTINE FIELD (IG, ROW, NB1)
      Bookkeeping, DIMENSION MENU (max (loops),3)
      DATA MENU /...; ...; .../
      CALL LOGIKA (IG,LABEL,LOOPS)
      DO 10 L = 1,LOOPS
      CALL TOPOL (IG,MENU(LABEL(L),1))
      CALL PHYSIK (MENU (LABEL(L),2))
      CALL PDE (MENU(LABEL(L),3))
      CALL NORMAL $ CALL LSFEM
   10 CALL INTEL
      CALL UITEL (IG, ROW)
      RETURN $ END
```

FIELD thus mainly consists of a number of calls to other modules.

Let us start with subroutine LOGIKA. The problem area is subdivided in a number of subsets, each of which is characterized by a logical predicate:

```
CASE (K) = true or false, as a function of IG.
```

Before returning to FIELD the indices (K) for which CASE (K) = .true. are stored into the array LABEL (LOOPS); so defining all the subsets of the mesh to which a current node IG belongs.

Each subset is characterized by the fact that it possesses a different topology, different physical properties and/or a different kind of "element" with respect to other subsets; Thus LOGIKA is a, user-made, *classification procedure* for TOPOL, PHYSIK and PDE.

The subroutines TOPOL, PHYSIK and PDE are also, in principle, user-supplied. They define respectively: *connectivity*, "material *constants*" and the kind of *discretization scheme* to be employed. These procedures are coded independently of each other and with no reference to the subsets of LOGIKA. They possess the following general structure:

```
      SUBROUTINE PHYSIK (LABEL), for example.
      Bookkeeping
      GOTO (1, 2, 3, 4, ...), LABEL
    1 CONTINUE
      calculation of constants type 1
      RETURN
    2 CONTINUE
      calculation of constants type 2
      RETURN
      ......
      ......
      END
```

The interconnection between the LABELs in LOGIKA and the LABELs in TOPOL/PHYSIK and PDE is accomplished by a, so-called, MENU *interface*, consisting of three entries, to assure the modularity of this part of the coding. It is remarked further that, contrary to the conventional F.E.M. packages, generating the connectivity and material properties can be kept completely in-core.

A few words should be said about the routine we called PDE. Although it is not impossible for the user to code such a routine himself, it is much more advantageous to supply the package with a number of preprogrammed modules, containing working discretization schemes for a number of commonly encountered partial differential equations. Reading through the following paragraphs, the reason for this will be still clearer. The task of routines like PDE is: to construct the approximation-matrix, according to the theory developed in §§2 and 3. Part of this task, again, will be accomplished elsewhere, by delegating the forming of elementary approximations

and isoparametrics to specialized routines from the library.

After constructing the approximation, governing the problem corresponding to LABEL (L), the A_i's are normed according to (10) with a call to NORMAL, and squared according to (13) with a call to LSFEM. They are subsequently counted into the system matrix (14) by a call to INTEL; this routine uses the frontal method [1]; it overwrites equations when they are no longer in use. Looping is performed through all the elements, needed to construct the complete matrix-row for the nodal point IG in question. Finally, ROW itself is delivered by UITEL, thereby leaving FIELD and returning control to the SOLVER-program.

5. STEADY STATE THERMAL HYDRAULICS

The numerical method has been worked out for three-dimensional, cylindrically symmetric flow and heat transfer in a heat exchanger [2,3]. To describe the complex transport phenomena in the flow around the tubes, use has been made of the "liquid-structure continuum" approach with continuously distributed flow resistance and heat transfer mechanism, described in [9,10,11,12,13]. This analysis leads to several different sets of partial differential equations. They are all solved with the same Least Squares programming package, and with the same kind of finite elements.

5.1. Governing equations

The shell-side flow field was treated as an example in §3. The equations once again, are:

$$(8) \qquad \begin{bmatrix} \partial/\partial r.r & r.\partial/\partial z \\ -\partial/\partial z & \partial/\partial r \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix} = 0.$$

Proper boundary conditions for this elliptic problem are: impermeability $u = 0$ or $v = 0$ at the solid boundaries, and prescribed (mutually consistent) velocity-profiles at the inflow and outflow openings. (Nummerically, *one* point of the boundary can be left free.) As it has been pointed out, the flow field can be modeled with a mesh, consisting of quadrilaterals with total reduction. In thiscase, the least squares method is fully equivalent with a volume method (finite differences), as indicated in §4. It was shown how the equations (8) are coded in FORTRAN, by using the matrix method: see §3.

The flow field, thus described, has to be calculated only once. Under transient conditions, multiplying it by a time-dependent scaling factor proves to be a sufficient approximation: see [13].
After solving the equations (8) for the velocities (v,u), which is a first order linear problem, the streamlines at shell-side can be found, being the contour map of the stream function $\psi(r,z)$:

$$(16) \qquad \frac{\partial \psi}{\partial r} = r.u \quad ; \quad \frac{\partial \psi}{\partial z} = -r.v.$$

One zero point value for ψ must be prescribed. Also the pressure distribution $p(r,z)$ can be calculated, using the equations:

$$(17) \qquad \begin{aligned} -1/\rho \frac{\partial p}{\partial r} &= v \frac{\partial v}{\partial r} + u \frac{\partial v}{\partial z} + \tfrac{1}{2}\sqrt{K_z u^2 + K_r v^2}.v \\ -1/\rho \frac{\partial p}{\partial z} &= v \frac{\partial u}{\partial r} + u \frac{\partial u}{\partial z} + \tfrac{1}{2}\sqrt{K_z u^2 + K_r v^2}.u \end{aligned}$$

where it has been assumed that the flow is steady. Also, *one* zero-point for the pressure must be given as a "boundary condition". A mesh of simple linear triangles can be used, to descretize the streamline and pressure equations (16-17) properly. The least squares procedure, in this case, has no finite difference equivalent. Undoubtedly, it is much more straightforward than designing marching schemes to any point in the pressure field; or solving a (second order) Poisson equation with conditions for the pressure gradient at the boundary, often difficult to find. Setting up the least squares "variational principle" for streamlines:

$$(18) \qquad \iint \left(\frac{\partial \psi}{\partial r} - r.u\right)^2 + \left(\frac{\partial \psi}{\partial z} + r.v\right)^2 \, r dr dz = \text{minimum}$$

we notice that the minimum will be zero only approximately.
The Euler equations, derived from (18), are:

$$\frac{\partial}{\partial r} r \frac{\partial}{\partial r} + r \frac{\partial^2 \psi}{\partial z^2} = \frac{\partial}{\partial r}(r^2 u) - \frac{\partial}{\partial z}(r^2 v).$$

So that: $\nabla^2 \psi = 2u$.

This is a Poisson equation for ψ, with non-zero right hand side due to cylindrical symmetry. So (18) establishes a link between least squares and conventional finite elements, for these kinds of "marching" problems.

As has been shown before, in [13], using the continuum-approach, also the laws of energy conservation and heat transfer turn out to be, again,

first order partial differential equations, of the hyperbolic type. Assuming cylindrical symmetry and steady state operating conditions, these equations simplify to:

$$\text{PRM}\left(v\frac{\partial T_p}{\partial r} + u\frac{\partial T_p}{\partial z}\right) + \text{COEF}_p(T_p - T_s) = 0, \tag{19}$$

$$\text{SEK}\frac{\partial T_s}{\partial z} + \text{COEF}_s(T_s - T_p) = 0. \tag{20}$$

Where r/z = cylindrical coordinates, T = temperature of p/s = shell-side/ tube-side sodium respectively.
As has been stated, the flow field (v,u) is calculated once. It is scaled up by: PRM/SEK = (time dependent) velocity scaling factors; COEF = coefficients of heat transfer and -capacity, depending upon the constructive details of the apparatus. Boundary conditions for the shell- and tube-side temperatures should be prescribed *only* at the respective inflow openings.

5.2. Stability

The rest of this chapter is devoted to the question which element should be adopted for appropriate discretization of the temperature equations. It turns out that, if it is required that the numerical scheme be *stable* under all operating conditions, no compromise, as suggested in [2,3], can be made.

The quadrilateral with reduced integration is adopted as an element, to discretize the shell-side equation (19), at a rectangular mesh:

$$[\text{PRM}(v.\partial/\partial r + u.\partial/\partial z) + \text{COEF}_p].T_p - 1.\text{COEF}_p.T_s = 0. \tag{21}$$

The fundamental approximations are then given by:

$$\partial/\partial r = \tfrac{1}{2}[-(1-\eta) \quad +(1-\eta) \quad -(1+\eta) \quad +(1+\eta)]/\Delta r$$

$$\partial/\partial z = \tfrac{1}{2}[-(1-\xi) \quad -(1+\xi) \quad +(1-\xi) \quad +(1+\xi)]/\Delta z$$

$$1 = \tfrac{1}{4}[(1-\xi)(1-\eta) \qquad (1+\xi)(1-\eta)$$

$$(1-\xi)(1+\eta) \quad (1+\xi)(1+\eta)],$$

where the position of the integration point (ξ,η) has to be chosen properly. Let us define certain *dimensionless numbers* H, K, as follows:

(22) $$H = \frac{PRM.v}{\frac{1}{2}\Delta r.COEF_p}, \quad K = \frac{PRM.u}{\frac{1}{2}\Delta z.COEF_p}.$$

Working out now the approximation scheme (21) yields:

$$\begin{aligned}
&[-H(1-\eta)-K(1-\xi)+(1-\xi)(1-\eta)]T_1^p+\\
&+[+H(1-\eta)-K(1+\xi)+(1+\xi)(1-\eta)]T_2^p+\\
&+[-H(1+\eta)+K(1-\xi)+(1-\xi)(1+\eta)]T_3^p+\\
&+[+H(1+\eta)+K(1+\xi)+(1+\xi)(1+\eta)]T_4^p-\\
&-(1\pm\xi)(1\pm\eta)T^s = 0.
\end{aligned} \tag{23}$$

Let us assume, for simplicity, that:

$$H > 0, \quad K > 0 \quad \text{and} \quad H > K.$$

Interpreting (H,K) as a dimensioneless velocity vector, this situation can be depicted as in figure 2.

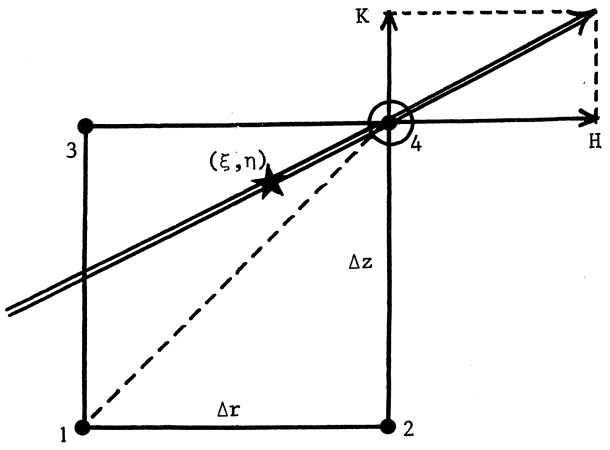


Figure 2. Reduced upwind integration.

A more serious restriction can be imposed by demanding that, in case of $H \to 0$ or $K \to 0$, the two-dimensional description automatically simplifies to one-dimensional theory.
This is accomplished most easily by positioning the integration point at

the *characteristic* going through a nodal point. In such a way, however, that the coordinates (ξ,η) lie *within* the element. In the situation sketched, only the characteristics through (1) and (4) can be chosen.
Now in the very special case that $H = K = 0$, which corresponds to a total blockage of the shell-side flow, the differential equation (21) reduces to an algebraic one, namely:

$$T_p - T_s = 0.$$

Such an equation is discretized best at a zero dimensional element. This must be one of the nodal points (1) or (4), if going continuously from $H, K > 0$ to the situation of zero velocity. However, at the nodal point (1), since it is situated in the upstream direction, a boundary condition could already be imposed. So, for reasons of consistency, an integration point can be placed only at the characteristic through (4). To be more precise, the coordinates (ξ,η) must obey the relationship:

$$H(1-\eta) = K(1-\xi). \tag{24}$$

To proceed further, let us interprete (23) as a finite difference equation to calculate the shell-side temperature T^p at nodal point (4). Suppose we have a complete system of such equations, say:

$$A_{ij}T_j = 0 \quad \text{for} \quad i,j = 1,\ldots,N. \tag{25}$$

Now a unique solution of the system (25) can be proven to exist, if it can be shown that the following conditions hold:

$$\begin{aligned} &(1)\ A_{ii} > 0 \ ; \quad A_{ij} \leq 0 \quad \text{for all} \quad j \neq i; \\ &(2)\ A_{ii} \geq \sum_{j=1,j\neq i}^{N} |A_{ij}| \ \text{with strict inequality for some } i; \\ &(3)\ A \text{ is irreducible.} \end{aligned} \tag{26}$$

For a proof of the above theorem, see Ames [7] page 101.
It is more or less implicitly stated in this proof that a solution of (25-26) will also be unconditionally *stable*; which is even of greater importance, from a practical point of view.
Recalling the equations (23) we obtain the following set of inequality conditions:

$$
\begin{aligned}
A_{41} &= -H(1-\eta)-K(1-\xi)+(1-\xi)(1-\eta) \leq 0 \\
A_{42} &= +H(1-\eta)-K(1+\xi)+(1+\xi)(1-\eta) \leq 0 \\
A_{43} &= -H(1+\eta)+K(1-\xi)+(1-\xi)(1+\eta) \leq 0 \\
A_{44} &= +H(1+\eta)+K(1+\xi)+(1+\xi)(1+\eta) > 0 \\
A_{4j} &= -(1\pm\xi)(1\pm\eta) \leq 0 \quad \text{for } j = 5,\ldots,8.
\end{aligned} \tag{27}
$$

Because of $-1 \leq \xi,\eta \leq +1$ and $H,K > 0$, it is clear that the condition $A_{44} > 0$ is fulfilled without question.
The same thing can be asserted for: $A_{4j} \leq 0$, $j = 5,\ldots,8$.

Looking at figure 2, it is seen that the nodal characteristic is embedded in a "domain of influence", which is thought to be bounded by the lines (3)-(4) and (1)-(4). Therefore it is suggested that, physically speaking, (4) obtains temperature information from (1) and (3), but is not to be influenced by (2), which is outside the domain. For this reason, it would probably be a good idea to set the corresponding matrix coefficient to zero: $A_{42} = 0$ in (27). Combining this with the characteristic equation (24) we obtain for ξ:

$$-2H\xi + 1 - \xi^2 = 0.$$

(Here it is tacitly assumed that $K \neq 0$.)
The solution of which must be written as:

$$\xi = 1/(H+\sqrt{(1+H^2)}). \tag{28}$$

The reasoning is completed by asserting that, with this value for ξ:

$$A_{41} \leq 0 \quad \text{and} \quad A_{43} \leq 0$$

(note that: $\xi > 1 - H$).

Once the sign of the matrix coefficients has been established, it is a simple matter to show that the remainder of the conditions (26) are fulfilled almost automatically. So we have proved that, with (24) and (28) determining the position of the integration points, the shell-side temperature problem as stated in (21) has a unique solution. It can also be asserted that this solution is *unconditionally stable*.

6. TEMPERATURE TRANSIENTS

The equations which describe temperature transients in the bundle of a shell-and-tube heat exchanger are given by [13]. Assuming cylindrical symmetry, they simplify to the following:

$$\frac{\partial T_p}{\partial t} + PRM\left(v\frac{\partial T_p}{\partial r} + u\frac{\partial T_p}{\partial z}\right) + COEF_p(T_p - T_w) = 0 \tag{29}$$

$$\frac{\partial T_s}{\partial t} + SEK \cdot \frac{\partial T_s}{\partial z} + COEF_s(T_s - T_w) = 0 \tag{30}$$

$$\frac{\partial T_w}{\partial t} = a\,\frac{1}{r}\,\frac{\partial}{\partial r}\left(r\,\frac{\partial T_w}{\partial r}\right). \tag{31}$$

Where: r/ z/ t = space-time coordinates; T = temperatures of p/ s/ w = shell-side/tube-side sodium and tube-walls; (v,u) = irrotational flow field, calculated once; PRM/SEK = time depending velocity scaling factors; COEF/a = coefficients of heat transfer and -capacity.

6.1. Space-time cube

Problems in space and time, such as temperature transients in a heat exchanger, will conveniently be described by space-time elements. In our opinion there is no reason why space coordinates and the time coordinate should *not* be handled at an equal footing [9].

Let us consider for example the space-time cube, depicted in figure 3. The F.D.-base of this finite cluster is given by:

$$\ell_i = \{1,\xi,\eta,\zeta,\xi\eta,\xi\zeta,\eta\zeta,\xi\eta\zeta\};\quad i = 1,\ldots,8.$$

which expresses an arbitrary approximation in a Taylor-series expansion around the origin (0,0,0).
The F.E.-base of the cluster is given by the shape-functions:

$$N_i = \{\tfrac{1}{8}(1\pm\xi)(1\pm\eta)(1\pm\zeta)\}\quad ;\quad i = 1,\ldots,8.$$

The eight node cluster is conceived as a parent element for the general cubic, which is derived from it by an isoparametric mapping (5). We can construct then differentiation-matrices like (6):

$$DDR_j = \frac{\partial N_j}{\partial r}\quad ,\quad DDZ_j = \frac{\partial N_j}{\partial z}\quad ,\quad DDT_j = \frac{\partial N_j}{\partial t}.$$

For reasons of stability, elements with reduced upwind integration will be used for the temperature problem.
The eight-node cluster can be turned into a more conventional finite element, if we supply it with 8 integration points. This case is worked out in De Bruijn [2], and will not be considered further.

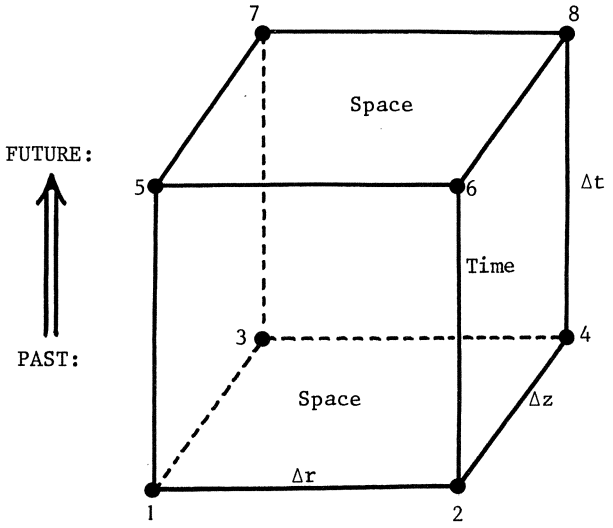


Figure 3. Space-time element.

Let us consider now in more detail the differential equation for calculating the shell-side temperature field T_p, that is equation (29):

$$[\partial/\partial t + PRM(v.\partial/\partial r + u.\partial/\partial z) + COEF_p].T_p - COEF_p.T_w = 0.$$

In analogy with (22), let us define the dimensionless numbers H, K, L, according to:

$$H = \frac{PRM.v}{\frac{1}{2}\Delta r.COEF_p}, \quad K = \frac{PRM.u}{\frac{1}{2}\Delta z.COEF_p}, \quad L = \frac{1}{\frac{1}{2}\Delta t.COEF_p}.$$

The integration point (ξ,η,ζ) must be chosen in such a way that the resulting F.D. scheme is unconditionally stable. Reasoning along the same lines as in §5, the following recipe has been found:

$$(\xi,\eta,\zeta) = sign(H,K,L).1 - \lambda.(H,K,L)/P,$$

(32) where $P = \max(|H|,|K|,|L|)$

and $\lambda = 1-1/(P+\sqrt{(1+P^2)})$.

In the case of $L = 0$, this reduces, indeed, to the stability criterion of §5.

Applying the matrix-method (§3) results in an approximation for each space-time element, of the form (11):

$$r_E = A_j^E \phi_j .$$

And the least squares principle (12) could, in principle, be applied.

A discrepancy with previous theory occurs, however, due to the fact that variables in a space-time element, when they belong to the *past*, are no longer *un*known. Only those of the future are. The approximation matrix can thus be split up in two parts, see figure 3:

$$\begin{bmatrix} A_j & \vdots & A_j \\ j = 1,2,3,4 & \vdots & j = 5,6,7,8 \end{bmatrix} \begin{bmatrix} \phi_{1,\ldots,4} \\ \phi_{5,\ldots,8} \end{bmatrix}.$$

By differentiating only to the *future* variables $\phi_{5,\ldots,8}$, most of the procedures can be maintained. Element-matrix and -vector are now constructed, moving known values to the right-hand side, as follows:

$$(33) \qquad E_{ij} = A_i A_j \quad ; \quad b_i = -A_i A_j \phi_j$$

where $5 \le i,j \le 8$; $5 \le i \le 8$, $1 \le j \le 4$; respectively.
These element-contributions are counted into the global matrix and load vector (S_{ij}, B_i), a system which, in principle, must be solved at each time-step. The resulting time-stepping scheme is thus an implicit one.

6.2. Substructuring

To discretize the tube-wall equations (31), which are of second order in space, no use can be made of cubic elements, since the second derivative would then disappear. A proper element for discretization of the tube-walls is depicted in figure 4.
Its F.D.-base is given by:

$$(1,\xi,\xi^2) \oplus (1,\eta)$$

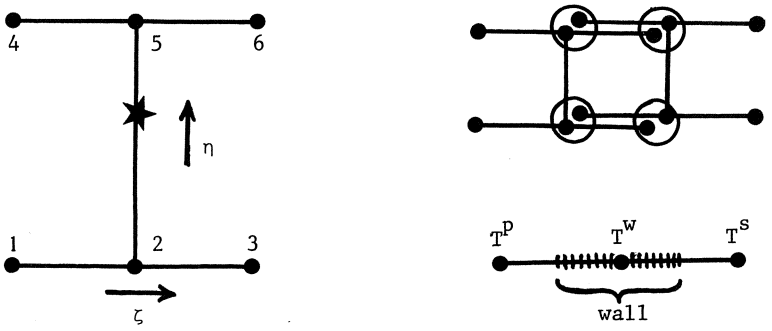


Figure 4. tube wall element

And its F.E.-base turns out to be:

$$(-\tfrac{1}{2}\xi+\tfrac{1}{2}\xi^2, 1-\xi^2, +\tfrac{1}{2}\xi+\tfrac{1}{2}\xi^2) \oplus \tfrac{1}{2}(1-\eta, 1+\eta).$$

The finite difference character of this element becomes still more clear if one looks at the way it is interconnected with other schemes: see figure 4. The integration point is chosen at $(0,\eta)$, where η is determined again by conditions of stability.

It can be justified that the tube walls can be discretized with sufficient accuracy by adopting just one element over the wall thickness, as depicted. The approximations for the equations (31), after time-stepping, then turn out to be of the following form:

$$\begin{bmatrix} A_1^w & A_2^w & A_3^w \end{bmatrix} \begin{bmatrix} T^p \\ T^w \\ T^s \end{bmatrix} = A_4^w.$$

Solving for T^w:

$$T^w = (A_4^w - A_1^w.T^p - A_3^w.T^s)/A_2^w. \tag{34}$$

After discretization and time-stepping, also an approximation is found for the tube-side equations (30):

$$\begin{bmatrix} A_1^s & A_2^s & A_3^s & A_4^s \end{bmatrix} \begin{bmatrix} T_1^s \\ T_1^w \\ T_2^s \\ T_2^w \end{bmatrix} = A_5^s. \tag{35}$$

Substitution of the expression (34) in (35) eliminates the tube-wall temperatures T^w, resulting in:

$$\begin{bmatrix} A_1^s - A_2^s A_3^w/A_2^w & ; & -A_2^s A_1^w/A_2^w & ; \\ ; A_3^s - A_4^s A_3^w/A_w^w & ; & -A_4^s A_1^w/A_2^w \end{bmatrix} \begin{Bmatrix} T_1^s \\ T_1^p \\ T_2^s \\ T_2^p \end{Bmatrix} =$$

$$= A_5^s - A_2^s A_4^w/A_2^w - A_4^s A_4^w/A_2^w.$$

In exactly the same way, the tube wall temperatures T^w can be eliminated from the shell-side approximations.
Thus, the resulting global matrix needs only to be solved for the primary and secondary temperatures (T_p,T_s). This means a considerable computer memory and computation time saving. After solving for the temperatures of the fluid, the solid temperatures can be found using (34).
The technique used here is known in finite element theory as *substructuring* or making *superelements*. Analogous procedures can be used to construct much more efficient (one-dimensional) temperature elements for the parallel flow region in a heat exchanger with straight tubes. See Peters [8].

7. MODELLING AND RESULTS

The SNR is a sodium cooled nuclear power plant, with a rated electrical output of 300 MW, situated at Kalkar in West Germany.
The heat exchanging system is looptype and consists of:
- the reactor
- the primary pumps
- the *intermediate heat exchangers*
- the secondary sodium circuit with steam generators and secondary pumps
- the tertiary circuit, a water/steam circuit, with turbines, condensors and feedwater pumps.

The primary objective of a simulation of the thermal-hydraulic behaviour of an intermediate heat exchanger in a sodium-colled fast breeder reactor is to predict the temperature field under steady-state and transient conditions. Very often, the transient conditions arise from postulated hypothetical accidents, and the predicted temperature fields serve as a basis for stress analyses, to prove the integrity of the component.

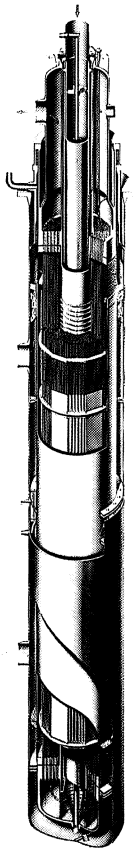


Figure 5. Intermediate Heat exchanger.

7.1. Hardware

A picture of the Intermediate Heat Exchanger (I.H.X.) is shown in figure 5.

At the shell-side hot primary sodium flows through a nozzle into the apparatus, and via an inlet perforation into the bundle. In this tube-bundle, heat exchange takes place with secondary sodium. The cooled primary sodium then flows through an outlet perforation and streams out.

At the tube-side, cold secondary sodium flows down through a central tube. At the bottom of the apparatus, its direction is reversed, and via the lower tube-sheet it enters the bundle. In the tube-bundle it warms up by exchanging heat with the primary sodium. Then, the heated sodium flows through the upper tube-sheet and, finally, streams out.

Geometrical data of the prototype are presented in the following table:

outer diameter of inner shell	0.460 m
inner diameter of outer shell	0.960 m
length of tube bank	7.154 m
height of inflow and outflow openings	0.370 m
outer tube diameter	0.0210 m
inner tube diameter	0.0182 m
pitch (triangular)	0.0270 m
number of tubes	846

Process data are: shell-side mass flow rate 360 kg/s, tube-side mass flow rate 256 kg/s, shell-side inflow temperature 490^{o}C, and tube-side inflow temperature 349^{o}C.
Our heat exchanging components were tested at full scale, with true liquid sodium in it, at high temperatures. The experiments were carried out by TNO and Neratoom at the 50 MW Test facilities in Hengelo, the Netherlands.
In the following short discussion, numerical results will be compared to experimental data.
These data have been obtained from the test model with the dimensions described in the above-presented table. Calculated and measured shell-side temperature profiles are shown in figure 6.
The calculated temperature profiles, in axial direction, are almost straight lines. The measurements are plotted as separate points. The agreement perhaps seems to be rather bad at first sight, but this is readily explained.
One always has to complete the picture with the radial temperature profiles, also shown. It is seen that a large scatter in the measurements correponds, very precisely, to steep gradients in the calculated lines.

7.2. Software

Least squares finite difference elements were used to model numerically the tube-bundle region of the I.H.X. The final mesh consists of 20 lines in radial direction, and 89 lines in axial direction, which gives us a total of 1780 unknown temperatures and a bandwidth of 23.

As has been explained in §4, a program MAIN must be written by the user, mainly to define the sequence in which the calculations must be performed. In our case, first the flow field must be calculated, followed by a calculation of steady state temperatures. Both serve as an input for the transient calculations:

