CLM - P 442

CLM - P 442

L

UKAEA RESEARCH GROUP

Preprint

ADIABATIC RELAXATION TO 1D MHD PRESSURE EQUILIBRIUM.

EQUIL: A FORTRAN MODULE AND TEST PROGRAM

K V ROBERTS
J P CHRISTIANSEN
J W LONG

CULHAM LABORATORY
Abingdon Oxfordshire

1975

CLM-P442

ADIABATIC RELAXATION TO 1D MHD PRESSURE EQUILIBRIUM.

EQUIL: A FORTRAN MODULE AND TEST PROGRAM

K V Roberts, J P Christiansen

Culham Laboratory, Abingdon, Oxon OX14 3DB, UK.
(Euratom/UKAEA Fusion Association)

and

J W Long

Oxford Polytechnic, Headington, Oxon OX3 OBP, UK.

ABSTRACT

A numerical method is described by which a 1D MHD non-linear Lagrangian system may be relaxed adiabatically to pressure equilibrium by an iterative process. The method is appropriate for problems involving a slow passage through a sequence of quasi-equilibrium states, and although described here in the context of the cylindrical plasma pinch it should be equally appropriate in other contexts, e.g. stellar evolution. The rate of convergence of the iterations can be given a simple physical interpretation and is very rapid: it corresponds to squaring the residual error at every iteration step. Results of numerical tests in double precision are given. Program EQUIL described in this paper represents one module of a larger plasma equilibrium and diffusion code ATHENE 1 which will be published in full elsewhere.

As a demonstration of program construction and documentation methods, the module has been developed and tested using an on-line utility, and a listing of the code is given together with a discussion of the programming techniques that have been found useful. The Fortran listing is reproduced in parallel with a version of the code in ordinary mathematical notation which is easier to follow. It is believed that this documentation technique could have many applications.

(To be published in Computer Physics Communications)

October 1975

CONTENTS

		page
1.	Introduction	1
2.	Basic Equations	2
3.	Mesh	3
4.	Physical Model	4
5.	Fractional Step Method	5
6.	Linearization of the Equilibrium Equation	6
7.	Difference Scheme	7
8.	Mesh Adjustment	9
9.	Adiabatic Updating of Variables and Convergence Test	9
10.	Convergence Analysis	10
11.	Test Results	11
12.	Documentation of Fortran Programs	12
Appendix A.	Program EQUIL	A1
Appendix B.	On-line Programming Techniques	B1

References

1. INTRODUCTION

Several 1-dimensional calculations of interest in hydrodynamics and magnetohydrodynamics involve the slow passage of a physical system through a sequence of states which are in almost exact pressure equilibrium. Examples occur in the evolution of stars [1] and in the time development of a high-β toroidal CTR plasma in the cylindrical approximation. An equilibrium model should be valid provided that

$$T \gg R/V_s$$

where T is the evolutionary timescale, R is the radius of the system and V_s is the acoustic or magnetosonic speed. As usual β is the ratio of particle to magnetic pressure.

The main purpose of this paper is to describe in some detail the method used to achieve pressure equilibrium at each successive timestep of the Culham 1D MHD fully-ionized plasma equilibrium and diffusion code ATHENE 1 [2]. This section of the code constitutes a module EQUIL which can conveniently be discussed, developed and tested on its own. We explain the numerical method, briefly discuss the theoretical and measured rate of convergence to pressure equilibrium (which is found to be quadratic), and describe the difference scheme and the program notation. Only the CTR problem will be dealt with here although the method of solution is more general. The full ATHENE 1 code will be published elsewhere [2]. Later versions in the ATHENE series will add further physical effects such as finite inertia (ATHENE 2 code [3]) and partial ionization.

A secondary purpose is to illustrate, by means of a concrete example, some of the documentation and programming techniques which have proved useful. Computational physics relies on the development and dissemination of algorithms and programs, but at present a serious barrier is imposed by the limitations of existing programming languages and by the restricted character sets that are available. These are evidently much less general than the physicist is accustomed to use in his ordinary theoretical work, and they do in fact make algorithms and programming methods quite hard to discuss, to record, to communicate and to teach.

To resolve this impasse we resort to the simple practical expedient of reproducing the Fortran code (Appendix A) in two parallel versions which are in 1-1 correspondence with one another. Version I uses the informal notation of standard mathematics including lower-case and Greek symbols, superscripts and subscripts, special mathematical signs and so on, and is therefore easy to understand; it can where necessary be supplemented by diagrams, tables etc. like an ordinary mathematical text. Version II is a straightforward Fortran listing, although in this particular case it has been produced on a commercial timesharing system which allows certain useful extensions of the language, and the comments have been printed in lower case for clarity. Each version can be used to help interpret the other. We believe that this technique can usefully be applied to other types of program and to other programming languages, and a fuller discussion is given in §12.

The commercial timesharing system used for the version of EQUIL described in this paper is the General Electric Mark III International Information Service marketed in the UK by Honeywell. Appendix B describes the programming techniques that have been developed for on-line use and indicates how some of the OLYMPUS conventions [4,5] have been adapted for this purpose.

2. BASIC EQUATIONS

We use a Lagrangian description of the plasma so that the equation of motion in SI units is

$$\rho \frac{D\underline{u}}{Dt} = - \underline{\nabla} P + \underline{J} \times \underline{B} \tag{1}$$

where ρ is the density, $\underline{u}$ the velocity, P the pressure, $\underline{B}$ the magnetic field and $\underline{J} = \text{curl}\ \underline{B}/\mu_o$ is the current density. Viscous terms are omitted because the motion is assumed to be very slow. Equation (1) is supplemented in ATHENE 1 by other equations describing heat conduction, magnetic field diffusion, bremsstrahlung, thermonuclear reactions and other processes in which entropy is not conserved. These are not included in EQUIL.

The inertial term on the left-hand side of (1) can be neglected if the motion is sufficiently slow and in this case the equilibrium equation

$$\underline{\nabla} P = \underline{J} \times \underline{B} \tag{2}$$

is to be satisfied at all times. A simple physical picture of the evolution of the plasma is that the entropy-generating processes continually cause a departure from exact pressure equilibrium, which is restored by a small displacement $\underline{\xi} = \underline{u}\ Dt$ that generates an adiabatic change in the density, electron and ion temperatures, pressure and magnetic field :

$$D\rho = - \rho\, \underline{\nabla} . \underline{\xi} \tag{3}$$

$$D T_e = -(\gamma-1) T_e\, \underline{\nabla} . \underline{\xi} \tag{4}$$

$$D T_i = -(\gamma-1) T_i\, \underline{\nabla} . \underline{\xi} \tag{5}$$

$$DP = -\gamma P \underline{\nabla} . \underline{\xi} \tag{6}$$

$$D\underline{B} = \underline{\nabla} \times (\underline{\xi} \times \underline{B}) \quad . \tag{7}$$

Here the density and pressure are related to the electron and ion particle densities N_e, N_i by

$$\rho = N_e m_e + N_i m_i \tag{8}$$

$$P = N_e k T_e + N_i k T_i \tag{9}$$

subject to the charge-neutrality condition

$$N_e = Z N_i \quad . \tag{10}$$

We assume for simplicity that there is a single ionic species with charge number $Z = 1$ so that $N_e = N_i = N$, and that the electron and ion temperatures are isotropic with the same specific heat ratio γ , but there is no difficulty in extending the method to more general situations provided that suitable adiabatic relations can be defined.