```
PROGRAM MAIN
Bookkeeping
CALL RASTER (: generating the grid)
CALL SOLVER (..., FLOW, ...)
Evaluate flow field/prepare other data
CALL SOLVER (..., STEADY, ...)
Evaluate steady state temperature field
```

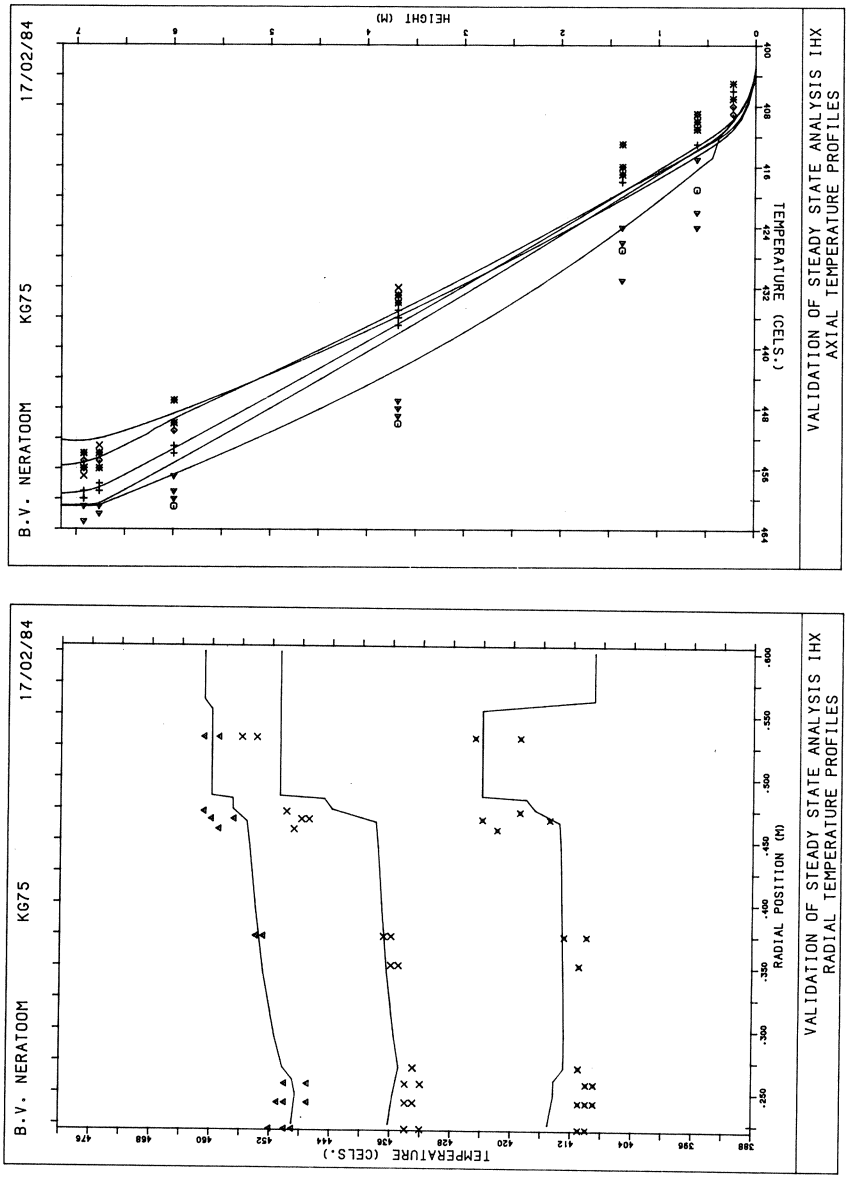

B.V. NERATOOM KG75 17/02/84
HEIGHT (M)
TEMPERATURE (CELS.)
VALIDATION OF STEADY STATE ANALYSIS IHX
AXIAL TEMPERATURE PROFILES
B.V. NERATOOM KG75 17/02/84
TEMPERATURE (CELS.)
RADIAL POSITION (M)
VALIDATION OF STEADY STATE ANALYSIS IHX
RADIAL TEMPERATURE PROFILES

Figure 6.

```
      DO 10 IT = 1, number of time-steps
      Define transient operating conditions
      CALL SOLVER (..., TRANSIENT, ...)
      Evaluate transient temperature field
   10 CONTINUE
      STOP & END
```

The routines FLOW, STEADY and TRANSIENT are provided with a structure analoguous to FIELD, as has been explained in §4. Assuming for a moment that a suitable library of approximation schemes PDE exists, then each of the routines FLOW, STEADY and TRANSIENT must still be equipped with a classification LOGIKA, a connectivity TOPOL, physical properties PHYSIK and a software interface defined by MENU data. The detailed structure of these subroutines is highly problem dependent.

To model the steady state temperature field of a SNR-300 heat exchanger, 27 distinct domains, were distinguished. The MENU card connected 15 different topologies, 15 kinds of physical properties and 8 discretization schemes. Details of this modelling include boundary effects near the central tube and at the other region of the bundle. These details, being one-dimensional, are visualized as broken lines: see figure 6.

8. SUMMARY

The numerical method, developed in this paper could be characterized by the following features:

1. the construction of elements in space and time, most of them being members of the isoparametric quadrilateral family with reduced integration;
2. partially upwind integration of the elements describing convection, thus assuring numerical stability;
3. a matrix representation for the differential operators, which enables a very efficient coding of the differential equations governing the problem;
4. an automatic scaling procedure, which ensures that the global equations of the problem will be optimally conditioned;
5. a least squares finite element procedure, resulting in a global matrix which is positive definite and symmetric;

6. techniques of substructuring.

The numerical method has been successfully applied to analyse three-dimensional, cylindrically symmetric flow and heat transfer in a heat exchanger.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Mr. Wouter Zijl of T.N.O. Groundwater Survey, for his continuous help and support.

9. REFERENCES

[1] ZIENKIEWICZ, O.C., *The finite element method*, third edition; Mc. Graw-Hill (U.K. 1977).

[2] DE BRUIJN, J.G.M. & W. ZIJL, *Least squares finite element analysis of the steady and transient thermal hydraulic behaviour of L.M.F.B.R. heat exchangers.* Proceedings of the international symposium on Refined modelling of flows, Paris, France, september 1982.

[3] DE BRUIJN, J.G.M. & W. ZIJL, *Least squares finite element solution of several thermal-hydraulic problems in a heat exchanger,* Proceedings of the second international conference on numerical methods in thermal problems, Venice, Italy, July 1981.

[4] CSENDES, Z.J., *A finite element method for the general solution of ordinary differential equations,* Int. J. Num. Meth. Engng. 9, 551-561 (1975).

[5] CSENDES, Z.J., *A fortran program to generate difference formulas,* Int. J. Num. Meth. Engng. 9, 581-599 (1975).

[6] BOISSERIE, J.M., *Generation of two- and three-dimensional finite elements,* Int. J. Num. Meth. Engng. 3, 327-347 (1971).

[7] AMES, W.F., *Numerical methods for partial differential equations,* second edition, Acad. Press 1977; chapter 3-3.

[8] PETERS, F.J., *A novel organization of finite element calculations and homogeneous structures,* Int. J. Num. Meth. Engng. vol. 17, no. 9, 1430-1435 (1981).

[9] PATANKAR, S.V. & D.B. SPALDING, *A Calculation Procedure for the Transient and Steady-State Behaviour of Shell-and-Tube Heat Exchangers*, in: *Heat Exchangers: Design and Theory Sourcebook*, Ed. N. Afgan and E.U. Schlünder, McGraw-Hill, New York (1974).

[10] KAO, T.T. & S.M. CHO, *A numerical Solution Method for the Prediction of Flow and Thermal Distribution in Shell-and-Tube Heat Exchangers*, Joint ASME/AICHE 18th Nat. Heat Transfer Conf., San Diego (1979).

[11] SHA, W.T., *An Overview on Rod-Bundle Thermal-Hydraulic Analysis*, Nuclear Engineering and Design, 62, 1-24 (1980).

[12] DOMANUS, H.M., V.L. SHAH & W.T. SHA, *Applications of the Commix Code Using the Porous Medium Formulation*, Nuclear Engineering and Design, 62, 81-100 (1980).

[13] ZIJL, W. & H. DE BRUIJN, *Continuum Equations for the Prediction of Shell-Side Flow and Temperature Patterns in Heat Exchangers*, Int. J. Heat Mass Transfer, 26, no. 3, pp. 411-424 (1983).

[14] METCALF, M., FORTRAN *optimization*, Acad. Press, London, 1982.

[15] SPALDING, D.B., *Physical and Mathematical Basis of PHOENICS*, CHAM Limited 1982, Technical Report.

SOME SOFTWARE FOR GRAPH THEORETICAL PROBLEMS

R. de BRUIN

0. INTRODUCTION

Problems related to graph theory are numerous. Algorithms solving a great number of problems are widespread but yet often not available in general program libraries. Program libraries as NAG[9], IMSL[6], CALGO[3] or CERN[4] offer little, if any, comfort to the user with some graph theoretical problem. Locally, but not very general, some software is available, e.g. NET, CREW, QUEUS of ACCU (Academisch Computer Centrum Utrecht) and of course numerous specialized micro-computer software (e.g. network planning techniques). A basic difficulty seems to standarise the representation of a graph which can be very extensive. As a good example of a program library which has overcome this diffulty GRADAP[5] may serve, originating from social sciences and developed at the universities of Amsterdam, Groningen and Nijmegen. GRADAP consists of an organised set of programs with a standard input file. It has an user-oriented, SPSS-like, language structure and a direct transfer to SPSS. Moreover it has facilities to generate new graphs (subgraphs, partial graphs, condensed graphs, etc.) from the original grahp and facilities to analyse graphs. The GRADAP subprograms that analyse graphs are:

ADJACENCY	computes the adjacency matrix (neighbours)
DISTANCE	computes the distance matrix of directed or undirected graphs
REDUCE	reduces graphs stepwise to the nill graph according to a criterion specified by the user
RUSH	computes thc rush of points based on a model where each point sends a unit of flow to each of its contacts. The rush refers to the proportion of flow that passes that point
SUBGRAPHS	detects several types of subgraphs such as blocks, trees N-cliques, NM-clans and N-clubs

CENTRALITY both CENTRALITY and RUSH are aids to analyse problems related to the following

(on point level)

* adjacency
* betweenness
* distance
* status (number of points to which it is indirectly connected within a certain path length)

(on graph level)

* integrated graphs
* unipolarity
* centralization

Hence one can state that simple analysis can be performed on graphs using GRADAP. However often the analysis of graphs is more complex in at least two ways: the problem is less simple than those enumerated above or the user wishes more complex output. For both cases an example is given. In section 2 an acyclic subgraph problem is examined while section 3 discusses some general network analysis techniques.

1. ACYCLIC SUBGRAPH PROBLEM

The acyclic subgraph problem is introduced by the following example (LENSTRA[7]):

> A certain number of persons has to rank n alternatives according to desirability. To this end each of the persons determines the ranking he prefers. Let a_{ij} be the number of persons putting alternative i before alternative j, for $1 \leq i, j \leq n$. This asks for a ranking which minimizes the number of neglected preferences.

In this example all possible alternatives come to turn. More general, some preferences, or even many, are not known.

DEFINITION. A graph (I,R) consists of

* a finite set $I = \{x_1, x_2, \ldots, x_n\}$, whose elements are called vertices,
** a subset R of the Cartesian product $I \times I$, the elements of which are called edges.

Furthermore we assume that the graph is anti-reflexive, i.e.

$$(x_i,x_i) \notin R \text{ for all } x_i \in I,$$

and anti-symmetric, i.e.

$$\text{if } (x_i,x_j) \in R \text{ then } (x_j,x_i) \notin R.$$

We can state the acyclic subgraph problem more precisely as follows:

(1.1) Let (I,R) be a graph and $a: R \to \mathbb{R}_0^+$ a weight function. Determine an acyclic subgraph (I,S) such that

(1.1) $$\sum_{(i,j)\in R\setminus S} a_{ij}$$

is minimal.

DEFINITION. Let $i_j \in I$, $1 \le j \le n$ and $(i_j,i_{j+1}) \in R$, $1 \le j \le n-1$, $n > 1$. A graph (I,R) is called acyclic if there does not exist a path, say $((i_1,i_2),(i_2,i_3),\ldots,(i_{n-2},i_{n-1}),(i_{n-1},i_n))$, such that $i_1 = i_n$, i.e. (I,R) is a graph without cycli.

For our purpose it is more convenient to rewrite problem (1.1) as follows:

(1.2) Let (I,R) be a graph and $a: R \to \mathbb{R}_0^+$ a weight function.

Minimize

$$\sum_{(i,j)\in R} a_{ij}x_{ij},$$

subject to the constraints

(1.2a) for all cycli, say $((i,j),(j,k),\ldots,(p,q),(q,i))$,

$$x_{ij} + x_{jk} + \ldots + x_{pq} + x_{qi} \ge 1$$

(1.2b) $x_{ij} \in \{0,1\}$ for all $(i,j) \in R$, where

$x_{ij} = 0$ denotes $(i,j) \in S$ and

$x_{ij} = 1$ denotes $(i,j) \in R\setminus S$.

In words this means: remove from all cycli in (I,R) at least one edge in such a way that the total weight of the subgraph (I,S) is maximal. In this manner we find a ordering on I. Hence, even more informal, one can say:

determine a ordering which "resembles" R as closely as possible.

EXAMPLE. Figures (1.1a) and (1.1b) show, somewhat schematic, the graph (I,R) respectively the maximal acyclic subgraph (I,S) as described above.

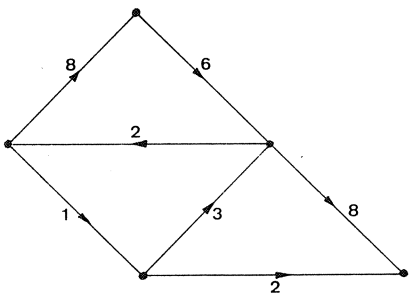


Fig. (1.1a). The graph (I,R)

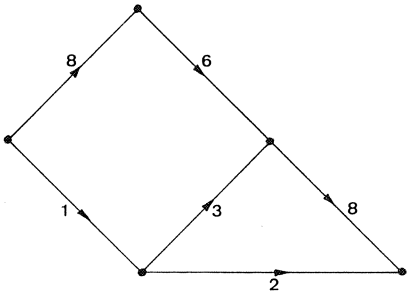


Fig. (1.1b). The maximal acyclic subgraph (I,S)

Replacing constraints (1.2b) by

$$0 \leq x_{ij} \leq 1, \quad (i,j) \in R,$$

yields a Linear Programming problem. In the sequel we shall refer to this problem as the LP-problem. Unfortunately an optimal solution to the LP-problem need not be a solution to the discrete linear programming problem (1.2). As an example to this phenomenon consider the following (LENSTRA[7]):

EXAMPLE. Let the graph (I,R) be a tournament (i.e. I is finite, $R \cap R^{-1} = \emptyset$ and for all $i \in I$: for all $j \in I$: $i \neq j \Rightarrow (i,j) \in R \cup R^{-1}$). It can be shown that tournaments (I,R), $I = \{1,2,\ldots,n\}$, exist for which, for sufficiently large n,

$$|R \backslash *| \geq 1/3\binom{n}{2}$$

for all orderings $*$. This implies that at least $1/3\binom{n}{2}$ edges are to be removed to achieve ordering. So for every feasible solution, with e.g. $a_{ij} = 1$, $(i,j) \in R$, this yields

$$\max \sum_{(i,j) \in R} a_{ij} x_{ij} \leq 2/3\binom{n}{2}.$$

Letting $x_{ij} = 2/3$ for all $(i,j) \in R$, we find a non-discrete optimal solution to this example problem.

Although the solution to the LP-problem may not be the discrete solution we are searching for, it can be used to determine this discrete solution to problem (1.2).

ALGORITHM TO THE ACYCLIC SUBGRAPH

The proposed algorithm to find an acyclic subgraph of maximal weight consists basically of three parts:

* Search for strongly connected components
** Search for a heuristic solution
*** Search for an optimal solution

These subdivisions are treated in the sequel:

* Search for strongly connnected components

DEFINITION. Let (I,R) be a finite graph. Suppose (J,C) is a maximal subgraph of (I,R) such that for all $i,j \in J$ there exist paths in (J,C) from i to j and from j to i, then (J,C) is called a strongly connected component.

So, if (I,R) contains a cycle than this cycle is contained in one strongly connected component, if (I,R) contains no cycli the strongly connected components will consist of one vertex only. The problem (1.2) is now solved for the strongly connected components separately. Strongly connected components consisting of but one vertex are disregarded. The following depth first algorithm from AHO, HOPCROFT and ULLMAN[1] is used to detect the connected components:

```
proc DFS = (vertex v) void:
           (vertex w; mark[v] := 1;
            for each vertex w being a neighbour of v
                do if mark[w] = 0
                   then DFS(w)
                   fi
                od ;
            number[v] := count;
            count := count + 1);
```

First perform a depth first search (DFS) on the graph (I,R) and number the vertices in order of their completion of the recursive calls. After that perform a DFS on (I,R^{-1}), the reversed directed graph, starting from the highest numbered vertex. If DFS does not reach all vertices then start the next DFS from the highest numbered remaining vertex, i.e. detect a new tree. Each such tree in the resulting spanning forest is a strongly connected component.

EXAMPLE. Fig. (1.2a) shows, schematically, the original graph. Problem (1.2) is solved for the strongly connected components as depicted in Figures (1.2b) and (1.2c) and hence for the problem (1.2) on the original graph. Fig. (1.2d) shows the remainder which is a trivial case.

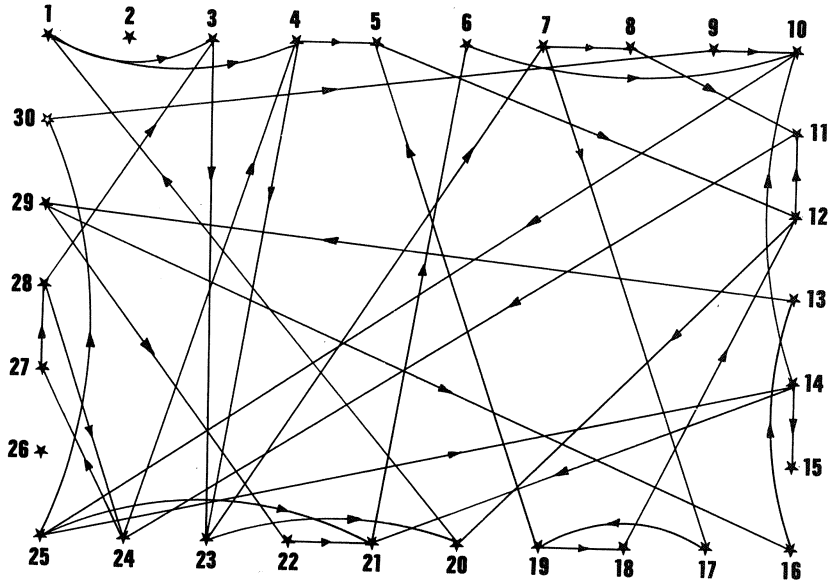


Fig. (1.2a). The original graph.

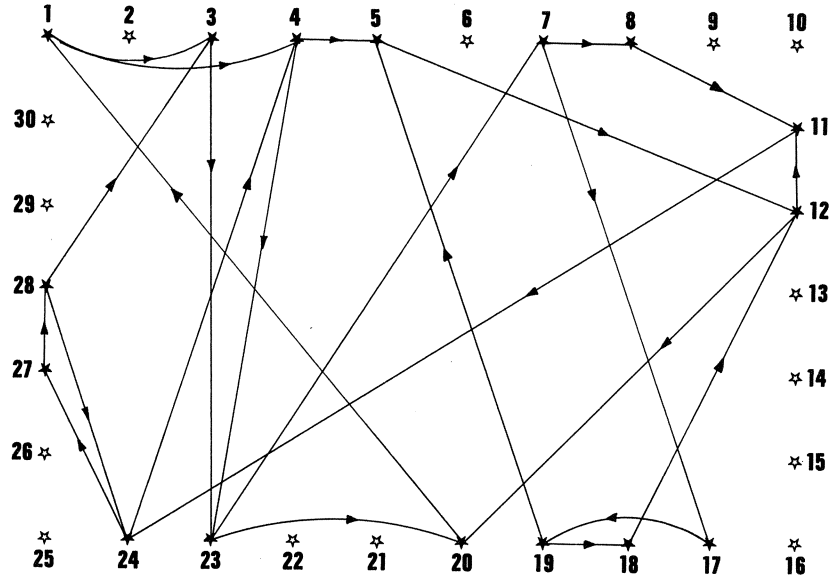


Fig. (1.2b). A strongly connected component.

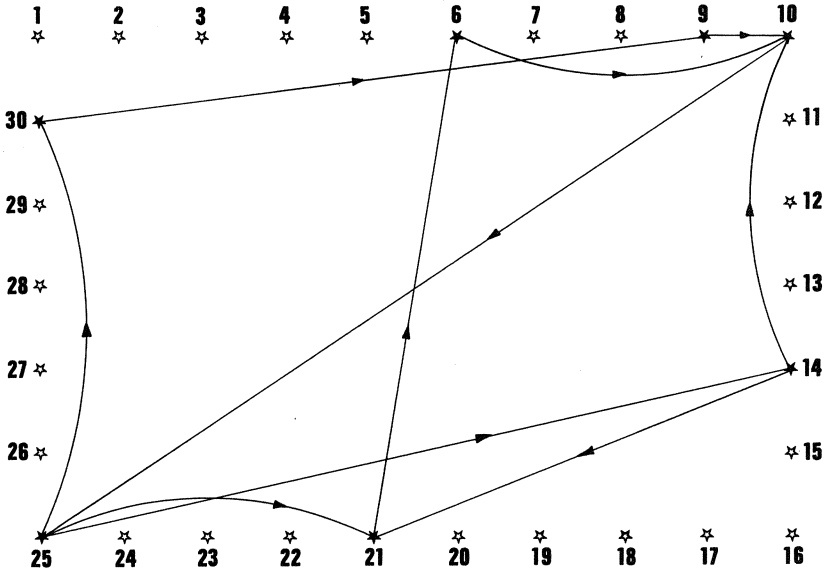


Fig. (1.2c). A strongly connected component.

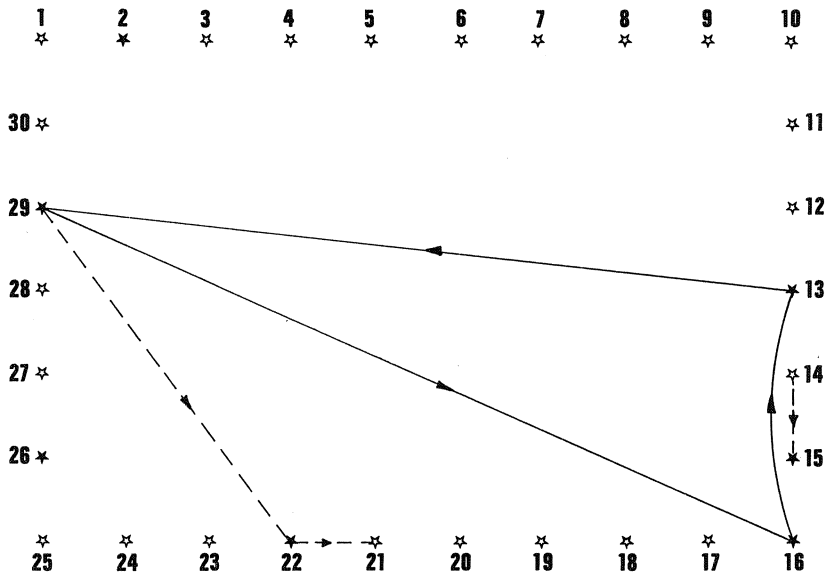


Fig. (1.2d). The remaining, trivial, strongly connected components.

** Search for heuristic solution

As an initial solution for the recursive algorithm in *** we use the following heuristic solution: let

$$S = \sum_{(i,j)\in R\setminus S} a_{ij}.$$

Initially set S = 0.
Start in an arbitrary vertex if there is one, otherwise stop - remember strongly connected components consisting of one vertex are disregarded. Now locate a cycle by starting in this vertex - there certainly exists such a cycle. Remove the edge of this cycle with minimal weight, say (v,w). Let $S = S + a_{vw}$. If the remaining edges of v or w are only incoming or outgoing then remove these edges as well. And so on. Remove remaining "loose" vertices. In this manner the remaining subgraph remains strongly connected. Return to the above: Start in an arbitrary vertex,... etc.

EXAMPLE. Consider the graph in Fig. (1.3a). Suppose vertex 1 is a starting point and (1,3,5,2,1) a found cycle. Remove the cheapest edge, i.e. (5,2) and hence ((2,3) and (2,1)) and (1,3). Remove the vertices 1 and 2.

The remaining subgraph is depicted in Fig. (1.3b).

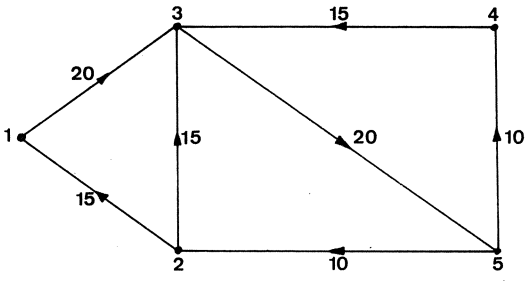


Fig. (1.3a). A strongly connected graph

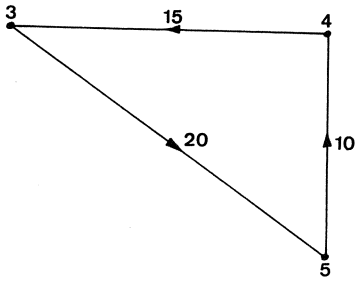


Fig. (1.3b). A strongly connected subgraph of Fig. (1.3a)

*** Search for an optimal solution

Consider the cycli found in the heuristic solution as constraints like formulated in (1.2a). We are now able to solve the LP-problem. Make a copy of the graph and delete all edges (i,j) for which the solution vector to the LP-problem contains $x_{ij} \geq 0$. If the remaining subgraph contains cycli then create extra constraints like described in **.
If the solution vector of the LP-problem contains only integer values then the problem is solved else a branch-and-bound technique is used in the following sense:

Branch-rule: Let x_{ij} and a_{ij} be single indexed for simplicity reasons. Consider the constraints due to the cycli $B\bar{x} \geq \bar{1}$, where $b_{kl} \in \{0,1\}, k=1,\ldots,m$, $1 = 1,\ldots,n$ are the elements of matrix B. Let $T \subset \{1,2,\ldots,n\}$ be such that for all $t \in T$

$$\sum_{k=1}^{m} b_{kt} = \max_{k} \sum_{k=1}^{m} b_{kl}.$$

Let $t^* \in T$ be such that $a_t^* \leq a_t$ for $t \in T$. Consider two subproblems where $x_t^*=1$ or $x_t^* = 0$. One of the values may violate the constraints. If so, then set x_t^* to the other value and expand further. If $x_t^* = 1$ according to the branch-rule then restrictions, due to the cycli including t^* are superfluous.

Bound-rule. A branch is bounded if the solution is an integer solution and hence no better solution in that branch is to be expected. Moreover a branch is not expanded further if the non-integer solution in the present point is worse then the best integer solution found so far. As a third bound-rule we state that a non-integer solution is not expanded any further if it differs only a certain small value from the best found integer solution.

DISCUSSION

The problem is NP-complete. However for moderate numbers of edges in the strongly connected components the chance of running across a non-integer solution is unusual. Since in the LP-problem the objective function is linear (as the constraints are) an optimal non-integer solution should have at least as much independent constraints - due to the cycli - as it has edges to be a better solution than an integer solution. This means a graph with m edges should contain at least m independent cycli. As the tournament example has shown, this is quite possible but for our object-graphs (number of vertices about 40, number of edges about 100) not very likely to occur. It is for this reason that we did not bother to use sharper and more complicated bounding rules as can be found in LENSTRA[7].

2. TWO GENERAL NETWORK ANALYSIS TECHNIQUES

This section discusses briefly the following two basic variants of network planning techniques out of many: CPS (Critical Path Scheduling - based on deterministic duration of time) and PERT (Program Evaluation and Review Techniques - based on stochastic duration of time). CPS and PERT are intended to accompany a project for which a network planning is present (not for the planning itself, although it can be used to check the network). Basically the programs are divided into three items:

* check of the network and computation of the critical path which defines the duration of the total project,
** continuation check,
*** various representations of the network or parts of it, e.g. Gantt-charts, divisions according to earliest start, slack time, account, etc..

The continuation check covers a smaller period of time than the project as a whole, moreover only specific subgraphs (subnetworks) can be regarded. During this period of time the continuation check gives information about e.g. which activities should be completed, can be completed, should be started or can be started. Various continuation checks can be executed without recomputing the network. The presentations of items ** and *** depend highly on the programmers taste and the costumers demand. This, laborious though straightforward, part will not be discussed here though some examples are given at the end of this section (Figure (2.4)). At the Rijksuniversiteit of Groningen the program PERT of CERN [4] is favourite whenever the activities have stochastic duration of time. For the case that activities have deterministic periods of time a CPS-program called NETWERK has been developed (based on BOSMAN & WEZEMAN [2]). The latter is more easy to use and faster then the former which has more facilities.

CRITICAL PATH SCHEDULING

Consider a network, that is a finite set of points, $I = \{i_0, i_1, \ldots, i_n\}$, and a ordering relation on I. This yields that a network has no cycli. Usual the ordering relation is something like: activity i_k, $1 \leq k \leq n-1$, is preceded by one or more activities and followed by one or more activities. The network has a logical starting point: i_0 (which is not preceded by any activity) and a final point x_k (and hence not followed by any activity). In most cases a first and last logical activity should be present in the network. Between two points there exists at most one activity. This restriction can be bypassed by the use of dummy activities; these are activities with zero duration (see Figures (2.1a) and (2.1b)).

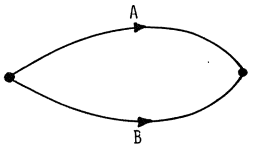


Fig. (2.1a) duplicity of activities.

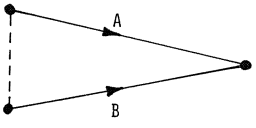


Fig. (2.1b) Use of dummy activity to avoid duplicity.

Dummy activities are also used as depicted in Figures (2.2a) and (2.2b). They describe two distinct cases: Fig. (2.2a) shows the case where activities B and D can start whenever activities A and C are completed, while Fig. (2.2b) shows the case where activity B can start after completion of A and C, as above, but D may start whenever activity C is completed.

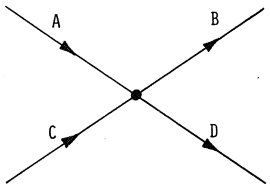


Fig. (2.2a) starting points A and B depend on completion of A and C.

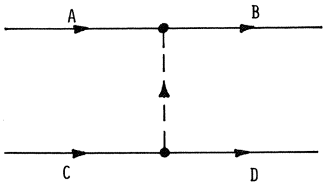


Fig. (2.2b) Starting point of B depends on completion of A and C whilst starting point on D depends on completion of C only.

A path from the starting point to the final point of a network with the longest duration of time is called critical and defines the duration of the whole project. A summary of the most important items of a network planning and their definitions are given below. T_{ij} denotes the duration of time for the activity $(i,j) \in R$.

* earliest possible starting moment: T_i^e

$$T_j^e = \max(T_i^e + T_{ij}), \quad j \neq 0, \quad (i,j) \in R$$

$$T_0^e = 0$$

* last acceptable moment of completion: T_j^l

$$T_j^l = \min(T_j^l - T_{ji}), \quad j \neq n, \quad (j,i) \in R$$

$$T_n^l = \lambda \text{ (total duration of the project)}$$

* earliest possible moment of completion: $T^e_{(i,j)}$

$$T^e_{(i,j)} = T_i^e + T_{ij}, \quad (i,j) \in R$$

* last acceptable starting moment: $T^e_{(i,j)}$

$$T^l_{(i,j)} = T^l_j - T_{ij}, \quad (i,j) \in R.$$

It is clear that on a critical path the earliest possible starting moment and the last acceptable starting moment coincide. Any delay of the critical activities results in a delay of the total project. Activities not on the critical path do have some margin or slack as when to start this activity. The various slack times are defined as follows:

* total slack time: $S^t_{(i,j)}$

$$S^t_{(i,j)} = T^l_j - T^e_i - T_{ij} \quad (i,j) \in R$$

* free slack time: $S^f_{(i,j)}$

$$S^f_{(i,j)} = T^e_j - T^e_i - T_{ij} \quad (i,j) \in R$$

* dependent slack time: S^d_j

$$S^d_j = T^l_j - T^e_j,$$

* independent slack time: $S^i_{(i,j)}$

$$S^i_{(i,j)} = T^e_j - T^l_i - T_{ij}, \quad (i,j) \in R$$

EXAMPLE. Fig. (2.3) depicts a simple network with dummy activities. Table (2.1) gives some corresponding caracteristics. Clearly the critical path is ((0,3), (3,4)).

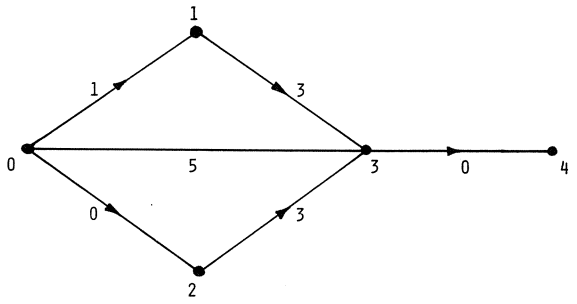


Fig. (2.3) An example network.

$T^e_0 = 0$	$T^l_0 = 0$	$S^t_{(0,1)} = 1$	$S^f_{(0,1)} = 0$
$T^e_1 = 1$	$T^l_1 = 2$	$S^t_{(0,2)} = 2$	$S^f_{(0,1)} = 0$
$T^e_2 = 0$	$T^l_2 = 2$	$S^t_{(0,3)} = 0$	$S^f_{(0,3)} = 0$
$T^e_3 = 5$	$T^l_3 = 2$	$S^t_{(1,3)} = 1$	$S^f_{(1,3)} = 1$
$T^e_4 = 5$	$T^l_4 = 5$	$S^t_{(2,3)} = 2$	$S^f_{(2,3)} = 2$
		$S^t_{(3,4)} = 0$	$S^f_{(3,4)} = 0$

Table (2.1). Some caracteristics of the network as depicted in Fig. (2.3).

All the network characteristics depend on T^e_i and T^l_i. Here an example algorithm is given to compute these values. Starting from the final point the entire network is passed through. The earliest possible starting moment of the starting point, T^e_0, is set to zero. A reverse algorithm finds the last acceptable starting moments, T^l_i, where $T^l_n = \lambda$.
The duration of the activities are represented by t[i], i = 1,...,n, as begin[i] and end[i] stand for the begin and end point of activity i. These points can be enumerated in arbitrary order. Initially te[ ] is set to -1.

EXAMPLE algorithm:

```
proc EST = (int j) int:
     (int start, rh, r;
      start := begin[j]; r := 0;
      for i to n
      do if start = end[i]
         then rh := if te[i] = -1
                    then EST(i) + t[i]
                    else te[i] + t[i]
                    fi;
              if rh > r then r := rh fi
         fi
      od;
      te[j] := r; r);
```

The main difference between PERT and CPS is, as stated, the deterministic duration of time of activities in a CPS-procedure; they are estimated and given by the user.

PROGRAM EVALUATION AND REVIEW TECHNIQUES

A PERT program considers the duration of an activity as a stochastic variable. The user supplies the duration of every activity, as in a CPS-program, say m_{ij}, but also an optimistic and a pessimistic estimation (a_{ij} respectively b_{ij}). PERT then assumes the duration of an activity can be approximated by a beta-distribution based on the three parameters a_{ij}, b_{ij} and m_{ij}. The standard deviation and the expectation are approximated by

$$\sigma = (b_{ij}-a_{ij})/6,$$

$$E(T_{ij}) = (a_{ij}+4m_{ij}+b_{ij})/6.$$

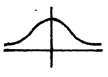
Next the earliest possible starting moment and the last acceptable starting moment can be computed in the same way as in the CPS-program but based on the stochastic values $E(T_{ij})$, $(i,j) \in R$.
If we assume that

* the network contains a great number of activities
** the durations of the activities are independent

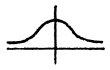
then the variation in the various moments can be regarded as normal-distributed. In case of a great number of activities this seems reliable for the total duration of time of the project - not for the duration of the separate activities. Exceeding the duration of time of an activity does not necessarily imply the exceeding of the duration of the total project like in a CPS-program.

EXAMPLE. The probability that some target date, T_i^t, is made has a normal distribution:

$$P(X \leq \frac{T_i^t - T_i^e}{\sigma(T_i^e)}) =$$



The total slack per activity is normal distributed:

$$P(X \leq \frac{T_j^l - T_i^e - T_{ij}}{\sqrt{\sigma^2(T_j^l)+\sigma^2(T_i^e)+\sigma^2(T_{ij})}}) =$$ 

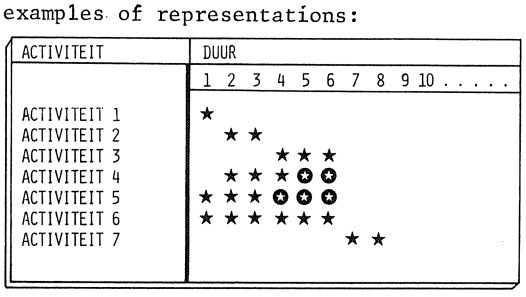
Consider again the PERT assumptions. Errors introduced by these assumptions are analysed by MacCRIMMON & RYAVEC [8]. Roughly this amounts to:

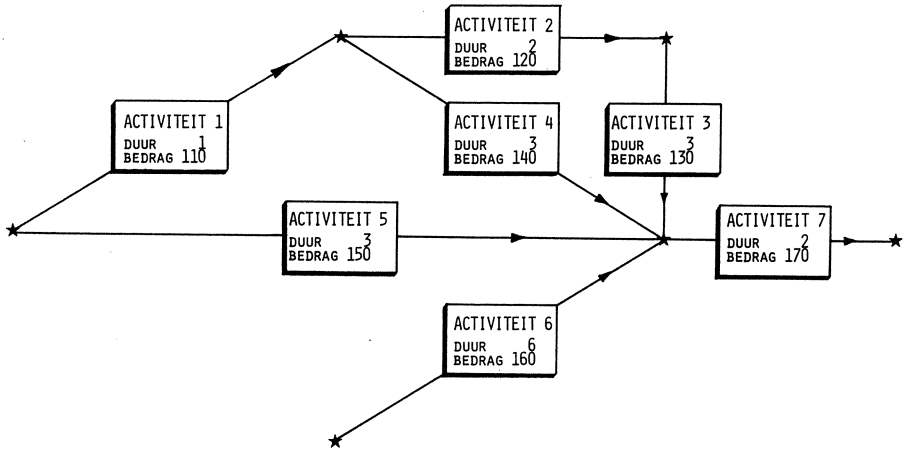
* The beta-distribution should not be too oblique, i.e. if $(a_{ij}+b_{ij})/2-m_{ij} < (b_{ij}-a_{ij})/6$ then the error in E is smaller than 0.05 $(b_{ij}-a_{ij})$.
* If the errors in the estimations a_{ij}, b_{ij} and m_{ij} are less then 10% then the error in E is smaller than 0.02 $(b_{ij}-a_{ij})$ and the error made in the standard deviation is less than 0.05 $(b_{ij}-a_{ij})$.
* Assuming $\sigma=(b_{ij}-a_{ij})/6$ yields a maximal error in E and σ^2 of about 0.12 $(b_{ij}-a_{ij})$.

Furthermore it is implicitly assumed that:

* The critical path is that much longer than the other paths that the probability that one of the others will turn critical can be neglected. One considers the probability an activity is on the critical path (similar to slack times in CPS) by computing the network with various values of $E(T_{ij})$. This can be done by recomputing the network a given number of times (laborious), or by using a Monte Carlo simulation, but more often an even rougher method is applied since the error, whenever the structure of the network is unfavourable, can be 15% and hence an accurate method seems superfluous.
* On the critical path there are enough activities to assume that the variations in the various moments are normally distributed.

Some examples of representations:





PROJECT: 123456
NETWERK - VOORBEELDJE

ALS UITGANGSPUNT IS GEKOZEN: WEEK NR 4
DE KOMENDE PERIODE HEEFT 4 WEKWEKEN

HIERIN MOET AAN DE VOLGENDE ONDERDELEN GEWERKT WORDEN:

		VS	LS	VV	LV
031006	ACTIVITEIT 6	1	1	6	6
021004	ACTIVITEIT 4	2	4	4	6
021003	ACTIVITEIT 3	4	4	6	6
031007	ACTIVITEIT 7	7	7	8	8

HIERIN KAN NOG OF AL AAN DE VOLGENDE ONDERDELEN GEWERKT WORDEN:

		VS	LS	VV	LV
031005	ACTIVITEIT 5	1	4	3	6

HIERIN MOETEN DE VOLGENDE ONDERDELEN VOLTOOID WORDEN OF ZIJN:

		VS	LS	VV	LV
031005	ACTIVITEIT 5	1	4	3	6
031006	ACTIVITEIT 6	1	1	6	6
021004	ACTIVITEIT 4	2	4	4	6
021003	ACTIVITEIT 3	4	4	6	6

HIERIN KUNNEN BOVENDIEN DE VOLGENDE ONDERDELEN VOLTOOID WORDEN:

		VS	LS	VV	LV

Fig. (2.4) Three examples of representations of networks.

REFERENCES

[1] AHO, A.V., J.E. HOPCROFT & J.D. ULLMAN, *Data Structures & Algorithms*, Reading, Mass., Addison-Wesley, 1983.

[2] BOSMAN, A. & K. WEZEMAN, *Netwerkplanningtechnieken*, Maandblad voor Accountancy en Bedrijfshuishoudkunde, 1967.

[3] CALGO Collected Algorithms from ACM, Association for Computing Machinery Inc., New York, 1978.

[4] CERN Computer Centre Program Library, CERN, Geneva, 1980.

[5] GRADAP User Manual (vol. 1,2), Ed. F.N. Stokman & F.J.A.M. van Veen, Universities of Amsterdam, Groningen and Nijmegen, 1981.

[6] IMSL Library edition 9, IMSL, Houston, 1982.

[7] LENSTRA, H.W., *The Acyclic Subgraph Problem*, BW 26/73, Mathematical Centre, Amsterdam, 1973.

[8] MACCRIMMON, K.R. & C.A. RYAVEC, *An Analytical Study of the PERT assumptions*, The Rand Corporation, Santa Monica, 1962.

[9] NAG Fortran Library Manual mark10, Numerical Algorithms Group, Oxford 1983.

LEAST-SQUARES B-SPLINE SURFACE RECONSTRUCTION IN TOMOGRAPHY

R.H.J. GMELIG MEYLING

This paper presents an accurate and C^2*-smooth bicubic* B*-spline surface reconstruction of a closed* 3*-dimensional object from data given at different photo levels. The approximation technique is restricted to so called starlike objects, which can be expressed in spherical coordinates. Boundary conditions are essential to guarantee the periodicity of the solution and a correct derivative behaviour at the upper and lower pole. A weighted least-squares method eliminates data errors caused by the digitizing of the surface contours. Some practical applications of the algorithm will be treated in detail, such as the calculation of the volume, the surface and the center of gravity and the determination of the outward directed normal vector in an arbitrary point of the surface.*

1. INTRODUCTION

In tomography one produces a number of parallel pictures of an organ in order to study its geometry. In such a photo the contour-line of the organ is visible as the boundary between darker and lighter areas. With the help of a digitizer the coordinates of a large amount of points on this curve are stored in the computer. From this information at different photo levels a complete reconstruction of the 3-dimensional object is required, together with some special properties of the organ, such as volume, surface and center of gravity.

In this contribution we restrict the class of closed 3-dimensional objects to the so called *starlike* surfaces. This means there is a point $\vec{m}$ in the interior of the organ, such that any straight line through this point meets the surface precisely twice. Note that a starlike condition is weaker than imposing convexity of the object. Some organs, satisfying this condition, are the eye, tumors and the boundary surface of the heart.

For a tumor 6 contour-lines are given in fig. 1.1.

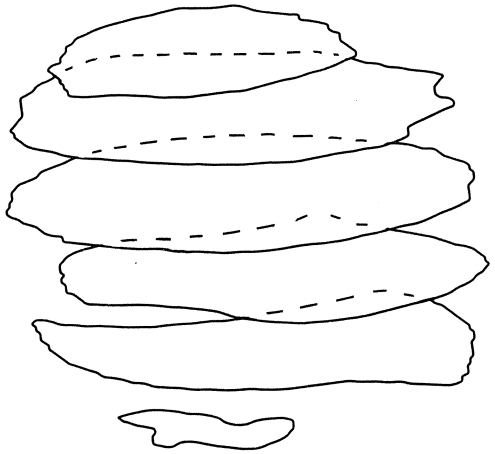


Fig. 1.1. Contour-lines of a tumor.

A starlike surface T can be represented by a spherical coordinate system r,ϕ,θ, with origin in $\vec{m}$ and pole direction $\vec{n}$. Usually this vector $\vec{n}$ will be orthogonal to the photo planes and $\vec{m}$ will be chosen so that the line $\vec{m} + \lambda\vec{n}$, $\lambda \in \mathbb{R}$ passes through the interior of all photo curves. A vector $\vec{l}$, orthogonal to $\vec{n}$, is used to normalize the angle ϕ. In fig. 1.2 this coordinate system is given.

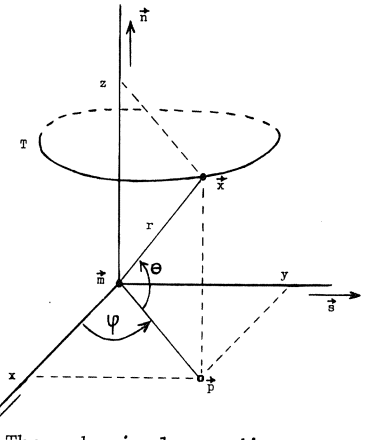


Fig. 1.2. The spherical coordinate system r,ϕ,θ.

The components (x,y,z) of a point $\vec{x}$ are the usual coordinates in the direction of the vectors $\vec{l}$, $\vec{s} = \vec{n} \times \vec{l}$ and $\vec{n}$ respectively. A point $\vec{x}$ on the surface T is represented by the length r of the vector $\vec{x} - \vec{m}$, the angle ϕ between $\vec{l}$ and the projection $\vec{p}$ of $\vec{x} - \vec{m}$ on the xy-plane and the angle θ between $\vec{p}$ and $\vec{x} - \vec{m}$. We call $\theta = \pi/2$ the *upper pole* and $\theta = -\pi/2$ the *lower pole* of the coordinate transformation.

A point $\vec{x}$ can be expressed in terms of r,ϕ and θ:

$$\vec{x} = \vec{m} + r.(\cos\phi\cos\theta, \sin\phi\cos\theta, \sin\theta)^T. \tag{1.1}$$

For a point $\vec{x}$ on the surface T the length $r(\phi,\theta)$ of $\vec{x} - \vec{m}$ is a function of ϕ and θ, defined on a rectangular domain $R = [0,2\pi] \times [-\pi/2,\pi/2]$. The contour-lines of the organ in the parallel photo planes correspond to curves in the (ϕ,θ)-rectangle. If T is a starlike surface and $\vec{m}$ is properly chosen, these curves do not intersect. In fig. 1.3 the (ϕ,θ)-curves, corresponding to the contour-lines of fig. 1.1, are shown.

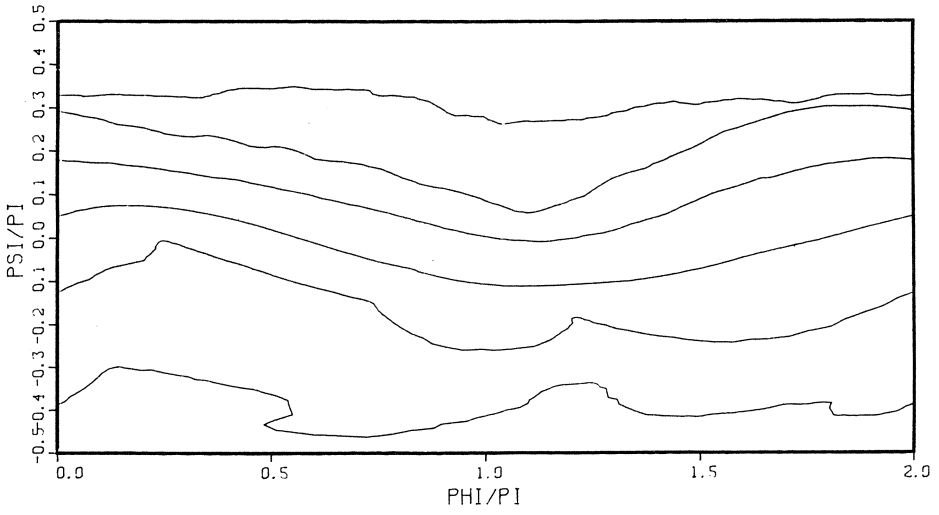


Fig. 1.3. Data curves in the (ϕ,θ)-rectangle, with the line $\vec{m} + \lambda\vec{n}$ passing through the interior of all 6 photo contours.

The data values of the function r on these (ϕ,θ)-curves will be approximated by spline functions in least-squares norm. The behaviour of the curves in the rectangle R is important to verify whether the origin $\vec{m}$ is properly chosen and to examine the data distribution, which has a direct effect on

the approximation quality.

2. SURFACE REPRESENTATION

In this section we derive conditions for the function $r(\phi,\theta)$ to guarantee that (1.1) represents a closed C^2 - smooth surface. It is clear that the function r should be continuous over the domain R. We consider the function values of r at the boundaries of the domain. The periodicity along the vertical boundaries implies

$$(2.1) \qquad r(0,\theta) = r(2\pi,\theta), \quad \theta \in [-\pi/2,\pi/2].$$

Closing the surface at the poles $\theta = \pm\pi/2$ gives the conditions

$$(2.2) \qquad r(\phi,\pi/2) = r_1, \quad r(\phi,-\pi/2) = r_2, \quad \phi \in [0,2\pi].$$

The constant value r_1 (resp. r_2) represents the distance from the origin $\vec{m}$ to the upper (resp. lower) pole. They will be used as extra parameters in the least-squares process to approximate the data values in an optimal way.

As a consequence of the medical background of the reconstruction problem the surface $r(\phi,\theta)$ should be smooth.

DEFINITION 2.1. *A surface* T *is of class* C^1 *if it has in every point* $\vec{x} \in T$ *a tangent plane, which varies continuously over the surface.*

The tangent plane is spanned by the two vectors

$$(2.3) \qquad \begin{aligned} \vec{x}_\phi(\phi,\theta) &= r_\phi(\phi,\theta).(\cos\phi\cos\theta,\sin\phi\cos\theta,\sin\theta)^T \\ &\quad + r(\phi,\theta).(-\sin\phi\cos\theta,\cos\phi\cos\theta,0)^T \\ \vec{x}_\theta(\phi,\theta) &= r_\theta(\phi,\theta).(\cos\phi\cos\theta,\sin\phi\cos\theta,\sin\theta)^T \\ &\quad + r(\phi,\theta).(-\cos\phi\sin\theta,-\sin\phi\sin\theta,\cos\theta)^T, \quad (\phi,\theta) \in R. \end{aligned}$$

In order to obtain a C^1-smooth surface the first derivatives r_ϕ and r_θ should be continuous over R. Along the vertical boundary we impose

$$(2.4) \qquad r_\phi(0,\theta) = r_\phi(2\pi,\theta), \quad \theta \in [-\pi/2,\pi/2].$$

The periodicity of r_θ follows directly from condition (2.1).

Because $\vec{x}_\phi(\phi,\pm\pi/2) = \vec{0}$ the parameters ϕ and θ are no longer admissible

in the poles and the vectors (2.3) are linearly dependent and therefore span no tangent plane.

Let us now consider the upper pole $\theta = \pi/2$ in detail. The tangent plane is spanned by the vectors

$$(2.5) \qquad \vec{x}_\theta(\phi,\pi/2) = (-r_1\cos\phi,-r_1\sin\phi,r_\theta(\phi,\pi/2))^T, \quad \phi \in [0,2\pi].$$

For all values $\phi \in [0,2\pi]$ the vector (2.5) should be in the same tangent plane, which is not necessarily the case for arbitrary functions r.

Define two independent vectors in the tangent plane

$$(2.6) \qquad \vec{x}_\theta(0,\pi/2) = (-r_1,0,r_\theta(0,\pi/2))^T$$

$$\vec{x}_\theta(\pi/2,\pi/2) = (0,-r_1,r_\theta(\pi/2,\pi/2))^T.$$

The following determinant has to vanish, since the vectors $\{\vec{x}_\theta(0,\pi/2), \vec{x}_\theta(\pi/2,\pi/2), \vec{x}_\theta(\phi,\pi/2)\}$ are linearly dependent.

$$(2.7) \qquad \begin{vmatrix} \vec{x}_\theta(0,\pi/2)^T \\ \vec{x}_\theta(\pi/2,\pi/2)^T \\ \vec{x}_\theta(\phi,\pi/2)^T \end{vmatrix} = \begin{vmatrix} -r_1 & 0 & r_\theta(0,\pi/2) \\ 0 & -r_1 & r_\theta(\pi/2,\pi/2) \\ -r_1\cos\phi & -r_1\sin\phi & r_\theta(\phi,\pi/2) \end{vmatrix} = 0, \quad \phi \in [0,2\pi].$$

Since we may assume that $r_1 \neq 0$ this leads to

$$(2.8) \qquad r_\theta(\phi,\pi/2) = r_\theta(0,\pi/2)\cos\phi + r_\theta(\pi/2,\pi/2)\sin\phi, \quad \phi \in [0,2\pi].$$

The first derivative r_θ in the upper pole is a *trigonometric function* with period 2π depending on given values $r_\theta(\phi,\pi/2)$ in $\phi = 0,\pi/2$. A similar condition holds for the lower pole.

It is clear that the tangent plane (2.5) in the upper pole is the limit of the tangent planes (2.3) in neighbouring points, since the length of the vector $\vec{x}_\phi(\phi,\theta)$ approaches zero for $\theta \uparrow \pi/2$.

Summarizing the previous considerations we can state

THEOREM 2.1. *The function* $r(\phi,\theta)$ *will represent a closed starlike surface of class* C^1 *if* r *and its first derivatives* r_ϕ, r_θ *are continuous over* R *and if, in addition to* (2.1), (2.2) *and* (2.4),

$$(2.9) \qquad r_\theta(\phi,\pm\pi/2) = r_\theta(0,\pm\pi/2)\cos\phi + r_\theta(\pi/2,\pm\pi/2)\sin\phi, \quad \phi \in [0,2\pi].$$

From (2.9) follows immediately

$$\vec{x}_\theta(\phi,\pm\pi/2) = -\vec{x}_\theta(\pi+\phi,\pm\pi/2), \quad \phi \in [0,\pi]. \tag{2.10}$$

The values of $r_\theta(0,\pm\pi/2)$ and $r_\theta(\pi/2,\pm\pi/2)$ determine the tangent plane in the poles and will be treated as extra parameters in the approximation procedure.

In this paper we require that the approximating surface has also continuous second order derivatives.

DEFINITION 2.2. *A surface* T *is of class* C^2 *if the second order derivatives of the vector* $\vec{x} \in T$ *exist and are continuous in every point of the surface, with respect to local admissible parameters.*

For all points on the surface, with the exception of the poles, this simply means that $\vec{x}_{\phi\phi}, \vec{x}_{\phi\theta}$ and $\vec{x}_{\theta\theta}$ must be continuous, implying $r_{\phi\phi}, r_{\phi\theta}$ and $r_{\theta\theta}$ to be continuous.

The periodicity condition along the vertical boundaries is given by

$$r_{\phi\phi}(0,\theta) = r_{\phi\phi}(2\pi,\theta), \quad \theta \in [-\pi/2,\pi/2]. \tag{2.11}$$

The periodicity of $r_{\phi\theta}$ and $r_{\theta\theta}$ is a consequence of condition (2.4) on r_ϕ and r_θ.

In the upper pole $\theta = \pi/2$ we introduce a new pair of admissible parameters

$$\xi = (\pi/2-\theta)\cos\phi, \quad \eta = (\pi/2-\theta)\sin\phi. \tag{2.12}$$

The second order derivative $\vec{x}_{\psi\psi}$ in an arbitrary direction ϕ with

$$\psi = \xi\cos\phi + \eta\sin\phi = \pi/2 - \theta, \tag{2.13}$$

is given by

$$\vec{x}_{\psi\psi} = \cos^2\phi\vec{x}_{\xi\xi} + 2\sin\phi\cos\phi\vec{x}_{\xi\eta} + \sin^2\phi\vec{x}_{\eta\eta} = \vec{x}_{\theta\theta}(\phi,\pi/2). \tag{2.14}$$

For a fixed value of ϕ these derivatives with respect to θ are defined in the pole. We find for $\vec{x}_{\xi\xi}, \vec{x}_{\xi\eta}, \vec{x}_{\eta\eta}$, expressed by (2.14) in terms of the $\vec{x}_{\theta\theta}$-derivatives for 3 different angles $\phi = 0,\pi/4,\pi/2$,

$$(2.15)\qquad \vec{x}_{\xi\xi} = \vec{x}_{\theta\theta}(0,\pi/2)$$

$$\vec{x}_{\xi\eta} = -(\vec{x}_{\theta\theta}(0,\pi/2) - 2\vec{x}_{\theta\theta}(\pi/4,\pi/2) + \vec{x}_{\theta\theta}(\pi/2,\pi/2))/2$$

$$\vec{x}_{\eta\eta} = \vec{x}_{\theta\theta}(\pi/2,\pi/2).$$

The length of the vectors $\vec{x}_{\phi\phi}$ and $\vec{x}_{\phi\theta}$ will tend to zero, due to previous assumptions on $r(\phi,\theta)$, as θ approaches the upper pole $\theta = \pi/2$. To ensure the continuity of the second order derivatives for $\theta = \pi/2$, we have the condition

$$(2.16)\qquad \vec{x}_{\theta\theta}(\phi,\pi/2) = \lim_{\theta\to\pi/2} \vec{x}_{\theta\theta}(\phi,\theta) = \vec{x}_{\psi\psi}, \quad \phi \in [0,2\pi].$$

Using (2.14), (2.15) $\vec{x}_{\psi\psi}$ can be rewritten in terms of derivatives with respect to ϕ and θ:

$$(2.17)\qquad \vec{x}_{\theta\theta}(\phi,\pi/2) = \cos\phi(\cos\phi-\sin\phi)\vec{x}_{\theta\theta}(0,\pi/2)+2\sin\phi\cos\phi\vec{x}_{\theta\theta}(\pi/4,\pi/2)$$
$$+ \sin\phi(\sin\phi-\cos\phi)\vec{x}_{\theta\theta}(\pi/2,\pi/2).$$

Differentiating $\vec{x}(\phi,\theta)$ twice with respect to θ and substituting $\theta = \pi/2$ gives

$$(2.18)\qquad \vec{x}_{\theta\theta}(\phi,\pi/2) = (-2\cos\phi r_\theta(\phi,\pi/2),-2\sin\phi r_\theta(\phi,\pi/2),r_{\theta\theta}(\phi,\pi/2)-r_1)^T$$

$$\vec{x}_{\theta\theta}(0,\pi/2) = (-2r_\theta(0,\pi/2),\ 0,\ r_{\theta\theta}(0,\pi/2)-r_1)^T$$

$$\vec{x}_{\theta\theta}(\pi/4,\pi/2) = (-\sqrt{2}\ r_\theta(\pi/4,\pi/2),-\sqrt{2}\ r_\theta(\pi/4,\pi/2),r_{\theta\theta}(\pi/4,\pi/2)-r_1)^T$$

$$\vec{x}_{\theta\theta}(\pi/2,\pi/2) = (0,\ -2r_\theta(\pi/2,\pi/2),r_{\theta\theta}(\pi/2,\pi/2)-r_1)^T$$

for arbitrary ϕ and the chosen values $\phi = 0,\pi/4,\pi/2$ respectively.

If we assume that condition (2.9) holds, a straightforward calculation shows that the vector equation (2.17) is satisfied in the first and second component. The third component of (2.17) reads

$$(2.19)\qquad r_{\theta\theta}(\phi,\pi/2)-r_1 = \cos\phi(\cos\phi-\sin\phi)(r_{\theta\theta}(0,\pi/2)-r_1) +$$
$$+ 2\sin\phi\cos\phi(r_{\theta\theta}(\pi/4,\pi/2)-r_1) + \sin\phi(\sin\phi-\cos\phi)(r_{\theta\theta}(\pi/2,\pi/2)-r_1).$$

Elimination of r_1 from (2.19) leads to an additional restriction on the function r.

THEOREM 2.2. *The function* $r(\phi,\theta)$ *will represent a closed starlike surface of class* C^2 *if* r *and its partial derivatives up to the second order are continuous over* R *and if, in addition to* (2.1), (2.2), (2.4), (2.9) *and* (2.11),

$$(2.20)\qquad r_{\theta\theta}(\phi,\pm\pi/2) = \cos\phi(\cos\phi-\sin\phi)r_{\theta\theta}(0,\pm\pi/2) + 2\sin\phi\cos\phi\, r_{\theta\theta}(\pi/4,\pm\pi/2)$$
$$+ \sin\phi(\sin\phi-\cos\phi)r_{\theta\theta}(\pi/2,\pm\pi/2),\quad \phi \in [0,2\pi].$$

It follows directly from (2.20) that

$$(2.21)\qquad \vec{x}_{\theta\theta}(\phi,\pm\pi/2) = \vec{x}_{\theta\theta}(\pi+\phi,\pm\pi/2),\quad \phi \in [0,\pi].$$

The values $r_{\theta\theta}(\phi,\pm\pi/2)$, $\phi = 0,\pi/4,\pi/2$ are free parameters, determining the second order derivatives in the poles.