The ATHENE 1 code uses cylindrical polar coordinates (r,θ,z) with all functions depending only on r , so that equations (2) - (7) become

$$\frac{\partial p}{\partial r} + \frac{1}{\mu_o}\left(\frac{B_\theta}{r}\frac{\partial}{\partial r}(r B_\theta) + B_z \frac{\partial B_z}{\partial r}\right) = 0 \tag{11}$$

$$DN = -N . \frac{1}{r} \frac{\partial}{\partial r}(r\xi) \tag{12}$$

$$DT_e = -(\gamma-1)T_e . \frac{1}{r}\frac{\partial}{\partial r}(r\xi) \tag{13}$$

$$DT_i = -(\gamma-1)T_i . \frac{1}{r}\frac{\partial}{\partial r}(r\xi) \tag{14}$$

$$DP = -\gamma P \frac{1}{r}\frac{\partial}{\partial r}(r\xi) \tag{15}$$

$$DB_\theta = -B_\theta \frac{\partial \xi}{\partial r} \tag{16}$$

$$DB_z = -B_z . \frac{1}{r}\frac{\partial}{\partial r}(r\xi) \tag{17}$$

where $\xi = Dr$. It is also convenient to employ the integral conservation laws (between any two radial limits r_a, r_b) :

$$D\int_{r_a}^{r_b} N dr = D\int_{r_a}^{r_b} B_\theta dr = D\int_{r_a}^{r_b} B_z r dr = 0 \tag{18}$$

and the adiabatic relations:

$$D(T_e \rho^{\gamma-1}) = D(T_i \rho^{\gamma-1}) = D(P \rho^{\gamma}) = 0 \quad . \tag{19}$$

3. MESH

A cylindrical Lagrangian mesh

$$0 = R_1, R_2, \ldots \quad R_{NINT}, R_{NINT+1} = R_W \tag{20}$$

is used as illustrated in Fig.1 where R_W is the fixed inner radius of the wall surrounding the plasma and NINT is the number of mesh intervals. Physical variables N, T_e, T_i, P, B_θ, B_z are defined at the half-integral radii (shown dashed in the figures)

$$R_{j+\frac{1}{2}} \equiv \tfrac{1}{2}(R_j + R_{j+1}), \quad j = 1,2, \ldots \text{NINT} \tag{21}$$

which are recomputed each time the mesh is moved. These variables are to be regarded as cell averages, so that for example the particle numbers and masses, fluxes and energies associated with cell $j+\frac{1}{2}$ are given (apart from constant numerical factors) by expressions such as

$$N\Delta A, \ B_\theta \Delta R, \ B_z \Delta A, \ \frac{P}{\gamma-1}\Delta A \tag{22}$$

where

$$\Delta R \equiv \Delta R_{j+\frac{1}{2}} = R_{j+1} - R_j \tag{23}$$

$$\Delta A \equiv \Delta A_{j+\frac{1}{2}} = R_{j+\frac{1}{2}} \Delta R_{j+\frac{1}{2}} = \tfrac{1}{2}(R_{j+1}^2 - R_j^2) \quad . \tag{24}$$

This guarantees exact mass and flux conservation. The subscript $j+\frac{1}{2}$ has been omitted in (22) for brevity.

4. PHYSICAL MODEL

Physically we picture the mesh (20) as an idealized mechanical, electrical and thermal system with the following properties :

(a) R_1 is an infinitely thin wire fixed at the axis, $r = o$, which is a perfect electrical conductor and has zero thermal capacity.

(b) R_2, ... R_{NINT} are thin massless shells, free to expand or contract radially but constrained to remain cylindrical, which are both perfect electrical conductors and perfect thermal insulators. Their radial motion is assumed to be damped.

(c) R_{NINT+1} is a fixed wall of radius R_W which is a perfect electrical conductor and a perfect thermal insulator.

It is intuitively clear that if an arbitrary set of individual plasma masses, electron and ion entropies and B_θ and B_z fluxes is distributed amongst the NINT cells, then the NINT-1 interior moveable shells will in general find themselves acted on by non-zero forces and will therefore begin to move, thus doing work against the damping forces and lowering the total energy of the system. This process will continue until a local minimum energy state is reached, i.e. an equilibrium which is stable against purely radial displacements. Although energy has been lost by the system the process is nevertheless adiabatic since all the cell masses, entropies and fluxes remain unchanged.

5. FRACTIONAL STEP METHOD

Each timestep of the ATHENE 1 code is divided into 2 stages, or 'fractional steps'.

Stage A. The difference equations for the entropy-changing processes are solved with a fixed non-uniform Eulerian mesh, defined by the positions of the Lagrangian coordinates at the end of the previous timestep. During Stage A the variables P, B_θ, B_z alter slightly, so causing a departure from pressure equilibrium, i.e. equation (11) is no longer exactly satisfied.

The appropriate physical picture for this stage is that the cell boundaries (20) now represent thin fixed layers of finite thermal insulation with finite electrical resistance, which therefore allow some heat and magnetic flux (but no plasma) to pass through. The model is to be considered as a 'lumped' system in which all temperature differences and electrical currents are concentrated in these layers. The ohmic heat generated in each layer is appropriately divided amongst the two cells on either side, suitable provision being made at the boundaries $r = o$ and $r = R_W$. The inner boundary $r = o$ does not of course allow any heat or B_z flux to pass through or carry any J_θ current, but it does carry a J_z current and allow B_θ flux to leave the system. Other entropy-changing processes are bremsstrahlung, synchrotron radiation, thermonuclear burning and electron-ion equipartition, although the latter does not cause any departure from pressure equilibrium if the electrons and ions have the same specific heat ratio γ.

Stage B. In Stage B, which is solved by the EQUIL module and is the subject of this paper, the entropy-changing processes are switched off and the interior cell boundaries are allowed to move according to

$$R_j^* = R_j + \xi_j \ , \quad (j = 2,3 \ldots \text{NINT}) \tag{25}$$

until pressure equilibrium is reached. During this stage the variables are adjusted adiabatically according to equations (12)-(19). Since equation (1) is non-linear an iterative approach is used in which equilibrium is reached by a sequence of successive mesh displacements

$$R_j^{m+1} = R_j^{m} + \xi_j^{m+1} \ , \quad (R_j^{o} = R_j) \tag{26}$$

with

$$R_j^* = \lim_{m \to \infty} R_j^{m+1} \tag{27}$$

and

$$\xi_j = \sum_{m=1}^{\infty} \xi_j^m \quad .$$

The adiabatic variation during each iteration step is expressed by

$$P^{m+1} (\Delta A^{m+1})^\gamma = P^m (\Delta A^m)^\gamma \tag{28}$$

$$B_\theta^{m+1} \Delta r^{m+1} = B_\theta^m \Delta r^m \tag{29}$$

$$B_z^{m+1} \Delta A^{m+1} = B_z^m \Delta A^m \tag{30}$$

and similarly for N, T_e, T_i although in fact it is not necessary to recompute these variables until the solution of (11) has been obtained. In (28) - (30) the subscript $j+\frac{1}{2}$ has again been omitted throughout.

6. LINEARIZATION OF THE EQUILIBRIUM EQUATION

Given a set of variables R^m, P^m, B_θ^m, B_z^m which approximately satisfy the equilibrium equation (11) after iteration step m, our aim in this section is to find a linearized equation to determine the displacement ξ^{m+1}. Actually it is preferable to replace (11) by the equivalent equation

$$\frac{\partial}{\partial r}\left(P + \frac{B_\theta^2 + B_z^2}{2\mu_o} \right) + \frac{B_\theta^2}{\mu_o r} = 0 \tag{31}$$

since otherwise the factor r which multiplies B_θ in the second term would cause trouble: there is no exact adiabatic relation for $R_{j+\frac{1}{2}}$ which has to be redefined at the end of each iteration step using (21).