3. B-SPLINE APPROXIMATION OF THE FUNCTION R

The function r will be approximated by a linear combination s of bicubic B-splines [6]

$$(3.1)\qquad s(\phi,\theta) = \sum_{i=-1}^{M+1}\sum_{j=-1}^{N+1} c_{ij}B_i(\phi)B_j(\theta)$$

over the rectangular domain R. The functions B_k are the well-known 1-dimensional *normalized cubic B-splines*, defined on a partition $a = u_0 < u_1 < \ldots < u_{P-1} < u_P = b$ of the interval [a,b]. In order to define all B-splines incident on [a,b], we add knots to the left of u_0 ($u_{-3} < u_{-2} < u_{-1} < u_0$) and to the right of u_P ($u_P < u_{P+1} < u_{P+2} < u_{P+3}$). The B-splines of order 4 have the properties

$$(3.2)\qquad \textit{limited support: } B_k^{(j)}(u) = 0,\ u \notin \langle u_{k-2}, u_{k+2}\rangle$$
$$j = 0,1,2,\quad k = -1,\ldots,P+1.$$
$$\textit{normalization: } \sum_{i=-1}^{P+1} B_i(u) = 1,\ \forall u \in [a,b].$$

Every B-spline B_k is numbered corresponding to the index of the center knot u_k in its support. The functions $\{B_k;\ k=-1,\ldots,P+1\}$ form a basis of the (P+3)-dimensional spline space [7] and have continuous derivatives up to the second order. The values of the B-splines will be calculated in an efficient way by a *condensed recursion scheme* [2].

In order to approximate r in two dimensions we use in the domain R a

grid (ϕ_i,θ_j), (i = -3,...,M+3), (j = -3,...,N+3), as given in fig. 3.1.

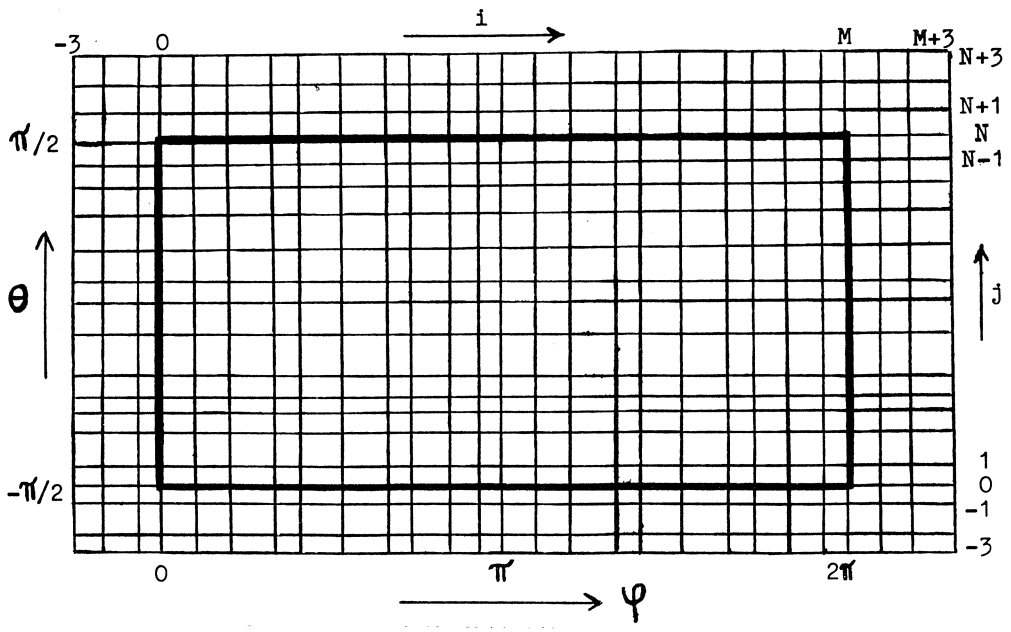


Fig. 3.1. B-spline grid in the rectangular domain R.

The interval $[0,2\pi]$ is divided in M subintervals $[\phi_j,\phi_{j+1}]$

$$\phi_j < \phi_{j+1}, \quad j = -3,\ldots,M+2, \quad \phi_0 = 0, \quad \phi_M = 2\pi \tag{3.3}$$

and the interval $[-\pi/2,\pi/2]$ in N subintervals $[\theta_k,\theta_{k+1}]$

$$\theta_k < \theta_{k+1}, \quad k = -3,\ldots,N+2, \quad \theta_0 = -\pi/2, \quad \theta_N = \pi/2. \tag{3.4}$$

Later considerations will result in some slight restrictions on these subdivisions.

The dimension of the spline space is (M+3)(N+3). The basis functions are

$$B_{ij}(\phi,\theta) = B_i(\phi)B_j(\theta), \quad i = -1,\ldots,M+1, \quad j = -1,\ldots,N+1, \tag{3.5}$$

corresponding to the center knot (ϕ_i,θ_j). The coefficients c_{ij} will be determined in such a way that $s(\phi,\theta)$ is the *least-squares approximation* of the function r, satisfying all extra conditions on r, developed in the previous section. These conditions on the function s will now be translated into equations for the B-spline coefficients.

Conditions (2.1), (2.4), (2.11) ensure the periodicity of $s, s_\phi, s_{\phi\phi}$ in the ϕ-direction. We choose the extra knots ϕ_{-j}, ϕ_{M+j}, (j=1,2,3) as a periodical extension of the interval $[0,2\pi]$ to both sides

$$\phi_j - \phi_{j-1} = \phi_{M+j} - \phi_{M+j-1}, \quad j = -2,\ldots,3. \tag{3.6}$$

The periodicity of the solution s can thus be obtained by identifying the knots (ϕ_j,θ_k) with (ϕ_{M+j},θ_k), j = -3,...,3, k = -1,...,N+1. This implies that the following B-spline coefficients must be identical, to ensure the periodicity of s and its derivatives up to the second order in the ϕ-direction

$$c_{j,k} = c_{M+j,k}, \quad j = -1,0,1, \quad k = -1,\ldots,N+1. \tag{3.7}$$

The number of unknown B-spline coefficients is therefore reduced to M(N+3).

In order to satisfy condition (2.2) for the upper pole, we impose in the knots

$$s(\phi_k,\pi/2) = \sum_{i=k-1}^{k+1} \sum_{j=N-1}^{N+1} c_{ij}B_i(\phi_k)B_j(\pi/2) = r_1, \tag{3.8}$$

$$k = 0,1,\ldots,M-1,$$

together with the periodicity condition (3.7).

Introducing new variables v_i leads to

$$\sum_{i=k-1}^{k+1} v_i B_i(\phi_k) = r_1, \quad v_i = \sum_{j=N-1}^{N+1} c_{ij}B_j(\pi/2). \tag{3.9}$$

$$(i = 0,1,\ldots,M-1)$$

The parameters v_i can be considered as the B-spline coefficients in a 1-dimensional problem to find a periodical spline function equal to the same value r_1 in all knots. These M equations (3.9) have precisely one solution [1] (namely the constant function r_1), since the B-splines are linearly independent. Using the normalization (3.2), we see immediately

$$v_i = r_1, \quad i = 0,1,\ldots,M-1. \tag{3.10}$$

THEOREM 3.1. *The function* $s(\phi,\theta)$ *will satisfy* (2.2) *if, in addition to the periodicity condition* (3.7),

$$\sum_{j=N-1}^{N+1} c_{ij}B_j(\pi/2) = r_1, \quad \sum_{j=-1}^{1} c_{ij}B_j(-\pi/2) = r_2 \tag{3.11}$$

$$i = 0,1,\ldots,M-1.$$

At the end of this section the equations (3.11) will be used to eliminate certain B-spline coefficients, expressed in the remaining parameters.

The conditions (2.9) and (2.20) are necessary to produce a C^2-smooth surface. However it will be impossible to satisfy these conditions exactly when using spline functions.

The derivatives s_θ, $s_{\theta\theta}$ are *piecewise polynomial* and will not be equal to a trigonometric function of the type

$$(3.12) \qquad a.\cos\phi + b.\sin\phi \text{ or } c.\cos^2\phi + d.\sin\phi\cos\phi + e.\sin^2\phi$$

for *all* $\phi \in [0,2\pi]$.

Therefore we impose the conditions (2.9) and (2.20) only in the knots $(\phi_k,\pm\pi/2)$ in *least-squares* norm. These equations for the B-spline coefficients will be treated simultaneously with the data equations in our algorithm. Formulas (2.9), (2.20), (3.11) introduce 12 new parameters for the solution s, namely

$$(3.13) \qquad r_1,\ r_2,\ s_\theta(0,\pm\pi/2),\ s_\theta(\pi/2,\pm\pi/2),\ s_{\theta\theta}(0,\pm\pi/2),$$

$$s_{\theta\theta}(\pi/4,\pm\pi/2) \quad \text{and} \quad s_{\theta\theta}(\pi/2,\pm\pi/2).$$

There is a weaker smoothness condition, which can be satisfied exactly in the poles. It is possible to guarantee that the approximating surface is *Gateaux differentiable* in the poles

$$(3.14) \qquad \vec{x}_\theta(\phi,\pm\pi/2) = -\vec{x}_\theta(\pi+\phi,\pm\pi/2), \quad \phi \in [0,\pi].$$

If we consider (2.3) this simply means

$$(3.15) \qquad s_\theta(\phi,\pm\pi/2) = -s_\theta(\pi+\phi,\pm\pi/2), \quad \phi \in [0,\pi].$$

Expanding the equations (3.15) for the upper pole in B-spline coefficients, gives

$$(3.16) \qquad \sum_{i=-1}^{M+1}\sum_{j=N-1}^{N+1} c_{ij}B_i(\phi)B_j'(\pi/2) = -\sum_{i=-1}^{M+1}\sum_{j=N-1}^{N+1} c_{ij}B_i(\pi+\phi)B_j'(\pi/2)$$

$$\phi \in [0,\pi].$$

It is convenient to choose the grid in a special way, namely the partition in the θ-direction symmetrical with respect to the upper pole $\theta_N = \pi/2$

(3.17) $$\theta_{N+j} - \theta_{N+j-1} = \theta_{N-j+1} - \theta_{N-j} = h_j, \quad j = 1,2,3$$

and similarly for the lower pole $\theta_0 = -\pi/2$

(3.18) $$\theta_j - \theta_{j-1} = \theta_{-j+1} - \theta_{-j} = h'_j, \quad j = 1,2,3.$$

In the ϕ-direction we use the same partition for both intervals $[0,\pi]$ and $[\pi,2\pi]$

(3.19) $$\phi_{k+M/2} = \phi_k + \pi, \quad k = 1,\ldots,M/2-1$$

$$\phi_{M/2} = \pi, \quad M \textit{ even}.$$

Due to the symmetry of the partition (3.17) the function $B_N(\theta)$ satisfies

(3.20) $$B'_N(\pi/2) = 0, \quad B'_{N-1}(\pi/2) = -B'_{N+1}(\pi/2).$$

Formula (3.16) can now be simplified, dividing by $B'_{N+1}(\pi/2)$

(3.21) $$\sum_{i=-1}^{M+1} (c_{i,N+1}-c_{i,N-1})B_i(\phi) = -\sum_{i=-1}^{M+1} (c_{i,N+1}-c_{i,N-1})B_i(\pi+\phi).$$

In (3.21) only a few terms are different from zero, since the B-splines have a limited support. If we suppose $\phi \in [\phi_k,\phi_{k+1}]$ (therefore $\pi + \phi \in [\phi_{k+M/2}, \phi_{k+M/2+1}]$) and if we use $B_i(\phi) = B_{i+M/2}(\pi+\phi)$, we find

(3.22) $$\sum_{i=k-1}^{k+2} (c_{i,N+1}-c_{i,N-1}+c_{i+M/2,N+1}-c_{i+M/2,N-1})B_i(\phi) = 0.$$

This equation should hold for all ϕ, implying

THEOREM 3.2. *The surface represented by* (3.1) *is Gateaux differentiable in the poles if, in addition to* (3.7) *and* (3.11), *the coefficients* c_{ij} *satisfy*

(3.23) $$c_{i,N+1}-c_{i,N-1} = -(c_{i+M/2,N+1}-c_{i+M/2,N-1})$$

$$c_{i,1}-c_{i,-1} = -(c_{i+M/2,1} - c_{i+M/2,-1})$$

$$i = 0,1,\ldots,M/2-1.$$

The special partitions in the ϕ- and θ-direction are necessary for a correct

matching of the grid, when closing the surface at the poles.

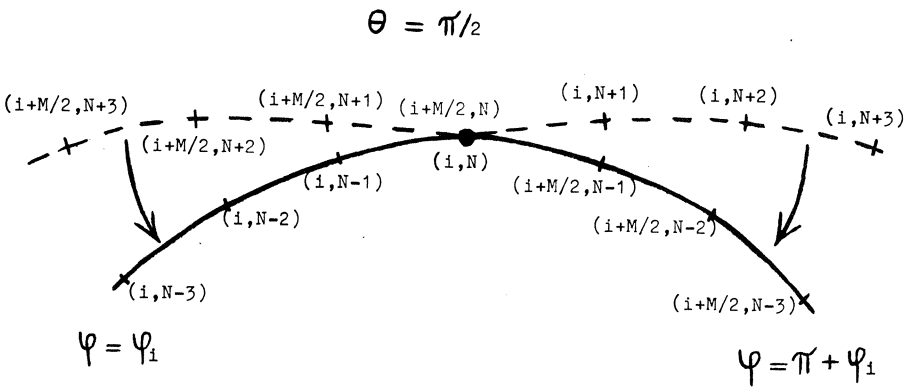


Fig. 3.2. Closing the surface at the upper pole.

Condition (3.19) implies that ϕ_i and $\pi + \phi_i$ are both vertical grid lines. Together they represent a closed curve on the surface T passing smoothly through both poles. The symmetry of the θ-partition provides the matching of corresponding knots: (i,N-j) versus (i+M/2,N+j) and (i,N+j) versus (i+M/2,N-j), j = 1,2,3 in the overlapping partitions, as indicated in fig. 3.2 for the upper pole.

The B-spline function $s(\phi,\theta)$ will be twice Gateaux differentiable in the poles, if

$$(3.24) \qquad s_{\theta\theta}(\phi,\pm\pi/2) = s_{\theta\theta}(\pi+\phi,\pm\pi/2), \quad \phi \in [0,\pi].$$

In the upper pole $s_{\theta\theta}$ is given by

$$(3.25) \qquad s_{\theta\theta}(\phi,\pi/2) = \sum_{i=-1}^{M+1} \sum_{j=N-1}^{N+1} c_{ij} B_i(\phi) B_j''(\pi/2),$$

with $$B_N''(\pi/2) = -2B_{N-1}''(\pi/2) = -2B_{N+1}''(\pi/2).$$

The same arguments as in the proof of theorem 3.2 lead us to

THEOREM 3.3. *The surface represented by* (3.1) *is twice Gateaux differentiable in the poles if, in addition to* (3.7), (3.11) *and* (3.23),

$$(3.26)\qquad c_{i,N+1}-2c_{i,N}+c_{i,N-1} = c_{i+M/2,N+1}-2c_{i+M/2,N}+c_{i+M/2,N-1}$$

$$c_{i,1}-2c_{i,0}+c_{i,-1} = c_{i+M/2,1}-2c_{i+M/2,0}+c_{i+M/2,-1}$$

$$(i=0,1,\ldots,M/2-1).$$

We have solved the B-spline coefficients c_{ij}, $j = -1,0,N,N+1$ from equations (3.11), (3.23) and (3.26), expressed in the pole-distances r_1, r_2 and the coefficients $c_{i,1}$ and $c_{i,N-1}$.

$$(3.27)\qquad c_{i,-1} = c_{i+M/2,1}$$

$$c_{i,0} = (r_2(2h_1'+h_2') - h_1'(c_{i,1}+c_{i+M/2,1})/2)/(h_1'+h_2')$$

$$c_{i,N} = (r_1(2h_1+h_2) - h_1(c_{i,N-1}+c_{i+M/2,N-1})/2)/(h_1+h_2)$$

$$c_{i,N+1} = c_{i+M/2,N-1}$$

(the index i+M/2 should be read as (i+M/2) mod M).
In the upper pole one uses for the derivation of (3.27)

$$(3.28)\qquad B_N(\pi/2) = \frac{h_1+h_2}{2h_1+h_2},\quad B_{N-1}(\pi/2) = B_{N+1}(\pi/2) = \frac{h_1}{2(2h_1+h_2)}.$$

(Similarly for the lower pole).

There remain M(N-1)+12 parameters to fit the B-spline surface to the given data values, namely c_{ij}, (i=0,1,...,M-1), (j=1,...,N-1) and the 12 parameters from (3.13). All constraints for the coefficients c_{ij} have been solved explicitly, with the exception of the trigonometric formulas (2.9) and (2.20). It is sufficient to impose these two conditions only in the knots $(\phi_i,\pm\pi/2)$, i = 0,1,...,M/2-1 of the first interval $[0,\pi>$, since (3.15) and (3.24) guarantee the same behaviour in the second interval $[\pi,2\pi>$. These 2M constraints in the poles can be considered as separate interpolation problems in the knots of a 1-dimensional partition and are therefore linearly independent [7].

The least-squares problem, consisting of the data equations and the 2M constraints, is in the full rank case solved by a QU factorization, obtained by *Householder transformations*.

A special advantage of the reconstruction method is the considerable *data reduction*. The surface $s(\phi,\theta)$ is determined completely by knots and B-spline coefficients. Only these parameters of the reconstructed organ will be stored in a data base, instead of all the original data points.

4. APPLICATIONS

In this section we give a number of applications of the B-spline surface reconstruction, which are interesting for medical research.

VOLUME V OF THE ORGAN

$$(4.1) \qquad V = \iiint d\vec{x} = \frac{1}{3} \int_{\phi=0}^{2\pi} \int_{\theta=-\pi/2}^{\pi/2} s^3(\phi,\theta)\cos\theta d\theta d\phi.$$

In formula (4.1) we used the determinant of the Jacobian matrix for the spherical coordinate system $|J| = s^2(\phi,\theta)\cos\theta$.

CENTER OF GRAVITY $\vec{g}$

Assuming homogeneous bodies, we find for the center of gravity

$$(4.2) \qquad \vec{g} = \frac{1}{V} \iiint \vec{x}\, d\vec{x} = \frac{1}{4V} \int_{\phi=0}^{2\pi} \int_{\theta=-\pi/2}^{\pi/2} s^4(\phi,\theta)\vec{u}(\phi,\theta)d\theta d\phi,$$

with $\vec{u} = \cos\theta\ (\cos\phi\cos\theta,\ \sin\phi\cos\theta,\ \sin\theta)^T$.

Similar formulas could be given for the moments of inertia of a 3-dimensional object.

OUTWARD DIRECTED NORMAL VECTOR $\vec{n}_T$

The normal vector $\vec{n}_T$ in an arbitrary point of the surface T is used to produce a shaded picture of the organ. The angle between the viewpoint and the normal $\vec{n}_T$ determines the light intensity of a point on the terminal screen.
The outward directed normal vector is defined by

$$(4.3) \qquad \vec{n}_T = \vec{v} \times \vec{w}, \text{ with a) } \vec{v} = \vec{x}_\phi,\ \vec{w} = \vec{x}_\theta,\ \theta \neq \pm\pi/2 \text{ (see formula (2.3))}.$$

b) $\vec{v} = \vec{x}_\theta(0,\pi/2)$, $\vec{w} = \vec{x}_\theta(\pi/2,\pi/2)$, for $\theta = \pi/2$.

c) $\vec{v} = \vec{x}_\theta(\pi/2,-\pi/2)$, $\vec{w} = \vec{x}_\theta(0,-\pi/2)$, for $\theta = -\pi/2$.

In the poles $\vec{x}_\phi = \vec{0}$ and therefore we use the alternative representation (4.3)b/c for the normal vector.

A direct evaluation of $\vec{v} \times \vec{w}$ may cause loss of accuracy as a result of the subtractions in the formula for the cross product. To avoid this problem, we apply a *modified Gram-Schmidt* procedure to $\{\vec{v},\vec{w}\}$ in order to determine $\vec{v} \times \vec{w}$. The correct orientation of $\vec{v},\vec{w}$ and $\vec{v} \times \vec{w}$ is found by inspecting the sign of $\det(\vec{v},\vec{w},\vec{v} \times \vec{w})$, using a QU factorization. Numerical tests show that this is a stable procedure to calculate the outward directed normal vector.

SURFACE S OF THE ORGAN

Integrating the length of the vector $\vec{x}_\phi \times \vec{x}_\theta$ over the rectangular domain R, gives the surface S

$$(4.4) \qquad S = \iint dS = \int_{\phi=0}^{2\pi} \int_{\theta=-\pi/2}^{\pi/2} \| \vec{x}_\phi \times \vec{x}_\theta \|_2 \, d\theta d\phi .$$

We calculate the 2-dimensional integrals (4.1), (4.2) and (4.4) by dividing the domain R in subsquares R_{ij} and applying a 2-dimensional 4-point *Gaussian quadrature* rule to each subsquare. This quadrature formula uses 16 function-evaluations for each element R_{ij}. The final approximation for the integral is obtained by an iterative rule: the square elements R_{ij} are divided into 4 smaller squares, until the error, due to the numerical integration, is smaller than a specified error bound.

We note that the evaluation of the surface S is approximately three times as expensive as the calculation of the volume V, because in every point not only s but also s_ϕ and s_θ are necessary to determine the vector $\vec{x}_\phi \times \vec{x}_\theta$.

Calculating the *lowest and highest points* of the surface reconstruction in the direction $\vec{f}$, orthogonal to the photo planes, is a useful tool to determine the quality of the approximation in practical situations. We obtain these points as solutions of the following optimization problems

$$(4.5) \qquad \underset{(\phi,\theta)\in R}{\text{minimize}} \; D(\phi,\theta) \quad \text{resp.} \quad \underset{(\phi,\theta)\in R}{\text{maximize}} \; D(\phi,\theta),$$

with

$$D(\phi,\theta) = (\vec{f},\vec{x}(\phi,\theta)) = f_1 x(\phi,\theta) + f_2 y(\phi,\theta) + f_3 z(\phi,\theta).$$

The *intersection* of a reconstructed surface

(4.6) $$\vec{x}(\phi,\theta) = s(\phi,\theta).(\cos\phi\cos\theta,\ \sin\phi\cos\theta,\ \sin\theta)^T$$

$$s(\phi,\theta) = \sum_{i,j} c_{ij}B_i(\phi)B_j(\theta)$$

with an arbitrary plane corresponds, for a fixed value $\phi = \phi_v \in [0,2\pi]$, with the problem of calculating all zeros of a 1-dimensional function in θ

(4.7) $$h(\theta) = \alpha_1 + s(\phi_v,\theta).(\alpha_2\sin\theta + \alpha_3(\phi_v)\cos\theta), \quad \theta \in [-\pi/2,\pi/2].$$

In fig 4.1 an original data contour and the intersection with the reconstructed tumor are given in the same photo plane.

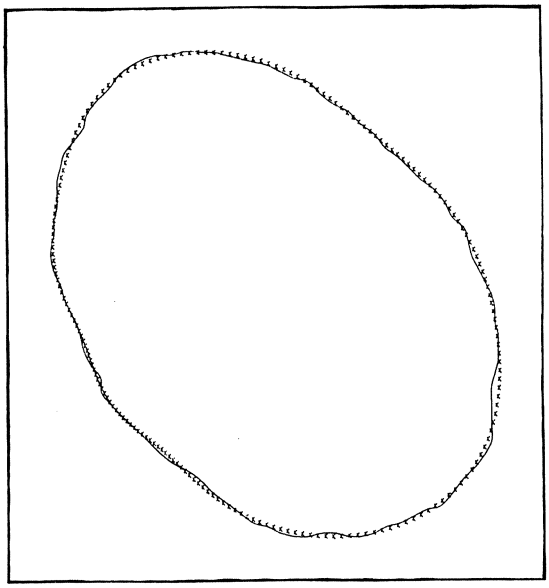


Fig. 4.1. Data contour and intersection (dotted line) for a tumor.

5. NUMERICAL RESULTS

We have tested the surface approximation by B-splines, described in section 3, on a number of 3-dimensional objects:

- constructed surfaces, such as a sphere, with random data errors and surfaces of revolution.
- biological material, namely data contours of a tumor and an eye.

The constructed surfaces were used to test the computer program and to

determine the accuracy of the reconstruction, since analytical expressions for volume, surface and outward directed normal vector are available. For the sphere the approximation gives the exact solution, with all B-spline coefficients equal to the radius.

The PASCAL computer program calculates the optimum values of the B-spline coefficients and the parameters in the poles, together with some of the special applications from section 4 for a general starlike surface. The algorithm reads the necessary information, concerning the coordinate transformation, the B-spline grid and the data points, from different input files. In this way the influence of the B-spline partition and the chosen data set on the approximation quality could separately be investigated.

Provided the data points are uniformly distributed over the rectangle R and the surface T is not too ill-conditioned, all numerical tests resulted in a C^2- smooth accurate surface approximation. The method is quite flexible, since the mesh (M,N), the B-spline knots (ϕ_i,θ_j) and the individual weights of the data points can be chosen to suit the local behaviour of the data in an optimum way.

The quality of the surface approximation is determined mainly by the distribution of the data points over the (ϕ,θ)-domain. The positions of the different data contours in the photo direction $\vec{f}$ determines the *condition* of the approximation problem. These photos should be chosen in such a way that the data points are distributed as uniformly as possible over the rectangle R. Especially large gaps [8] in the data should be avoided and enough data points should be in the direct surroundings of both poles. This last condition will ensure the correct calculation of the values of the function s and the first and second derivatives in the poles. If the data are not uniformly distributed over the domain R, the B-spline algorithm may produce an unacceptable surface.

In reconstruction problems one should choose a smaller distance between successive photos in the neighbourhood of the poles. For convex objects we found good surface approximations when the photos were placed at the *zeros of a Chebyshev polynomial* [3], defined on the interval in the photo direction between the lowest and highest points of the surface.

An important question is the *order of accuracy* of the surface reconstruction, in other words the rate of convergence of the spline approximation s to the exact solution r as the maximum mesh size approaches zero.

Numerical experiments indicate, in the case of well-distributed data,

the following error behaviour

(5.1) a) *normal vector of unit length:* $\| \vec{d}_{n_T} \|_\infty = O(h^2)$,

$$\vec{d}_{n_T} = \| \vec{n}_{T,s} - \vec{n}_{T,r} \|_2, \quad \| \vec{n}_{T,s} \|_2 = \| \vec{n}_{T,r} \|_2 = 1$$

b) *reconstruction:* $\| s - r \|_\infty = O(h^3)$

c) *surface:* $|S_s - S_r| = O(h^4)$

d) *volume:* $|V_s - V_r| = O(h^5)$,

with the maximum mesh size $h = \max_{i,j} \{\phi_{i+1} - \phi_i, \theta_{j+1} - \theta_j\} \downarrow 0$.
In formula (5.1) the indices s and r indicate the B-spline approximation $s(\phi,\theta)$ and the exact solution $r(\phi,\theta)$ respectively.

From the literature [4], [5] we would expect for bicubic interpolation and least-squares approximation of a function $r \in C^4(R)$ a fourth order rate of convergence $O(h^4)$. The experiments show that our error bound for $s(\phi,\theta)$ behaves like $O(h^3)$ and therefore that we have lost one order in the accuracy of the surface reconstruction. This is most likely caused by the special equality constraints in the poles. Some B-spline coefficients are eliminated to ensure periodicity and to close the surface. Other parameters appear in special equations to obtain a differentiable C^2-smooth surface in the poles. There remain $M(N-1)+12$ parameters to fit the data values, which is strictly less than the dimension $(M+3)(N+3)$ of a general bicubic spline space. The convergence rates for the volume V, the surface S and the outward directed normal vector $\vec{n}_T$ are consistent with (5.1) b, since we expect for a least-squares problem the order to increase by 2 for integration and to reduce by 1 for differentiation.

Provided enough data points are available near the poles, the error in $\frac{\partial^j r}{\partial \theta^j} (\phi, \pm\pi/2)$ will be of the order $O(h^{3-j})$.

For a theoretically constructed surface and different mesh sizes the behaviour of the errors in the normal vector of unit length, the reconstruction, the surface and the volume is given in fig. 5.1.

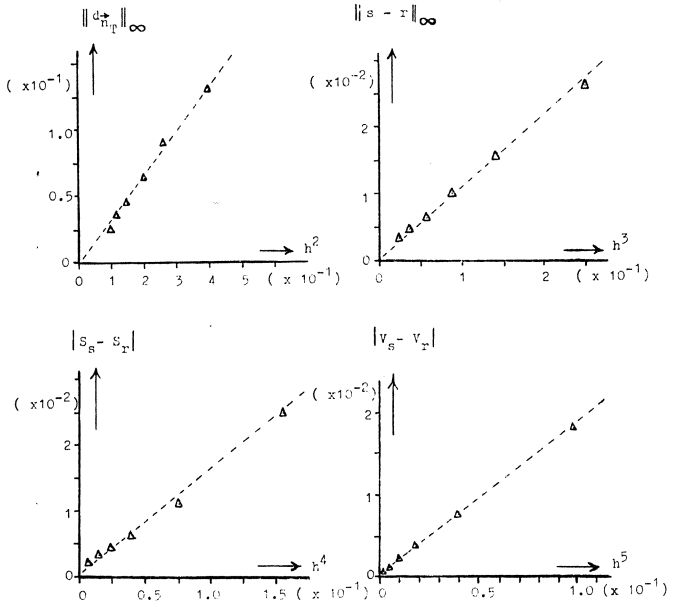


Fig. 5.1. Errors in the surface reconstruction: equally spaced knots M = 20, N = 5,...,10.

We consider now the approximation quality of s_θ and $s_{\theta\theta}$ along the horizontal boundaries. As indicated in section 3 the piecewise cubic splines in ϕ approximate the trigonometric functions $r_\theta(\phi,\pm\pi/2)$ and $r_{\theta\theta}(\phi,\pm\pi/2)$ in the knots $(\phi_i,\pm\pi/2)$ in least-squares norm. Due to the bounded higher derivatives of (2.9), (2.20) we can always find a good 1-dimensional spline approximation. In the numerical tests considered the errors were small, mainly depending on M and the chosen partition in the ϕ-direction.

In fig. 5.2 the piecewise cubic spline functions $s_\theta(\phi,\pi/2)$ and $s_{\theta\theta}(\phi,\pi/2)$, $\phi \in [0,2\pi]$ are given along the upper pole $\theta = \pi/2$ for a constructed surface. The trigonometric function values, imposed at the knots $(\phi_i,\pi/2)$, i = 0,1,...,M/2-1, are indicated by an asterisk. The maximum error over the interval $[0,2\pi]$ is given at the top of both figures.

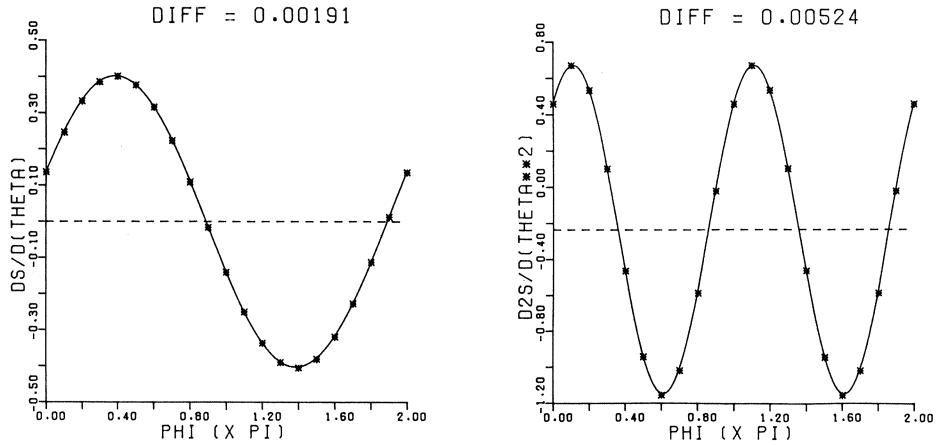


Fig. 5.2. Behaviour of the derivatives s_θ and $s_{\theta\theta}$ in the upper pole: equally spaced knots, M = 20.

The *stability* of the reconstruction algorithm has been investigated by considering the change in the approximation s for small perturbations in the data (for example a photo removed from the data set). We found only minor changes in the solution, unless the new data set was not properly distributed over the domain R (especially for a change in the data close to the poles). However, even in the case of properly distributed photos, for an ill-conditioned 3-dimensional object (such as a surface which is nearly not starlike) the reconstruction will lead to numerical problems.

Finally we note that a choice of the pole direction $\vec{n}$ with the line $\vec{m} + \lambda\vec{n}$ passing through the interior of all photo contours and the mesh (M,N) not too fine (with respect to the data distribution) usually gives the best results for the surface reconstruction.

ACKNOWLEDGEMENT

The research for this contribution was performed at the Mathematics Department of the University of Amsterdam in cooperation with dr. P. Pfluger and R. Houweling, who wrote a large part of the computer program. The author would like to thank dr. N. Huismans and dr.ir.J. Smith of the Laboratory of Medical Physics (3D Reconstruction Group, University of Amsterdam), who kindly supplied the data sets.

REFERENCES

[1] AHLBERG, J.H., E.N. NILSON & J.L. WALSH, *The theory of splines and their applications*, Academic Press, New York, 1967.

[2] COX, M.G., *Practical spline approximation*, Lecture Notes in Mathematics 965, pp. 79-112, Springer Verlag, New York, 1981.

[3] de BOOR, C., *A practical guide to splines*, Springer Verlag, New York, 1978.

[4] HALL, C.A., *On error bounds for spline interpolation*, J. Appr. Theory 1, 209-218, 1968.

[5] HALL, C.A. & W.W. MEYER, *Optimal error bounds for cubic spline interpolation*, J. Appr. Theory 16, 105-122, 1976.

[6] HAYES, J.G. & J. HALLIDAY, *The least-squares fitting of cubic spline surfaces to general data sets*, J. Inst. Maths. Applics. 14, 89-103, 1974.

[7] SCHUMAKER, L.L., *Spline functions: basis theory*, John Wiley & Sons, New York, 1981.

[8] SCHMIDT, R.M., *Fitting scattered surface data with large gaps*, Surfaces in CAGD, ed. Barnhill, R.E. & W. Boehm, North-Holland, Amsterdam, 1983.

WORKING WITH VECTOR COMPUTERS: AN INTRODUCTION

J.P. HOLLENBERG

Since the velocity of light has become an important factor in computer design, improvements in the basic hardware have been insufficient - by themselves - to produce the computational power that is required to solve some challenging problems. In order to speed up computation nevertheless, the designers of present day supercomputers use parallellism. We will focus on one form of parallellism, called pipelining. As will be shown, in that concept the results are manufactured like on an assembly-line, which is effective when working with vectors of numbers.

Two brands of highly pipelined vector machines, which are commercially available today (Cyber 205 and CRAY-1) will be briefly discussed. Furthermore the inpact of this type of architecture on the choice of algorithms and their coding will be illustrated, mainly by examples.

1. INTRODUCTION

Very unsophisticated, the traits of a computer can be explained by the equation

technology + demands = design.

At the technology term of this 'sum', we see a considerable decrease in the price of computer components. Also there is a slowdown in the growth of computing power available from a single processor. This growth has a severe constraint in the velocity of light (approximately one feet per nanosecond). One can try to shorten the connections, of course, but components generate heat, which limits how closely they can be packed together without burning up.

The demand term was recently investigated, when the Japanese supercomputer project, fuelled by talent and money [4], seemed to challenge the clear lead of the U.S.A. in that field. The work of an 'emergency' US-government

panel resulted in, among other things, an up-to-date inventory of the needs of the scientific community for computing power [23]. As areas of major impact are considered:

- Aircraft design. The wings, tails, *etcetera* are designed individually now and the test pilot sees how well they work together. An aircraft that is designed as a whole will have superior performance at less costs. There is a similar situation with submarine design.
- Geophysical exploration. Only crude simplified models of oil fields can be handled. More accurate simulations will improve recovery.
- Atmospheric models. More detailed models are needed to improve short range predictions and allow studies of long term patterns.
- Nuclear weapons. Drastic simplifications are made in the modelling and simulation of these, and yet enormous amounts of computer time are consumed.
- Electronic devices. Current designs are two-dimensional, but better and cheaper chips can be obtained by three-dimensional concepts. The computational requirements to design the layout and the components are formidable. Today already, the supercomputer is a vital tool in the development of supercomputers. To give an example, two Cyber 205 computers are working around the clock, assisting with the design of successor products.
- Disease control. Even simple viruses are complex molecules. To test their reaction to various drugs, with the aim to control them, is a major computational task.

In general we see that experimenting is sometimes expensive, dangerous or impossible, so simulation is used. When traditional analytic means are not helping to solve the mathematical models, one has to use numerical modelling. The ever-increasing demands are then generated by realistic (3D!) models.

The design result is - sometimes - in terms of computing power an order of magnitude below the presently defined needs. Anyway it is clear that SISD (single instruction, single data) is not good enough anymore; so parallellism is used. Two types of parallellism can be distinguished:

- Replication. Examples of processor arrays are DAP (ICL), ILLIAC IV (Burrough), MPP (Goodyear Aerospace) and HEP (Denelcor); the last one is MIMD (multiple instruction, multiple data) and the others SIMD (single instruction, multiple data).
- Pipelining. If pipelining is applied the machine is called an array processor; these are all SIMD.

Of course individual designs (CRAY X-MP) can combine these features. Pipelining can be used in:

- Peripheral devices. Examples of array processors are Floating Point Systems products, ST-100 (Star technologies), AP 500 (Analogic) and many more.
- Mainframes. Examples of vector computers are CRAY-1 (Cray Research, Inc.), Cyber 205 (Control Data Corp.) and SX (NEC Corp.).

Machines of National Advanced Systems can be provided with an Integrated Array Processor. Other Japanese products, HITAC S-820 (Hitachi, Ltd.) and FACOM VP (Fujitsu, Ltd.), can be used as a peripheral and in stand-alone mode. A far more detailed overview can be found in [12], in fact most of the subjects to follow are treated in this book; as short treatments on this subject about the use of vector computers for scientific purposes, see [3].

2. PIPELINES

When different (sub)operations can overlap in time, the assembly-line principle, like in - for example - a car factory, can be applied. This method, called pipelining, is effective when applied to a sequence of identical operations, as occurs when working with vectors. For an example we look at the floating-point addition. In most Control Data machines the operands must undergo

sign control, exponent compare and alignment shift

before the actual addition, and on the result

normalize count, normalize shift and end case detection

are applied. Having the suboperations overlapping in time, we can perform a sequence of identical operations in parallel. Again the example is addition (of two vectors of real numbers). Every partial operation is synchronized to last one time unit, called cycle time. We see that a result is generated every clock cycle, although one addition takes five clock cycles.

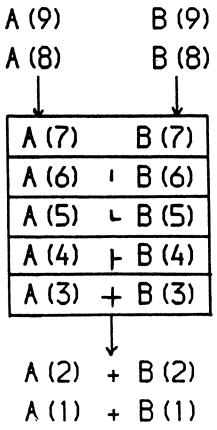


Fig. 1

For additional literature on pipes see [30] or [11]. From the family of vector computers we will now treat - shortly - two current models.

3. CRAY-1

In 1972 Seymour Cray left CDC and founded Cray Research, with the aim of producing the fastest computer in the world. In four years the CRAY-1 was designed and the first model delivered. Now about sixty have been sold; a commercial success indeed. The physical organization of the CRAY-1 provoked the nickname 'The world's most expensive love-seat'. Its architecture is reviewed in [33].

An additional - and speed accelerating - feature of CRAY vector hardware is 'chaining'. In this process the result register of a vector operation becomes the operand register of the succeeding instruction [16]. It is easy to understand, that this is only possible when the two successive instructions do not have common operands or operators. In fig. 2 we see an example of chaining: A vector (V0) is read from memory, another vector (V1) is added to it and the result (V2) is multiplied by a third vector (V3).

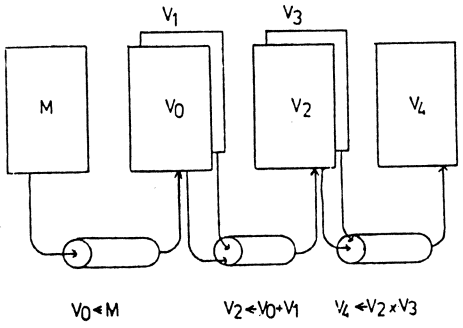


Fig. 2

Naturally, longer (series of) pipes give more Mflops (an abbrevation for millions of floating-point operations per second), also because the vectors then do not have to wait frequently for each other at the (only) access between memory and registers. The CRAY-1 has regularly achieved - while working with chains - rates of 130 Mflops.

4. CYBER 205

The STAR 100 was developed by Control Data about 1964 as a pipelined computer, with a set of hardware primitives based on Iverson's APL, that would be able to do 100 Mflops. However, the machines that were actually built (only four?) were only able to outperform general-purpose computers in carefully doctored circumstances. Completely re-engineered the machine was introduced as the Cyber 203 [19] and has meanwhile been upgraded to the Cyber 205 [29]. About twenty are installed.

In the Cyber 205, the equivalent of a chain is called a 'linked triad'. In a linked triad a multiplication is combined with an addition (or a subtraction). There is maximum of two input streams, so one of the operands has to be a scalar. (Note that a chain does not have that restriction.) The machine is very fast when operating on linked triads; in the next section we will see how fast.

5. COMPARISON

The CRAY-1 performs register-to-register vector operations, as opposed to the Cyber 205, which does these operations from and to memory. The

Cyber 205 has a cycle time of 20 nanoseconds, the CRAY-1 one of 12.5. The CRAY-1 has specialized pipes and the Cyber 205 has (up to four) general-pupose pipes. From the user/programmer's point of view, these technical details have the following consequences:

- Strides. On the CRAY-1 the elements of a vector can be stored with a constant increment, which may also be negative. For good results on the Cyber 205, the elements must be stored contiguously. In some applications (like the use of rows of 2-dimensional arrays in FORTRAN) this is considered a downright nuisance [35]. To compensate for this, there is an efficient gather/scatter facility among the many fancy hardware instructions of the Cyber 205; a call thereof is sometimes generated by the compiler.
- Speed. As said, the Cyber 205 performs very well when operating on linked triads. In that case 400 Mflops can be achieved with two pipes (and a wordlength of 32 bits), and even 800 Mflops with four pipes. With a chained operation of three vectors the CRAY-1 can do 160 Mflops. Of course these are theoretical figures, derived only from the cycle time and only useful in commercials, because in 'normal' circumstances some 50 Mflops are a very good result. For an example of a realistic comparison see [37].
- FORTRAN. (This language may be not ideal from the standpoint of computer scientists, but the fact that important software is written in it has resulted in a widespread use throughout the world and *vice versa*; also with vector computers FORTRAN is the *lingua franca*.) A disadvantage of the CRAY-1 is that sometimes the operand vectors have to be loaded and reloaded from and into registers to and from memory through one single unit. Therefore it is virtually impossible to keep the floating-point operators busy all the time with compiler-generated code. So resorting to Cray Assembler Language (CAL) can sometimes double the performance. On the Cyber 205 the arithmetic is performed memory-to-memory directly and little is gained by using machine code, also because of the many hardware instructions available. The use of these instructions in a FORTRAN program naturally destroys its portability. On the CRAY-1 even an optimized program can be of good use on other computers.
- Vectors. It is clear that the start-up time makes computations inefficient for short vectors. On different machines, however, the start-up time and 'short' are different.

Many questions in comparison (especially the ones connected with vectors) can be answered by observing the time required to perform n elemental

operations and fitting this to the generic form [13]:

$$t_n = r_\infty^{-1}(n+n_{\frac{1}{2}}),$$

where

r_∞ is the asymptotic performance in Mflops and

$n_{\frac{1}{2}}$ is the half-performance length, that is the length necessary to achieve half the maximum (theoretical) performance. The parameter is a measure of the parallelism in the hardware; it is zero for a serial computer.

This two-parameter characterization of a computer enables us to plot a two-dimensional spectrum of computers. For the CRAY-1 the half-performance length is around 10. In contrast, for the Cyber 205 this value ranges from 100 to 600, depending on the operations and the number of pipes. So we see that a vector length of 64 is good for the CRAY-1 and bad for the Cyber 205 (and 65 is bad for both, also for the CRAY-1 because of the size of its vector registers).

6. VECTORIZATION

The process of tuning a program for an array processor is called vectorization. It can be considered a problem or an opportunity [18], because, in vector computers, the ratio of best to worst performance has greatly increased. So, on one hand, it can be observed that a vector computer is like a bicycle without learning wheels; it is important where to go, but also how to get there. But, on the other hand, one might say that a vector computer confuses the original scientific problems by difficulties due to the - temporary - peculiarities of the machine and the compiler.

It should be noted that a new type of computer provides an opportunity to design new (better) algorithms; often this means that data structures are used which fit the machine better. We will see that this is also true for vector computers, but first we will consider vectorization as an opportunity to enhance the performance of existing software. So we have to (re)write programs, with a certain computer/compiler combination in mind, to take advantage of the hardware. Many examples and the resulting speed-up can be found in [28] and [24]. We will try to shed light on this matter by giving

three examples, pieces of FORTRAN programs. They will be changed, in order to gain speed (but, of course, with the same results).

For virtually all compilers, even on a serial computer, it is worthwhile to restructure the IF-loop