Because the $\partial/\partial r$ operator which acts on the first term has to be varied, it is convenient to write (31) temporarily as

$$\Delta\left(P + \frac{B_\theta^2 + B_z^2}{2\mu_o} \right) + \frac{(\tilde{B}_\theta)^2 \Delta r}{\mu_o r} = 0 \quad . \tag{32}$$

Here ~ denotes an average over adjacent half-integral points, and

$$\Delta R_j = R_{j+\frac{1}{2}} - R_{j-\frac{1}{2}} \equiv \tfrac{1}{2}(R_{j+1} + R_{j-1}) \quad . \tag{33}$$

We now assume that (32) is to hold for the 5 quantities r^{-1}, Δr, P, B_θ, B_z at step (m+1), using the linearized adiabatic relations

$$P^{m+1} = P^m \left(1 - \gamma \frac{1}{r^m} \frac{\partial}{\partial r^m} (r^m \xi^{m+1}) \right) \tag{34}$$

$$(B_\theta)^{m+1} = (B_\theta)^m \left(1 - \frac{\partial \xi^{m+1}}{\partial r^m} \right) \tag{35}$$

$$(B_z)^{m+1} = (B_z)^m \left(1 - \frac{1}{r^m} \frac{\partial}{\partial r^m} (r^m \xi^{m+1}) \right) \tag{36}$$

$$\frac{1}{r^{m+1}} = \frac{1}{r^m} \left(1 - \frac{\xi^{m+1}}{r^m} \right) \tag{37}$$

$$\Delta r^{m+1} = \Delta r^m \left(1 + \frac{\partial \xi^{m+1}}{\partial r^m} \right) \tag{38}$$

and drop all higher-order terms.

Dividing through by Δr once more, we obtain the inhomogeneous elliptic equation for ξ^{m+1}

$$\underset{[1]}{-\frac{\partial}{\partial r}\left(\left(\gamma P + \frac{B_z^2}{\mu_o} \right) \frac{1}{r} \frac{\partial}{\partial r} (r\xi) \right)} \underset{[2]}{- \frac{\partial}{\partial r} \left(\frac{B_\theta^2}{\mu_o} \frac{\partial \xi}{\partial r} \right)} \underset{[3]}{+ \frac{\tilde{B}_\theta^2}{\mu_o r} \frac{\partial \xi}{\partial r}} \underset{[4]}{- \frac{\tilde{B}_\theta^2}{\mu_o r} \frac{\xi}{r}} \underset{[5]}{- \frac{2}{\mu_o r} \tilde{B}_\theta \left(\widetilde{B_\theta \frac{\partial \xi}{\partial r}} \right)} = \underset{[6]}{- \frac{\partial}{\partial r} \left(P + \frac{B_\theta^2 + B_z^2}{2\mu_o} \right)} \underset{[7]}{- \frac{\tilde{B}_\theta^2}{\mu_o r}} \tag{39}$$

in which the index (m+1) on ξ and m on all other variables has been dropped and the individual terms are numbered for future reference.

7. DIFFERENCE SCHEME

Equation (39) is now to be written in the tridiagonal form [6]

$$- A_j \xi_{j+1} + B_j \xi_j - C_j \xi_{j-1} = D_j \tag{40}$$

and solved by the standard Gauss elimination procedure:

$$\xi_j = E_j \xi_{j+1} + F_j \quad , \quad (j = 2,..\text{NINT}) \tag{41}$$

with

$$E_j = \frac{A_j}{B_j - C_j E_{j-1}} \quad , \quad (j = 2, \text{NINT}) \tag{42}$$

$$F_j = \frac{D_j + C_j F_{j-1}}{B_j - C_j E_{j-1}} \quad , \quad (j = 2, \text{NINT}) \quad . \tag{43}$$

The boundary conditions are $\xi_1 = 0$, (giving $E_1 = F_1 = 0$) and $\xi_{\text{NINT}+1} = 0$. All terms in (39) are well-behaved at the origin.

To simplify both the difference scheme and the Fortran we now use a relative notation centred on point j, indicated in Fig.2 (i.e. (++) for (j+1), (+) for $(j+\frac{1}{2})$ and so on), and also remove the factor $1/\mu_o$. Lower-case letters denote internal variables. We then obtain

$$A = a_1 \, r_{++} + a_2 + a_3 + a_5 \tag{44}$$

$$C = c_1 \, r_{--} + c_2 + c_3 + c_5 \tag{45}$$

$$B = (a_1 + c_1) r + (a_2 + c_2) + b_4 + (a_5 + c_5) \tag{46}$$

$$D = d_6 + d_7 \tag{47}$$

where the individual terms are defined in Tables 1 and 2 together with their Fortran equivalents. So far as possible a strict 1-1 correspondence has been maintained between the structure of the individual terms in (39) and their two alternative representations in Table 2, including the arrangement and order of the factors in the denominators. The form of coding shown here is not quite the optimum for minimizing the <u>execution time</u> during production runs, since an extra improvement could be gained by removing unnecessary division operations and also by taking advantage of algebraic relations between C_j and A_{j-1}. However experience with several versions of EQUIL suggests that full optimization makes the code hard to follow, and can lead to considerable extra effort and machine time in removing algebraic and coding errors and checking out the program. The present scheme is therefore suggested for 1D codes where maximum execution efficiency is not the only important criterion, although not necessarily for 2D or 3D codes. Table 3 indicates the mnemonics that have been used.

Construction of the coefficients (44)-(47) is mostly straightforward but two comments may be made:

<u>Terms 2 and 3</u>

$$\left(\frac{\partial\xi}{\partial r}\right)_j \to \frac{\xi_{j+1} - \xi_{j-1}}{r_{j+1} - r_{j-1}} \to \left(\xi_{++} - \xi_{--}\right)/2\,\Delta r \tag{48}$$

<u>Term 5</u>

$$\frac{2\widetilde{B}_\theta}{\mu_o r}\left(\widetilde{B_\theta \frac{\partial\xi}{\partial r}}\right)_j \to \frac{\widetilde{B}_{\theta j}}{\mu_o r_j}\left(B_{\theta,j+\frac{1}{2}} \frac{\xi_{j+1} - \xi_j}{\Delta r_{j+\frac{1}{2}}} + B_{\theta,j-\frac{1}{2}} \frac{\xi_j - \xi_{j-1}}{\Delta r_{j-\frac{1}{2}}}\right)$$

$$\to \hat{b}_\theta \left(\frac{\hat{b}_{\theta+}}{r\Delta r_+} . \xi_{++} - \left(\frac{\hat{b}_{\theta+}}{r\Delta r_+} - \frac{\hat{b}_{\theta-}}{r\Delta r_-}\right)\xi - \frac{\hat{b}_{\theta-}}{r\Delta r_-} . \xi_{--}\right) \quad . \tag{49}$$

The solution (41) is obtained in #3 of the code shown in Appendix A.

8. MESH ADJUSTMENT

After ξ^{m+1} has been obtained the radii $R_2 \ldots R_{NINT}$ are adjusted in ## 4 & 8 of the code using equation (26). From the arguments given in §3 it seems clear on physical grounds that the system must eventually reach a valid equilibrium from <u>any</u> starting-point however far it may have to move: however since ξ is only calculated from (39) in linear approximation two adjacent mesh boundaries might cross if ξ were too large at the beginning of the iteration process. To prevent this happening we use the algorithm

$$\underline{\text{if}} \quad \operatorname*{Min}_{j}\left(\frac{\Delta r_{+}^{m+1}}{\Delta r_{+}^{m}}\right) < \sigma \quad \underline{\text{then}}\ \xi_j = \alpha\xi_j \quad (j=2,..\text{NINT}) \tag{50}$$

where σ,α are control parameters such that

$$0 < \sigma,\alpha < 1 \quad . \tag{51}$$

This simply causes ξ to be scaled down uniformly if any of the mesh cells becomes too compressed in one iteration step or if its boundaries cross: it may be thought of as an increase of the damping discussed in §3. The algorithm (50) is applied recursively, so that ξ will if necessary continue to be scaled down until the required condition is met. Typically

$$\begin{aligned}\sigma &= \text{SIGMA} = 0.5\\ \alpha &= \text{ALPHA} = 0.5 \quad .\end{aligned} \tag{52}$$

Scaling-down usually only occurs right at the beginning of a run when the initial conditions do not correspond to pressure equilibrium, since the timestep Δt ought to be chosen small enough so that Stage A of each time-step only produces a small imbalance for Stage B to correct.

9. ADIABATIC UPDATING OF VARIABLES AND CONVERGENCE TEST

The variables P, B_θ, B_z are transferred to the new mesh in #5 using the exact adiabatic relations (28)-(30) rather than the linear approximations (34)-(36) which were employed in formulating (39). Convergence is then tested in #6 according to the criterion

$$\operatorname*{Max}_{j=2,\text{NINT}} \frac{|\xi_j|}{R_{j+1} - R_{j-1}} \le \varepsilon_c = \text{EPSC} \quad . \tag{53}$$

The value chosen for ε_c is not critical, since convergence is very rapid; however it should not be chosen so small that convergence is prevented by round-off errors. In the tests discussed in §11 we have used double-precision arithmetic and chosen $\varepsilon_c = 10^{-15}$ to obtain enough iterations.

10. CONVERGENCE ANALYSIS

We examine the theoretical rate of convergence for the simple case of plane geometry and pure plasma pressure. In this case (2) is just

$$\frac{\partial P}{\partial x} = 0 \tag{54}$$

where

$$P = P^m \left(\frac{\partial x^m}{\partial x}\right)^{\gamma} \quad . \tag{55}$$

Let

$$x - x^m \equiv \xi = \xi^{m+1} + \delta\xi \tag{56}$$

be the displacement needed to achieve exact equilibrium from the mesh positions defined by x^m; then

$$\frac{\partial x}{\partial x^m} = 1 + \frac{\partial \xi}{\partial x^m} \tag{57}$$

so that (54) together with (55) becomes

$$\frac{\partial}{\partial x}\left\{ P^m \left(1 + \frac{\partial \xi}{\partial x^m}\right)^{\gamma} \right\} = 0 \quad . \tag{58}$$

We can replace $\frac{\partial}{\partial x}$ by $\frac{\partial}{\partial x^m}$ in (58); then expanding in a power series gives

$$\frac{\partial P^m}{\partial x^m} - \frac{\partial}{\partial x^m}\left(\gamma P^m \frac{\partial \xi}{\partial x^m}\right) + \frac{\partial}{\partial x^m}\left\{ \frac{\gamma(\gamma+1)}{2} \; P_m \left(\frac{\partial \xi}{\partial x^m}\right)^2 \right\} = 0 \quad . \tag{59}$$

By subtracting the equation corresponding to (31) satisfied by ξ^{m+1}, namely

$$\frac{\partial P^m}{\partial x^m} - \frac{\partial}{\partial x^m}\left(\gamma P^m \frac{\partial \xi^{m+1}}{\partial x^m}\right) \tag{60}$$

we find for the residual error term $\delta\xi \simeq \xi^{m+2}$ the equation

$$\frac{\partial}{\partial x^m}\left(\gamma P^m \frac{\partial(\delta\xi)}{\partial x^m}\right) = \frac{\partial}{\partial x^m}\left\{ \frac{\gamma(\gamma+1)}{2} \; P_m \left(\frac{\partial \xi}{\partial x^m}\right)^2 \right\} \quad \ldots \tag{61}$$

which demonstrates the quadratic convergence when ξ is replaced by ξ^{m+1} on the right-hand side.

The numerical constant in this formula has been verified by choosing examples in which an external pressure is applied at a free boundary and $\partial/\partial x^m = 0$. Then (61) becomes simply

$$\xi^{m+2} = \frac{\gamma+1}{2} \frac{(\xi^{m+1})^2}{\Delta x^m} \quad --- \tag{62}$$

which was verified to 4 significant figures within 3-4 iterations.

The analysis can be generalized to the cylindrical problem with magnetic fields at the cost of some algebraic complexity, and quadratic convergence is still expected although the simple ratio $(\gamma+1)/2\Delta x_m$ of equation (62) is no longer found. Note that it is important to include <u>all</u> the linear terms in equation (31), since if any linear term of relative magnitude f is omitted the relaxation will be incomplete and we shall then obtain a much slower rate of convergence

$$|\xi^{m+1}| = f|\xi^m| \quad . \tag{63}$$

This was observed to happen on several occasions before the code was fully checked out, and the predicted quadratic convergence provides a very sensitive check on the validity of the numerical analysis and programming.

11. NUMERICAL TESTS IN CYLINDRICAL GEOMETRY

Two tests were performed using a mesh with 10 intervals which initially were uniformly spaced,

$$\left.\begin{aligned} R &= 10 \quad \text{(radius)} \\ \Delta r &= 1 \end{aligned}\right\} \quad . \tag{64}$$

The initial fields and pressure distributions were :

<u>Case 1</u>

$$\left.\begin{aligned} B_\theta &= r/R \\ B_z &= 1 \\ P &= 1 \end{aligned}\right\} \tag{65}$$

<u>Case 2</u>

$$\left.\begin{aligned} B_\theta &= 0 \\ B_z &= 1 \\ P &= r/R \end{aligned}\right\} \quad . \tag{66}$$

In each case the plasma moved inwards to achieve pressure equilibrium satisfying (2). The tests were carried out with the Fortran program shown in Appendix A on the General Electric Mark III International Information Network, using double precision for all real variables with a double-word length of 72 bits. The iterations were continued until the accuracy was limited by round-off errors. In the results shown in Figs.3 & 4 we define

$$x^m = \max_{j=(2,\mathrm{NINT})} \left| \xi_j^m / \Delta r_j^{m-1} \right| \tag{67}$$

together with the ratio

$$\lambda^m = x^m/(x^{m-1})^2 \tag{68}$$

which should be constant if the convergence is quadratic. From §10 λ is expected to be of order 1.

12. DOCUMENTATION OF FORTRAN PROGRAMS

Computational physics relies on the development of new algorithms and programs. It is desirable that these should be readily available like ordinary mathematics or theoretical physics [7] otherwise each worker in the field is restricted to what he or a small group of colleagues is able to do. To some extent algorithms can be described in mathematical notation and published in journals such as COMPUTER PHYSICS COMMUNICATIONS or JOURNAL OF COMPUTATIONAL PHYSICS. However it is still a considerable step to convert these mathematical algorithms into actual working programs.

High level languages such as Fortran and Algol 60, which were introduced more than 15 years ago, do go some way towards improving program intelligibility especially when the listings themselves are well structured and documented [4]. However it is obvious that there is still a significant difference between the intelligibility of even the best-written program and that of a good mathematical text, and that this difference depends to a large extent on restrictions both of character set and also of syntax.

A glance at a typical mathematical or physics textbook will confirm that a very wide character set is used to convey meaning: upper and lower case letters of different sizes, italic and heavy type, Greek and Gothic alphabets, superscripts and subscripts, special mathematical signs as well as careful spacing and layout together with tables and diagrams. It is not surprising that computer programs are so difficult to understand when they are confined to a standard range of upper case letters, numerical digits and a few extra symbols such as (+ - * / . , =).

Mathematical notation is easy to write down on paper or at the blackboard but it is much harder to type accurately, especially when even a minor error would cause a program or a compilation to fail. Therefore a relatively limited character set is actually more suitable for computer input than the very extended set used for mathematical publications would be. It would certainly be useful to have a few awkward restrictions removed, but it would be neither convenient nor economically practicable to have a full mathematical character set available on ordinary input devices or printers, even though it might be generated as graphical output under program control. Therefore one must be reconciled to a limited character set for program input and for printed listings for some time to come.

The syntax of ordinary mathematics or theoretical physics is also very flexible. Exact definitions are not required provided that the meaning is clear to the reader, and much of this meaning is conveyed by the text.

Accepted notation develops gradually as the result of many individual published papers and textbooks and is only rarely defined formally by committees. By contrast, not only does a computer language require a precisely-defined syntax, but it is also necessary to develop a compiler and to make this internationally available. Once a language has reached the stage of widespread acceptance it becomes very difficult to change, in spite of obvious weaknesses, simply because it would be necessary to reach agreement between a large number of manufacturers and then to update all the compilers and manuals. As a consequence of this Algol 60 has remained frozen for 15 years, and although many manufacturers have introduced their own individual and useful extensions to Fortran the only internationally available version is still the very limited Standard Fortran [8] subset first defined in 1964 [9]. Many new high-level languages have indeed been developed and compilers written and used on a limited scale, but even powerfully-supported languages such as PL/1 and Algol 68 have not yet reached the stage at which compilers are available for most types of machine.