```
      I = 1
  100 R(I) = A(I)*B(I)+1.0
      I = I+1
      IF(I+6.LE.N)GO TO 100
```

into the DO-loop

```
      NM6 = N-6
      DO 100 I = 1, NM6
  100 R(I) = A(I)*B(I)+1.0
```

and it will run - for N = 1006 - ten (on the CRAY-1) to thirty (on the Cyber 205) times faster. The better performance is caused by the fact that the compiler will recognize the second loop as suitable for a pipeline (this property is called vectorizability).

Invariably on a serial computer, matrix multiplication (A=B*C) is programmed by the inner-product method:

```
      DO 10 I = 1,N
      DO 10 J = 1,N
      A(I,J) = 0.0
      DO 10 K = 1,N
   10 A(I,J) = A(I,J)+B(I,K)*C(K,J)
```

however, if the middle-product method is used instead

```
      DO 10 J = 1,N
      DO 11 I = 1,N
   11 A(I,J) = 0.0
      DO 10 K = 1,N
      DO 10 I = 1,N
   10 A(I,J) = A(I,J)+B(I,K)*C(K,J)
```

the CRAY-1 will be (for n=72) two times faster and the Cyber 205 three times. The reason is that the operands are fetched quicker. Note that in the line with label 10, the Cyber 205 can perform a linked triad (the element of C is a scalar in this inner loop). In general we see that matrix multiplication can be programmed in six different ways simply by interchanging the nestings of the loops. The execution speed of the six variants differ for a vector machine much more than on a scalar machine.

The unrolling [6] of the outer loop of

```
      DO 10 J = 1,N2
      DO 10 I = 1,N1
   10 Y(I) = Y(I)+X(J)*M(I,J)
```

into

```
      DO 10 I = 1,N1
      Y(I) = ((((Y(I))+X(J-3)*M(I,J-3))
     1                +X(J-2)*M(I,J-2))
     2                +X(J-1)*M(I,J-1))
     3                +X(J  )*M(I,J  )

   10 CONTINUE
```

enchances the performance of the CRAY-1 from fourty to eighty Mflops. The brackets are placed as they are in order to force the compiler to arrange an optimal transport of data between memory and registers. The speed-up is also due to the better ratio of data handling/actual calculations.

Not every loop is vectorizable. An easy example is a loop with recursion:

```
      DO 10 I = 2,N
   10 X(I) = A/X(I-1)
```

Clearly, this can not be vectorized, because an element of X would be needed as input to a pipe before it has been produced by that pipe. Also branching (for example by an IF-statement) prevents a loop from being vectorized. This is because all operations are- possibly -not identical and therefore the assembly-line principle can no longer be applied. Problems are also countered when we use indirect addressing and I/O. When one of the above mentioned conditions arises, the programmer (or the compiler) can sometimes use very fast build-in hardware functions.

Vectorization has not always to be done by humans. An example of a

program for automatic vectorization is VAST. The *raison d'etre* for automatic vectorization programs is the fact that compilers are - still - not perfect. Such programs are useful when thousands of lines of code have to be vectorized. However, the bulk of the work in a program is mostly done in a limited number of lines of code. On such a small scale, the natural lack of creativeness of such packages becomes evident and they are then easely beaten by a human vectorizer. Still, there is the advantage when a vectorizer is available for different machines - like VAST is - that it allows writing transportable FORTRAN and still get good results. For more on the subject see [2].

At a higher level, we see that there are, fortunately, some collections of standard routines which are vectorized. So an application programmer does not always have to keep the machine details in mind, because they are catered for already. One very basic collection is the Basic Linear Algebra Subprograms [22]. An optimized version of this BLAS should be available on every computer. The advantage would be that a programmer does not have to cope with problems that are already solved. (Note that with the BLAS some previous examples are artificial problems.) Moreover the use of the BLAS can be the beginning of writing portable software which will perform good on various machines.

Vectorization may not always be paying off. As early as from 1967 'Amdahls law' [1] warns us that the overall performance of a computer with a high-speed and a low-speed mode of operation, will be dominated by the low-speed mode, unless the fraction of the results, generated in low-speed mode can be eliminated. In [38] this is discussed and illustrated. These ideas can be expressed somewhat more formally by a simple analytic model [17]:

$$1/T = 1/(F_H T_H + F_L R_L),$$

where

$1/T$ = results generated per unit time;

F = the fraction of the results generated;

T = the time to generate a single result;

H refers to the high-speed mode and

L refers to the low-speed mode.

When plotted this looks like:

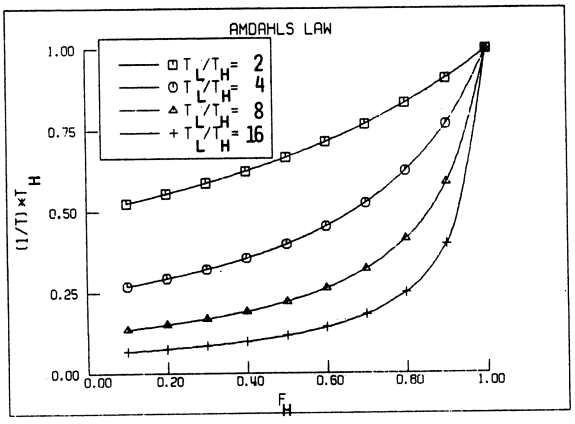


Fig. 3

Remark that vectorization is equivalent to moving to the right along the horizontal axis. Thus the paradoxical conclusion is that vectorization pays most when a computer has a relatively fast scalar operation mode.

7. ALGORITHMS

Many techniques that are used with good results on a serial computer are sequential algorithms and are therefore seldom suitable for use on vector computers. So for vector computers it is usually not enough to concentrate on such aspects as minimising the number of arithmetic operations, but we must pay close attention to the details of the coding, or perhaps even re-consider our choise of algorithm. In this section we will try to show that graceful use of new hardware can lead, by re-investigating from scratch, to new forms of 'old' algorithms. As a first example - rather simple and of no practical value - we will look at the calculations of factorials. A tridiagonal linear system of equations will serve as a second example; to make it stable, we suppose the system to be diagonally dominant. Such systems often occur as finite-difference approximations to the solution of differential equations with second derivatives. (A consequence of previous statements is that one should investigate the solution and/or discretisation method - as is done in [8] - but we follow the mainstream here.)

When writing a program for the calculation of some factorials one tends to use the recurrence relation for n factorial, n! = n*(n-1)!, as

the underlying algorithm. As already shown, this cannot go through the pipes of a vector computer. But a vector of contiguous factorials can be calculated by multiplying the result vector again and again by another vector, every time from a 'lower' starting point:

```
( 1)          ( 1)          ( 1)
( 2)  (1)     ( 2)          ( 2)
( 3)  (2)     ( 6)   (1)    ( 6)
( 4) * (3) → ( 12) * (2) → ( 24) * (1) → ...
( 5)  (4)     ( 20)  (3)    ( 60)  (2)
( 6)  (5)     ( 30)  (4)    (120)  (3)
( . )  (.)    ( . )  (.)    ( . )  (.)
( . )  (.)    ( . )  (.)    ( . )  (.)
( . )  (.)    ( . )  (.)    ( . )  (.)
```

We see, that by 'thinking in vectors' we have constructed a vectorizable algorithm.

In linear algebra we frequently try to solve the n by n system

$$Ax = b$$

by transforming A (and b with it)

$$A = A^{(1)} \to A^{(2)} \to \ldots \to A^{(n)} = U$$

to a form - *exampli gratis* upper triangular - that can be solved directly. This can be found in [9], or any other book that contains an introduction to linear algebra. No doubt there can also be found a classical transforming method, the Gaussian elimination. We recall that the tridiagonal system is only an example to show the 'remake' of algorithms; we do not treat other solution methods - for example iterative ones - for linear systems here.

Another basic, but not so well known, concept is the notation of a matrix in terms of diagonals [26]. We then write the (in this example tridiagonal) matrix as a sum of matrices with only one diagonal of non-zero entries:

$$A = A_{-1} + A_0 + A_1.$$

This sum-of-diagonal notation makes matrix multiplication transparent, because

$$A_k \times B_l = C_{k+l}.$$

Of course it is convenient to store only the diagonals as vectors of contiguous data.

We now apply the Gaussian elimination. In this case this means the elimination of the lower subdiagonal by adding a row multiplied by a suitable scalar to the next row, as follows:

$$U_0(i+1) = A_0(i+1) - A_1(i) \times A_{-1}(i)/U_0(i).$$

The previous element must be known before the present one is computed; so this form of the algorithm is recursive and as already pointed out, it will perform poorly on a vector computer. On top of that there is a - also recursive - back substitution to follow. In order to construct a vectorizable variant we observe that the recurrence relation for the elements of the main diagonal of U defines a continued fraction. As is well known [14], continued fractions are closely related to linear three-term recurrence relations. The calculation of the recurrence terms involves calculations of products, as in the factorial example. This can be parallellized by the recursive doubling technique [34], the final result is that we can transform A (and b) as follows:

$$A^{(2)} = (I-A_{-1}A_0^{-1})A = -A_{-1}A_0^{-1}A_{-1} + A_0 - A_{-1}A_0^{-1}A_1 + A_1 = A_{-2}^{(2)} + A_0^{(2)} + A_1^{(2)}.$$

If we treat the upper subdiagonal in the same way, we can keep on doubling the distance between the main diagonal and the subdiagonals until we have pushed the latters over the cliff. Comparisons have been done on a STAR 100 [21]. The stability of this algorithm is analyzed in [7].

For another algorithm, we suppose that we apply the permutation matrix, P, which numbers the odd rows first, to a tridiagonal matrix, of even order

$$PAP^T = \begin{array}{|c|c|} \hline S_0 & U_{-1}+U_0 \\ \hline V_0+V_1 & T_0 \\ \hline \end{array}$$

Now we can use a well known result in the theory of partitional matrices and get

$$(PAP^T)^{-1} = \begin{array}{|c|c|} \hline S^{-1} + S^{-1}UYVS^{-1} & -S^{-1}UY \\ \hline -YVS^{-1} & Y \\ \hline \end{array}$$

with $Y = (T-VS^{-1}U)^{-1}$.

Unattractive as this may look, we have halved the order of the problem and all the matrix products can be conceived as vector products. Of course, one does not calculate the inverse matrices but performs decompositions and back substitutions. This gives good results on a vector computer [21]. More general methods are treated in [27] and [31]. A combination of this method and the previous one can be found in [10]. More elaborate reports on the influence of architecture on (the choice of) algorithms are [32] and [36].

8. EPILOGUE

Numerical analysts often display a tendency to proclaim themselves independent of 'the' computer. This may be a good approach when dealing (first?) with a problem at 'high' level. An aim of this talk however, was to show that - at 'low' level - this attitude does not always yield good performance (and that is what both mathematicians and supercomputers are built for).

Another motivating issue is that parallel processors are here to stay in scientific computers; vector computers are not a passing fad and the scientist should learn how to utilize them.

REFERENCES

[1] AMDAHL, G., *Validity of the Single Processor Approach to Achieving Large-Scale Computing Requirements*, Comput. Des. 6, no. 12, 39-40, 1967.

[2] ARNOLD, C.N., *Performance Evaluation of Three Automatic Vectorizer Packages*, Control Data Corp.

[3] BURKE, P.G. & L.M. DELVES (eds.), *Vector and Parallel Processors in Computational Science*, Computer Physics Comm. 26, no. 3&4, 1982.

[4] BUZBEE, B.L., R.H. EWALD & W.J. WORLTON, *Japanese Supercomputer Technology*, Science 218, 1189-1193, 1982.

[5] COMPTE, D. & J.-C. SYRE, *Les Supercalculateurs*, La Recherche 14, 1084-1095, 1983.

[6] DONGARRA, J.J. & A.R. HINDS, *Unrolling Loops in FORTRAN*, Software-Practice and Experience 9, 219-226, 1979.

[7] DUBOIS, P. & G.H. RODRIGUE, *An analysis of the Recursive Doubling Algorithm*, in [20], 299-306.

[8] EVANS, D.J., *A New Explicit Method for the Diffusion Equation*, to appear.

[9] FORSYTHE, G.E., M.A. MALCOLM & C.B. MOLER, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, 1977.

[10] HELLER, D., *Some Aspects of the Cyclic Reduction Algorithm for Block Tridiagonal Linear Systems*, SIAM J. Numer. Anal. 13, 484-496, 1976.

[11] HIGBIE, L., *A Vector Processing Tutorial*, Datamation 29, no. 8, 180-200, 1983.

[12] HOCKNEY, R.W. & C.R. JESSHOPE, *Parallel Computers*, Adam Hilgers, Bristol, 1981.

[13] HOCKNEY, R.W., *Characterization of Parallel Computers and Algorithms*, in [3], 285-291.

[14] HOLLENBERG, J.P., *Continued Fractions*, Approximation of Functions, C.G. van der Laan & N.M. Temme (eds.), MC Syllabus, to appear.

[15] HWANG, K., S.-P. SU & L.M. NI, *Vector Computer Architecture and Processing Techniques*, Advances in Computers 20, M.C. Yovits (ed.), Acad. Press, 115-197, 1981.

[16] JOHNSON, P.M., *An Introduction to Vector Processing*, Comput. Des. 17, no. 2, 89-97, 1978.

[17] JORDAN, T.L., *A Guide to Parallel Computations and Some CRAY-1 Experiences*, to appear in Methods of Computational Physics, Academic Press.

[18] KASCIC, M.J. Jr., *Vector Processing Problem or Opportunity?*, IEEE Compcon, 1980.

[19] KASCIC, M.J. Jr., *Vector Processing on the Cyber 200*, Angew. Inf. 22, 27-37, 1980.

[20] KUCK, D.J., D.H. LAWRIE & A.H. SAMEH (eds.), *High Speed Computer and Algorithm Organization*, Academic Press, 1977.

[21] LAMBIOTTE, J.J. Jr. & R.G. VOIGT, *The Solution of Tridiagonal Linear Systems on the CDC STAR 100 Computer*, ACM Trans. Math. Software 1, 308-329, 1975.

[22] LAWSON, C., R. HANSON, D. KINCAID & F. KROGH, *Basic Linear Algebra Subprograms for FORTRAN Usage*, ACM Trans. Math. Software 5, 308-371, 1979.

[23] LAX, P.D. (chairman), *Report of the Panel on Large Scale Computing in Science and Engineering*, National Science Foundation, Washington, 1982.

[24] LAZOU, C., *Vectorization on the CRAY 1 - a Case Study*, IUCC Bulletin 5, 2-6, 1983.

[25] LEVINE, R.D., *Supercomputers*, Sci. Amer. 246, 118-135, 1982.

[26] MADSEN, N.K., G.H. RODRIGUE & J.I. KARUSH, *Matrix Multiplication by Diagonals on a Vector-Parallel Processor*, Inform. Proc. Lett. 5, 41-45, 1976.

[27] MADSEN, N.K. & G.H. RODRIGUE, *Odd-Even Reduction for Pentadiagonal Matrices*, Parallel Computers - Parallel Mathematics, M. Feilmeijer (ed.), North Holland Publishing Company, 103-108, 1977.

[28] PSR, *Efficient FORTRAN Techniques for Vector Processors; Student Workbook*, Pacific Sierra Research Inc., Santa Monica, 1983.

[29] PURCELL, C.J., *Using the Cyber 205*, in [3], 249-251.

[30] RAMAMOORTY, C.V. & H.F. LI, *Pipeline Architecture*, Comput. Surveys 9, 61-102, 1977.

[31] RODRIGUE, G.H., N.K. MADSEN & J.I. KARUSH, *Odd-Even Reduction for Banded Linear Equations*, J. Ass. Comput. Mach. 26, 72-81, 1979.

[32] RODRIGUE, G.H., *The Influence of Computer Architecture on Numerical Algorithms*, LLL Report UCRL-85679, 1983.

[33] RUSSELL, R.M., *The CRAY-1 Computer System,* Comm. ACM 21, 63-72, 1978.

[34] STONE, H.S., *Parallel Tridiagonal Equation Solvers,* ACM Trans. Math. Software 1, 289-307, 1975.

[35] TEMPERTON, C., *CRAY-1 V. Cyber 205: Some Comparisons,* SIGNUM Newsletter 18, no. 2, 20-21, 1983.

[36] VOIGT, R.G., *The Influence of Vector Computer Architecture on Numerical Algorithms,* in [20], 229-244.

[37] VORST, H.A. van der & J.M. van KATS, *Comparative Performance Tests on the CRAY-1 and the Cyber 205,* ACCU-reeks no. 36, ACCU, Utrecht, 1983.

[38] WORLTON, J., *A Philosophy of Supercomputing,* Report LA-8849-MS, Los Alamos, 1981.

MC SYLLABI

1.1 F. Göbel, J. van de Lune. *Leergang besliskunde, deel 1: wiskundige basiskennis.* 1965.

1.2 J. Hemelrijk, J. Kriens. *Leergang besliskunde, deel 2: kansberekening.* 1965.

1.3 J. Hemelrijk, J. Kriens. *Leergang besliskunde, deel 3: statistiek.* 1966.

1.4 G. de Leve, W. Molenaar. *Leergang besliskunde, deel 4: Markovketens en wachttijden.* 1966.

1.5 J. Kriens, G. de Leve. *Leergang besliskunde, deel 5: inleiding tot de mathematische besliskunde.* 1966.

1.6a B. Dorhout, J. Kriens. *Leergang besliskunde, deel 6a: wiskundige programmering 1.* 1968.

1.6b B. Dorhout, J. Kriens, J.Th. van Lieshout. *Leergang besliskunde, deel 6b: wiskundige programmering 2.* 1977.

1.7a G. de Leve. *Leergang besliskunde, deel 7a: dynamische programmering 1.* 1968.

1.7b G. de Leve, H.C. Tijms. *Leergang besliskunde, deel 7b: dynamische programmering 2.* 1970.

1.7c G. de Leve, H.C. Tijms. *Leergang besliskunde, deel 7c: dynamische programmering 3.* 1971.

1.8 J. Kriens, F. Göbel, W. Molenaar. *Leergang besliskunde, deel 8: minimaxmethode, netwerkplanning, simulatie.* 1968.

2.1 G.J.R. Förch, P.J. van der Houwen, R.P. van de Riet. *Colloquium stabiliteit van differentieschema's, deel 1.* 1967.

2.2 L. Dekker, T.J. Dekker, P.J. van der Houwen, M.N. Spijker. *Colloquium stabiliteit van differentieschema's, deel 2.* 1968.

3.1 H.A. Lauwerier. *Randwaardeproblemen, deel 1.* 1967.

3.2 H.A. Lauwerier. *Randwaardeproblemen, deel 2.* 1968.

3.3 H.A. Lauwerier. *Randwaardeproblemen, deel 3.* 1968.

4 H.A. Lauwerier. *Representaties van groepen.* 1968.

5 J.H. van Lint, J.J. Seidel, P.C. Baayen. *Colloquium discrete wiskunde.* 1968.

6 K.K. Koksma. *Cursus ALGOL 60.* 1969.

7.1 *Colloquium moderne rekenmachines, deel 1.* 1969.

7.2 *Colloquium moderne rekenmachines, deel 2.* 1969.

8 H. Bavinck, J. Grasman. *Relaxatietrillingen.* 1969.

9.1 T.M.T. Coolen, G.J.R. Förch, E.M. de Jager, H.G.J. Pijls. *Colloquium elliptische differentiaalvergelijkingen, deel 1.* 1970.

9.2 W.P. van den Brink, T.M.T. Coolen, B. Dijkhuis, P.P.N. de Groen, P.J. van der Houwen, E.M. de Jager, N.M. Temme, R.J. de Vogelaere. *Colloquium elliptische differentiaalvergelijkingen, deel 2.* 1970.

10 J. Fabius, W.R. van Zwet. *Grondbegrippen van de waarschijnlijkheidsrekening.* 1970.

11 H. Bart, M.A. Kaashoek. H.G.J. Pijls, W.J. de Schipper, J. de Vries. *Colloquium halfalgebra's en positieve operatoren.* 1971.

12 T.J. Dekker. *Numerieke algebra.* 1971.

13 F.E.J. Kruseman Aretz. *Programmeren voor rekenautomaten; de MC ALGOL 60 vertaler voor de EL X8.* 1971.

14 H. Bavinck, W. Gautschi, G.M. Willems. *Colloquium approximatietheorie.* 1971.

15.1 T.J. Dekker, P.W. Hemker, P.J. van der Houwen. *Colloquium stijve differentiaalvergelijkingen, deel 1.* 1972.

15.2 P.A. Beentjes, K. Dekker, H.C. Hemker, S.P.N. van Kampen, G.M. Willems. *Colloquium stijve differentiaalvergelijkingen, deel 2.* 1973.

15.3 P.A. Beentjes, K. Dekker, P.W. Hemker, M. van Veldhuizen. *Colloquium stijve differentiaalvergelijkingen, deel 3.* 1975.

16.1 L. Geurts. *Cursus programmeren, deel 1: de elementen van het programmeren.* 1973.

16.2 L. Geurts. *Cursus programmeren, deel 2: de programmeertaal ALGOL 60.* 1973.

17.1 P.S. Stobbe. *Lineaire algebra, deel 1.* 1973.

17.2 P.S. Stobbe. *Lineaire algebra, deel 2.* 1973.

17.3 N.M. Temme. *Lineaire algebra, deel 3.* 1976.

18 F. van der Blij, H. Freudenthal, J.J. de Iongh, J.J. Seidel, A. van Wijngaarden. *Een kwart eeuw wiskunde 1946-1971, syllabus van de vakantiecursus 1971.* 1973.

19 A. Hordijk, R. Potharst, J.Th. Runnenburg. *Optimaal stoppen van Markovketens.* 1973.

20 T.M.T. Coolen, P.W. Hemker, P.J. van der Houwen, E. Slagt. *ALGOL 60 procedures voor begin- en randwaardeproblemen.* 1976.

21 J.W. de Bakker (red.). *Colloquium programmacorrectheid.* 1975.

22 R. Helmers, J. Oosterhoff, F.H. Ruymgaart, M.C.A. van Zuylen. *Asymptotische methoden in de toetsingstheorie; toepassingen van naburigheid.* 1976.

23.1 J.W. de Roever (red.). *Colloquium onderwerpen uit de biomathematica, deel 1.* 1976.

23.2 J.W. de Roever (red.). *Colloquium onderwerpen uit de biomathematica, deel 2.* 1977.

24.1 P.J. van der Houwen. *Numerieke integratie van differentiaalvergelijkingen, deel 1: eenstapsmethoden.* 1974.

25 *Colloquium structuur van programmeertalen.* 1976.

26.1 N.M. Temme (ed.). *Nonlinear analysis, volume 1.* 1976.

26.2 N.M. Temme (ed.). *Nonlinear analysis, volume 2.* 1976.

27 M. Bakker, P.W. Hemker, P.J. van der Houwen, S.J. Polak, M. van Veldhuizen. *Colloquium discretiseringsmethoden.* 1976.

28 O. Diekmann, N.M. Temme (eds.). *Nonlinear diffusion problems.* 1976.

29.1 J.C.P. Bus (red.). *Colloquium numerieke programmatuur, deel 1A, deel 1B.* 1976.

29.2 H.J.J. te Riele (red.). *Colloquium numerieke programmatuur, deel 2.* 1977.

30 J. Heering, P. Klint (red.). *Colloquium programmeeromgevingen.* 1983.

31 J.H. van Lint (red.). *Inleiding in de coderingstheorie.* 1976.

32 L. Geurts (red.). *Colloquium bedrijfsystemen.* 1976.

33 P.J. van der Houwen. *Berekening van waterstanden in zeeën en rivieren.* 1977.

34 J. Hemelrijk. *Oriënterende cursus mathematische statistiek.* 1977.

35 P.J.W. ten Hagen (red.). *Colloquium computer graphics.* 1978.

36 J.M. Aarts, J. de Vries. *Colloquium topologische dynamische systemen.* 1977.

37 J.C. van Vliet (red.). *Colloquium capita datastructuren.* 1978.

38.1 T.H. Koornwinder (ed.). *Representations of locally compact groups with applications, part I.* 1979.

38.2 T.H. Koornwinder (ed.). *Representations of locally compact groups with applications, part II.* 1979.

39 O.J. Vrieze, G.L. Wanrooy. *Colloquium stochastische spelen.* 1978.

40 J. van Tiel. *Convexe analyse.* 1979.

41 H.J.J. te Riele (ed.). *Colloquium numerical treatment of integral equations.* 1979.

42 J.C. van Vliet (red.). *Colloquium capita implementatie van programmeertalen.* 1980.

43 A.M. Cohen, H.A. Wilbrink. *Eindige groepen (een inleidende cursus).* 1980.

44 J.G. Verwer (ed.). *Colloquium numerical solution of partial differential equations.* 1980.

45 P. Klint (red.). *Colloquium hogere programmeertalen en computerarchitectuur.* 1980.

46.1 P.M.G. Apers (red.). *Colloquium databankorganisatie, deel 1.* 1981.

46.2 P.G.M. Apers (red.). *Colloquium databankorganisatie, deel 2.* 1981.

47.1 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60: general information and indices.* 1981.

47.2 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 1: elementary procedures; vol. 2: algebraic evaluations.* 1981.

47.3 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 3A: linear algebra, part I.* 1981.

47.4 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 3B: linear algebra, part II.* 1981.

47.5 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 4: analytical evaluations; vol. 5A: analytical problems, part I.* 1981.

47.6 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 5B: analytical problems, part II.* 1981.

47.7 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 6: special functions and constants; vol. 7: interpolation and approximation.* 1981.

48.1 P.M.B. Vitányi, J. van Leeuwen, P. van Emde Boas (red.). *Colloquium complexiteit en algoritmen, deel 1.* 1982.

48.2 P.M.B. Vitányi, J. van Leeuwen, P. van Emde Boas (red.). *Colloquium complexiteit en algoritmen, deel 2.* 1982.

49 T.H. Koornwinder (ed.). *The structure of real semisimple Lie groups.* 1982.

50 H. Nijmeijer. *Inleiding systeemtheorie.* 1982.

51 P.J. Hoogendoorn (red.). *Cursus cryptografie.* 1983.

CWI SYLLABI

1 Vacantiecursus 1984 *Hewet - plus wiskunde.* 1984.

2 E.M. de Jager, H.G.J. Pijls (eds.). *Proceedings Seminar 1981-1982. Mathematical structures in field theories.* 1984.

3 W.C.M. Kallenberg, et.al. *Testing statistical hypotheses: worked solutions.* 1984.

4 J.G. Verwer (ed.). *Colloquium topics in applied numerical analysis, volume 1.* 1984.

5 J.G. Verwer (ed.). *Colloquium topics in applied numerical analysis, volume 2.* 1984.