It therefore seems unreasonable to expect, in the foreseeable future, to have a high-level programming language which is optimized for human communication and which at the same time is internationally available for use by computers.

The solution which is illustrated in this paper and proposed for general use is simply to reproduce the program in two parallel versions, side by side as in Appendix A. Version I is intended to be similar to ordinary mathematics and can use whatever notation is convenient including tables and diagrams where appropriate: even if the language of the program is Fortran there is no reason why Version I should not borrow notation from other well-known languages such as Algol 60 where this helps to make the notation clearer or the symbolism more concise. Version II is a straightforward listing in a regular programming language, in this case Fortran, although the same technique can be used for other high-level languages and also for assembler code.

There should be a 1-1 correspondence between the two versions in layout, numbering and notation, e.g.

$$\hat{b}_{z+}^{2} \longleftrightarrow \text{ZBZSHP} \tag{69}$$

as indicated in Tables 1-3. Each version then supports the other, since any lack of precision in Version I can be removed by looking at the listing of Version II and if necessary by running the program, whilst any lack of intelligibility in Version II should be removed by looking at Version I.

The most suitable style for Version I and the appropriate number of comments in Version II can best be decided in the light of experience. It is intended to publish a number of full-scale OLYMPUS Fortran [5] listings in this dual format as Culham Laboratory Reports, and also to submit some shorter Algol 60 and Fortran programs for publication in COMPUTER PHYSICS COMMUNICATIONS. We hope that others will feel encouraged to do likewise and that this type of publication will prove useful both as a means of communication between workers in the field and also as background material for teaching courses in computational physics and programming techniques. A further advantage may be that once a new form of notation has been introduced into Version I and become generally accepted, it could in due course find its way into the formal programming languages, thus enabling them to grow more freely than at present.

The Fortran Program EQUIL reproduced as Version II of Appendix A has been developed on-line using the General Electric Mark III International Information Service marketed in the UK by Honeywell. Mark III FOREGROUND Fortran IV [10] contains some extensions of Standard Fortran [8] which have been adopted here for convenience, notably the ability to have several statements on one line which makes the published listing more manageable, the use of lower-case letters (which are confined here to comments), and the simplified PRINT statement. The listing would normally be tidied up for publication by automatically removing the line numbers and transferring the Fortran statements to col. 7 or 10, but these features have been retained in this example to illustrate how the system is used in practice (Appendix B).

TABLE 1

Auxiliary Variables

Point	Index notation	Relative notation	Fortran
j+1	r_{j+1}	r_{++}	ZRPP
j+½	$r_{j+\frac{1}{2}} = \frac{1}{2}(r_{j+1} + r_j)$	r_+	ZRP
	$\Delta r_{j+\frac{1}{2}} = r_{j+1} - r_j$	Δr_+	ZDRP
	$P_{j+\frac{1}{2}}$	P_+	ZPP
	$B_{\theta, j+\frac{1}{2}}/\sqrt{\mu_o}$	$\hat{b}_{\theta+}$	ZBTHP
	$B^2_{\theta, j+\frac{1}{2}}/\mu_o$	$\hat{b}^2_{\theta+}$	ZBTSHP
	$B^2_{z, j+\frac{1}{2}}/\mu_o$	$\hat{b}^2_{z+}$	ZBZSHP
j	r_j	r	ZR
	Δr_j	Δr	ZDR
	$\tilde{B}_{\theta, j}/\sqrt{\mu_o}$	$\hat{b}_\theta = \frac{1}{2}(\hat{b}_{\theta+} + \hat{b}_{\theta-})$	ZBTH
	$\tilde{B}^2_{\theta, j}/\mu_o$	$\hat{b}^2_\theta$	ZBTSH
j-½	$r_{j-\frac{1}{2}} = \frac{1}{2}(r_j - r_{j-1})$	r_-	ZRM
	$\Delta r_{j-\frac{1}{2}} = r_j - r_{j-1}$	Δr_-	ZDRM
	$P_{j-\frac{1}{2}}$	P_-	ZPM
	$B_{\theta, j-\frac{1}{2}}/\sqrt{\mu_o}$	$\hat{b}_{\theta-}$	ZBTHM
	$B^2_{\theta, j-\frac{1}{2}}/\mu_o$	$\hat{b}^2_{\theta-}$	ZBTSHM
	$B^2_{z, j-\frac{1}{2}}/\mu_o$	$\hat{b}^2_{z-}$	ZBZSHM
j-1	r_{j-1}	r_{--}	ZRMM
	0.0		Z0
	Double 0.5		ZHALF
	precision 1.0		Z1
	2.0		Z2

TABLE 2

Components of A, B, C, D

Relative notation	Fortran
$a_1 = (\gamma p_+ + \hat{b}^2_{z+})/(\Delta r . r_+ \Delta r_+)$	ZA1 = (GAMMA*ZPP+ZBZSHP)/(ZDR*ZRP*ZDRP)
$a_2 = \hat{b}^2_{\theta+}/(\Delta r . \Delta r_+)$	ZA2 = ZBTSHP/(ZDR*ZDRP)
$a_3 = -\hat{b}^2_{\theta}/(r . 2\Delta r)$	ZA3 = -ZBTSH/(ZR*Z2*ZDR)
$a_5 = \hat{b}_\theta \; \hat{b}_{\theta+}/(r . \Delta r_+)$	ZA5 = ZBTH*ZBTHP/(ZR*ZDRP)
$c_1 = (\gamma p_- + \hat{b}^2_{z-})/(\Delta r . r_- \Delta r_-)$	ZC1 = (GAMMA*ZPM+ZBZSHM)/(ZDR*ZRM*ZDRM)
$c_2 = \hat{b}^2_{\theta-}/(\Delta r . \Delta r_-)$	ZC2 = ZBTSHM/(ZDR*ZDRM)
$c_3 = \hat{b}^2_{\theta}/(r . 2\Delta r)$	ZC3 = ZBTSH/(ZR*Z2*ZDR)
$c_5 = -\hat{b}_\theta \; \hat{b}_{\theta-}/(r . \Delta r_-)$	ZC5 = -ZBTH*ZBTHM/(ZR*ZDRP)
$b_4 = -\hat{b}^2_{\theta}/(r . r)$	ZB4 = -ZBTSH/(ZR*ZR)
$d_6 = -\left\{(p_+ + \tfrac{1}{2}(\hat{b}^2_{\theta+} + \hat{b}^2_{z+})) - (p_- + \tfrac{1}{2}(\hat{b}^2_{\theta-} + \hat{b}^2_{z-}))\right\}/\Delta r$	ZD6 = -((ZPP+ZHALF*(ZBTSHP+ZBZSHP))-(ZPM+ZHALF*(ZBTSHM+ZBZSHM)))/ZDR
$d_7 = -\hat{b}^2_{\theta}/r$	ZD7 = -ZBTSH/ZR

TABLE 3

Fortran Mnemonics

Letter	Meaning
B	magnetic field
D	Δ
H	$\wedge$ (hat) (μ_o removed)
M	-
MM	--
P	pressure
P	+
PP	++
R	r
S	square
S	* (new mesh)
T	θ-component
Z	z-component
Z	internal real variable

REFERENCES

[1] D D Clayton, 'Principles of Stellar Evolution and Nucleosynthesis', McGraw-Hill, New York (1968).

[2] J P Christiansen, J Long and K V Roberts (to be submitted to Computer Physics Communications).

[3] K V Roberts and G G Lister (to be submitted to Computer Physics Communications).

[4] K V Roberts, Comp.Phys.Comm. 7, 237 (1974).

[5] J P Christiansen and K V Roberts, Comp.Phys.Comm. 7, 245 (1974).

[6] R D Richtmeyer and K W Morton, 'Difference Methods for Initial Value Problems', Interscience, New York (1967).

[7] K V Roberts, Comp.Phys.Comm. 1, 1 (1969).

[8] 'Standard Fortran Programming Manual', Computer Standards Series, National Computing Centre Ltd., Manchester, England (1970).

[9] Comm.Assoc.Compt.Mach. 7, 591 (1964).

[10] Honeywell Mark III Foreground Fortran IV Reference Manual, Order No. 3102.01E (March 1974).

[11] Honeywell Mark III Foreground Command System Reference Manual, Order No.3501.01 Rev.1 (Sept.1973).

[12] Honeywell Mark III Foreground Editing Commands Reference Manual, Order No. 3400.01F (Jan.1974).

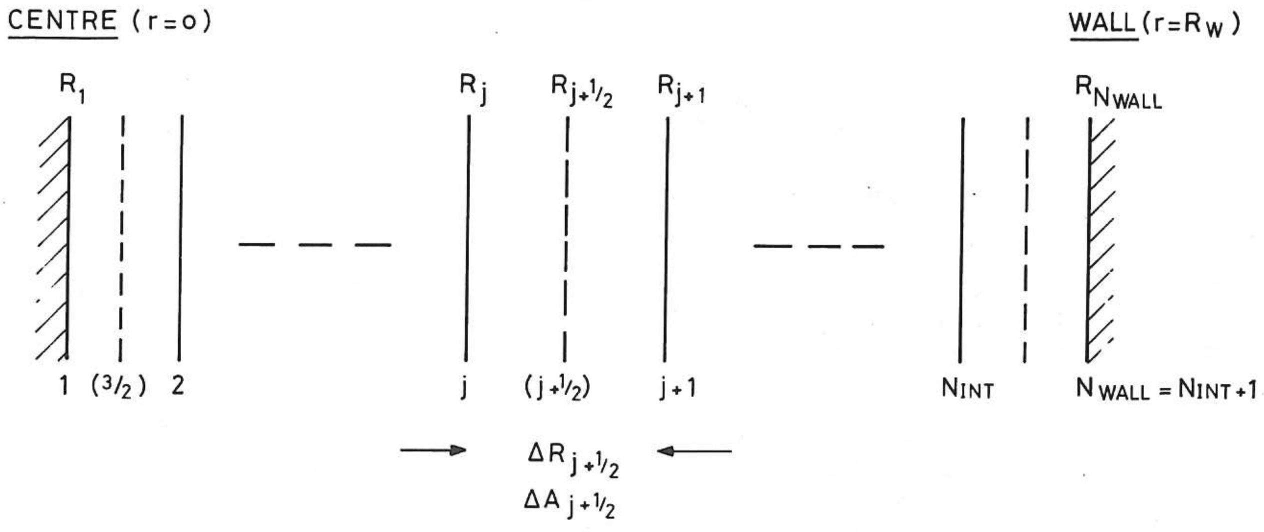

Fig.1 Lagrangian Mesh

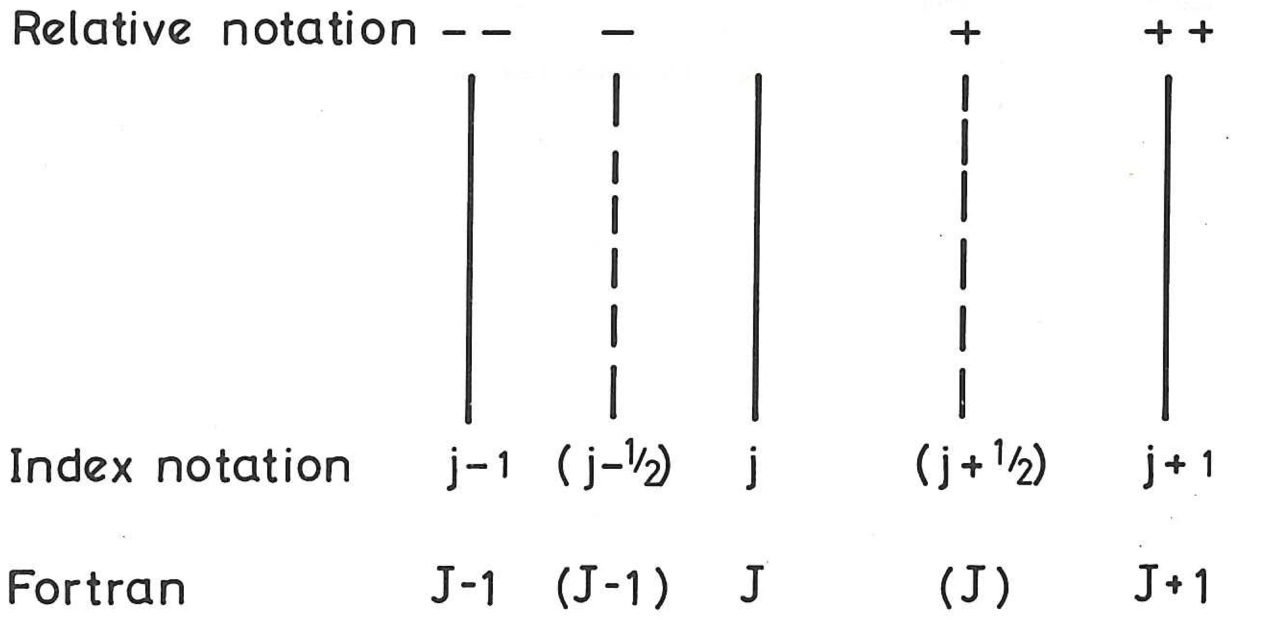
Relative notation -- - + ++
Index notation j-1 (j-1/2) j (j+1/2) j+1
Fortran J-1 (J-1) J (J) J+1

Fig.2 Relative Notation

```
EQUIL          10:33          11/17/75


 TEST CASE 1. BT=R/RWALL, BZ=P=1

R  =  0.      1.00   2.00   3.00   4.00   5.00   6.00   7.00   8.00   9.00  10.00
P  =  1.00    1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   0.
BT =  0.05    0.15   0.25   0.35   0.45   0.55   0.65   0.75   0.85   0.95   0.
BZ =  1.00    1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   0.

               M              X                              LAMBDA
               1  2.49189073458500257YD-01
               2  2.22386598666722271 5D-02   3.58138179713080 2934D-01
               3  6.13436127536666 7726D-04   1.240372487421357348D+00
               4  7.65862066060684 9092D-07   2.035222333705141106D+00
               5  2.20372539630785 6833D-12   3.757131179281661400D+00
               6  3.27134686767662 9734D-19   6.736148341137969652D+04

  ITERATION CONVERGES

R  =  0.      0.88   1.76   2.66   3.58   4.53   5.52   6.55   7.64   8.78  10.00
P  =  1.17    1.16   1.15   1.13   1.10   1.06   1.03   0.98   0.94   0.90   0.
BT =  0.06    0.17   0.28   0.38   0.47   0.56   0.63   0.69   0.74   0.78   0.
BZ =  1.30    1.28   1.26   1.22   1.17   1.11   1.04   0.97   0.90   0.83   0.

PROGRAM STOP AT 92200

USED    10.11 UNITS
```

Fig.3 Output from Test Case 1

```
EQUIL          10:45          11/17/75


 TEST CASE 2. BT=0, BZ=1, P=R/RWALL

R  =  0.      1.00   2.00   3.00   4.00   5.00   6.00   7.00   8.00   9.00  10.00
P  =  0.05    0.15   0.25   0.35   0.45   0.55   0.65   0.75   0.85   0.95  0.
BT =  0.      0.     0.     0.     0.     0.     0.     0.     0.     0.    0.
BZ =  1.00    1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00  0.

                M             X                             LAMBDA
                1   3.40839322394542 4743D-01
                2   7.18976012120428555 8D-02   6.18892207311031 7505D-01
                3   1.64453422329263187 9D-02   3.18136950647683495 7D+00
                4   1.34791981366724350 0D-03   4.98400220512851822 6D+00
                5   9.24705017450885723 1D-06   5.08949977646958305 1D+00
                6   4.30620990913697839 0D-10   5.03603532461323498 6D+00
                7   9.72376441368614476 8D-19   5.24377485015009435 6D+00

  ITERATION CONVERGES

R  =  0.      0.82   1.67   2.55   3.48   4.44   5.45   6.51   7.61   8.78  10.00
P  =  0.06    0.18   0.30   0.40   0.50   0.58   0.66   0.73   0.79   0.85  0.
BT =  0.      0.     0.     0.     0.     0.     0.     0.     0.     0.    0.
BZ =  1.50    1.42   1.34   1.26   1.18   1.10   1.03   0.96   0.89   0.83  0.

PROGRAM STOP AT 92200

USED     10.23 UNITS
```

Fig.4 Output from Test Case 2

```
OLD P7C2S1


READY
COM



P7C2S1        12:02        11/17/75


REAL SCALAR DT NEVER ASSIGNED A VALUE, LINE 9100.
REAL SCALAR C NOT USED; LINE 21100.
INTEGER SCALAR LZ NEVER ASSIGNED A VALUE, LINE 21100.
INTEGER SCALAR LY NEVER ASSIGNED A VALUE, LINE 21100.
INTEGER SCALAR NZP2 NEVER ASSIGNED A VALUE, LINE 22050.
INTEGER SCALAR MZ NOT USED; LINE 22100.
INTEGER SCALAR NYP3 NEVER ASSIGNED A VALUE, LINE 23050.
INTEGER SCALAR MY NOT USED; LINE 23100.
INTEGER SCALAR NXP4 NEVER ASSIGNED A VALUE, LINE 24100.
MAIN PROGRAM REQUIRED FOR EXECUTION.



          Ready


USED      1.14 UNITS
```

Fig.5 Mark III Compiler Diagnostics. These diagnostics were produced by an on-line compilation in which the COMMON block was deliberately omitted.

Version I. Mathematical Notation

Program EQUIL tests the adiabatic relaxation of a Lagrangian mesh to pressure equilibrium. An iterative method is used. At each iteration step m+1 the mesh displacement ξ^{m+1} is determined by solving the second-order linear inhomogeneous equation (39) which has the symbolic form

$$L\xi = f, \qquad (\xi_1 = \xi_{NWALL} = 0; \quad NWALL = NINT+1) \tag{A.1}$$

the individual terms being numbered sequentially:

$$L = L_1 + L_2 + L_3 + L_4 + L_5, \quad f = f_6 + f_7 \tag{A.2}$$

as indicated in §6. Equation (A.1) is differenced and written as

$$-A_j\ \xi_{j+1} + B_j\ \xi_j - C_j\ \xi_{j-1} \ = \ D_j \qquad \text{(A3)}$$

for solution by the standard tridiagonal method [6]. Fig.2 indicates the relative notation employed.

The program is initialized in #1. In #1.5 we set up Test Case 1: $P = B_z = 1$, $B_\theta = {}^r/R_W$ and print the initial conditions in #1.6. This case is readily altered by the user. The individual terms 1-7 are computed in #3.2 and the coefficient vectors A,B,C,D in # 3.3, the displacement ξ^{m+1} being obtained in #3.5. In #4 we construct the new mesh R^*, if necessary scaling down the displacement as explained in §8. The variables P, B_θ, B_z are transferred to the new mesh in #5 and the mesh is reset in #8. The convergence test (53) is imposed in ##6 & 9. For convenience we set $\mu_o = 1$.

Program:

Dimension A, B, C, D, E, F, B_θ, B_z, P, R, R^*, ξ [11]

Use double precision

#1. Initialize

1.1 Constants

Set double precision constants; $\mu_o = 1$; MAXDIM = 11

1.2 Clear arrays

$A_j = B_j = C_j = D_j = E_j = F_j = R^*_j = \xi_j = 0$, (j = 1, MAXDIM)

1.3 Parameters

$\alpha = \sigma = \frac{1}{2}$; $\gamma = {}^5/3$; $\epsilon_c = 10^{-15}$; $R_W = 10$

NINT = NITMAX = 10

1.4 Mesh

NWALL = NINT + 1; $\Delta r = R_W/NINT$

$R_j = (j-1)\Delta r$, (j = 1,NWALL)

1.5 Dependent variables

Test Case 1: $B_\theta = {}^r/R_W$; $B_z = P = 1$

1.6 Initial output

print R,P,B_θ,B_z

print heading for #7

Version II. Fortran

```
110 DIMENSION A(11),B(11),C(11),D(11),E(11),F(11)
120 DIMENSION BT(11),BZ(11),XP(11)
130 DIMENSION R(11),RS(11),XSI(11)
1000 IMPLICIT DOUBLE PRECISION (A-H,O-Z)
9999C
10000C--------------------
10001C*1. Initialize
11000C
11001C* 1.1 Constants
11050 Z0=0.0D0; ZHALF=0.5D0; Z1=1.0D0; Z2=2.0D0
11100 Z3=3.0D0; Z5=5.0D0; Z10=1.0D1; ZEPS=1.0D-15
11150 FCMU0=Z1; MAXDIM=11
11200 SQRMU0=DSQRT(FCMU0)
12000C
12001C* 1.2 Clear arrays
12050 DO 121 J=1,MAXDIM
12100 A(J)=Z0; B(J)=Z0; C(J)=Z0; D(J)=Z0; E(J)=Z0; F(J)=Z0
12150 RS(J)=Z0; XSI(J)=Z0
12200 121 CONTINUE
13000C
13001C* 1.3 Parameters
13050 ALPHA=ZHALF; SIGMA=ZHALF; GAMMA=Z3/Z5
13100 EPSC=ZEPS; RWALL=Z10
13150 NINT=10; NITMAX=10
14000C
14001C* 1.4 Mesh
14050 NWALL=NINT+1; ZDR=RWALL/FLOAT(NINT)
14100 DO 141 J=1,NWALL
14150 R(J)=ZDR*DFLOAT(J-1)
14200 141 CONTINUE
15000C
15001C* 1.5 Dependent variables
15050 Z=DFLOAT(NINT)
15100 DO 151 J=1,NINT
15150 ZR=ZHALF*(R(J)+R(J+1))
15200 BT(J)=ZR/RWALL; BZ(J)=Z1; XP(J)=Z1
15250 151 CONTINUE
16000C
16001C* 1.6 Initial output
16025 PRINT," TEST CASE 1. BT=R/RWALL, BZ=P=1"
16035 PRINT 9400
16050 PRINT 9401,R; PRINT 9402,XP; PRINT 9403,BT; PRINT 9404,BZ
16100 PRINT 9400
16150 PRINT 9405
19999C
```

2. <u>Begin iteration loop</u>

<u>do</u> 911 m = 1, NITMAX

~~#~~3. <u>Calculate ξ</u>

3.1 Auxiliary variables on relative mesh

<u>do</u> 331 j = 2, NINT

radii: $r_{++} = R_{j+1}$; $r = R_j$; $r_{--} = R_{j-1}$; $r_+ = \frac{1}{2}(r_{++}+r)$; $r_- = \frac{1}{2}(r+r_{--})$

differences: $\Delta r_+ = r_{++} - r$; $\Delta r_- = r - r_{--}$

pressure and magnetic fields:

$p_\pm = P_{j\pm\frac{1}{2}}$; $\underline{\hat{b}}_\pm = \underline{B}_{j\pm}/\sqrt{\mu_o}$; $\underline{\hat{b}} = \frac{1}{2}(\underline{\hat{b}}_+ + \underline{\hat{b}}_-)$;

3.2 Individual numbered coefficients

$a_1 = (\gamma p_+ + \hat{b}^2_{z+})/(\Delta r.r_+\Delta r_+)$; $\quad a_2 = \hat{b}^2_{\theta+}/(\Delta r.\Delta r_+)$

$a_3 = - \hat{b}^2_\theta/(r.2\Delta r)$; $\quad a_5 = \hat{b}_\theta\,\hat{b}_{\theta+}/(r.\Delta r_+)$

$c_1 = (\gamma p_- + \hat{b}^2_{z-})/(\Delta r.r_-\Delta r_-)$; $\quad c_2 = \hat{b}^2_{\theta-}/(\Delta r.\Delta r_-)$

$c_3 = \hat{b}^2_\theta/(r.2\Delta r)$; $\quad c_5 = -\hat{b}_\theta\,\hat{b}_{\theta-}/(r.\Delta r_-)$

$b_4 = - \hat{b}^2_\theta/r^2$

$d_6 = - \left\{(p_+ + \frac{1}{2}(\hat{b}^2_{\theta+}+\hat{b}^2_{z+}) - (p_- + \frac{1}{2}(\hat{b}^2_{\theta-}+\hat{b}^2_{z-}))\right\}/\Delta r$; $\quad d_7 = - \hat{b}^2_\theta/r$

3.3 Form coefficient vectors A,B,C,D

$A_j = a_1\,r_{++} + a_2 + a_3 + a_5$

$B_j = (a_1 + c_1)r + (a_2 + c_2) + b_4 + (a_5 + c_5)$

$C_j = c_1 r_{--} + c_2 + c_3 + c_5$

$D_j = d_6 + d_7$

331 <u>continue</u>

3.4 Construct vectors E,F

$E_1 = F_1 = 0$

$E_j = A_j/(B_j - C_j E_{j-1})$

$F_j = (D_j + C_j\,F_{j-1})/(B_j - C_j\,E_{j-1})$ (j = 2,NINT)

3.5 Solve for new ξ

$\xi_1 = \xi_{NWALL} = 0$

$\xi_j = E_j\,\xi_{j+1} + F_j$, (j = 2,NINT)

```
20000C--------------------
20001C*2. Begin iteration loop
20002C
20050 DO 911 JIT=1,NITMAX
29999C
30000C--------------------
30001C*3. Calculate xsi
31000C
31001C* 3.1 Auxiliary variables on relative mesh
31050 DO 331 J=2,NINT
31200C Radii
31250 ZRPP=R(J+1); ZR=R(J); ZRMM=R(J-1)
31300 ZRP=ZHALF*(ZR+ZRPP); ZRM=ZHALF*(ZR+ZRMM)
31325C
31350C Differences
31400 ZDRP=ZRPP-ZR; ZDR=ZRP-ZRM; ZDRM=ZR-ZRMM
31425C
31450C Pressure and magnetic fields
31500 ZPP=XP(J); ZPM=XP(J-1)
31550 ZBTHP=BT(J)/SQRMUO; ZBTHM=BT(J-1)/SQRMUO
31570 ZBTSHP=ZBTHP*ZBTHP; ZBTSHM=ZBTHM*ZBTHM
31600 ZBZSHP=BZ(J)*BZ(J)/FCMUO; ZBZSHM=BZ(J-1)*BZ(J-1)/FCMUO
31650 ZBTH=ZHALF*(ZBTHP+ZBTHM); ZBTSH=ZBTH*ZBTH
32000C
32001C* 3.2 Individual numbered coefficients
32050 ZA1=(GAMMA*ZPP+ZBZSHP)/(ZDR*ZRP*ZDRP)
32100 ZA2=ZBTSHP/(ZDR*ZDRP)
32150 ZA3=-ZBTSH/(ZR*Z2*ZDR)
32200 ZA5=ZBTH*ZBTHP/(ZR*ZDRP)
32250C
32300 ZC1=(GAMMA*ZPM+ZBZSHM)/(ZDR*ZRM*ZDRM)
32350 ZC2=ZBTSHM/(ZDR*ZDRM)
32400 ZC3=ZBTSH/(ZR*Z2*ZDR)
32450 ZC5=-ZBTH*ZBTHM/(ZR*ZDRM)
32500C
32550 ZB4=-ZBTSH/(ZR*ZR)
32600C
32650 ZD6=-((ZPP+ZHALF*(ZBTSHP+ZBZSHP))-(ZPM+ZHALF*(ZBTSHM+ZBZSHM)))/ZDR
32700 ZD7=-ZBTSH/ZR
33000C
33001C* 3.3 Form coefficient vectors a,b,c,d
33050 A(J)=ZA1*ZRPP+ZA2+ZA3+ZA5
33100 B(J)=(ZA1+ZC1)*ZR+(ZA2+ZC2)+ZB4+(ZA5+ZC5)
33150 C(J)=ZC1*ZRMM+ZC2+ZC3+ZC5
33200 D(J)=ZD6+ZD7
33250C
33300 331 CONTINUE
34000C
34001C* 3.4 Construct vectors e,f
34050 E(1)=Z0; F(1)=Z0
34100 DO 341 J=2,NINT
34150 Z=Z1/(B(J)-C(J)*E(J-1))
34200 E(J)=A(J)*Z
34250 F(J)=(D(J)+C(J)*F(J-1))*Z
34300 341 CONTINUE
35000C
35001C* 3.5 Solve for new xsi
35050 XSI(1)=Z0; XSI(NWALL)=Z0
35100 DO 351 J=2,NINT
35150 I=NINT+2-J
35200 XSI(I)=E(I)*XSI(I+1)+F(I)
35250 351 CONTINUE
39999C
```

#4. Construct new mesh

4.1 Trial displacement

Reset factor: $f = 1$

End points: $R_1^* = R_1$; $R_{NWALL}^* = R_{NWALL}$

411 continue

Displace mesh: $R_j^* = R_j + f\,\xi_j$, $(j = 2,NINT)$

4.2 Check for excessive compression or overlap

go to 430 if $(R_{j+1}^* - R_{j-1}^*)/(R_{j+1} - R_{j-1}) < \sigma$, (any $j = 1,NINT$)

mesh OK: go to 500

4.3 Reduce displacement factor

$f = \alpha f$

print warning message and values of j, Δr^*, Δr

go to 411

#5. Adiabatically transform variables to new mesh

500 continue

$$\left.\begin{array}{ll} \text{cell centres:} & r^* = \tfrac{1}{2}(R_{j+1}^* + R_j^*);\quad r = \tfrac{1}{2}(R_{j+1} + R_j) \\ \text{intervals:} & \Delta r^* = R_{j+1}^* - R_j^*;\quad \Delta r = R_{j+1} - R_j \\ \text{compression factors:} & f_r = r/r^*;\quad f_\Delta = \Delta r/\Delta r^*;\quad f_A = f_r f_\Delta \\ \text{transform variables:} & P_j = P_j\, f_A^\gamma;\quad B_{\theta j} = B_{\theta j}\, f_\Delta;\quad B_{zj} = B_{zj}\, f_A \end{array}\right\}\ (j = 2,NINT)$$

#6. Form convergence parameter

if $(m \neq 1)$ $\delta_\ell = \delta$

$$\delta = \text{Max}\left\{ |\xi_j| / (R_{j+1}^* - R_{j-1}) \right\} \qquad (j = 2,NINT)$$

if $(m \neq 1)$ $\lambda = \delta/\delta_\ell^2$

```
40000C--------------------
40001C*4. Construct new mesh
41000C
41001C* 4.1 Trial displacement
41050 ZFAC=Z1
41100 RS(1)=R(1); RS(NWALL)=R(NWALL)
41150 411 CONTINUE
41200C
41250 DO 412 J=2,NINT
41300 RS(J)=R(J)+ZFAC*XSI(J)
41350 412 CONTINUE
42000C
42001C* 4.2 Check for excessive compression or overlap
42050 DO 422 J=1,NINT
42100 I=J
42150 ZDRS=RS(J+1)-RS(J); ZDR=R(J+1)-R(J)
42200 IF(ZDRS/ZDR.LT.SIGMA) GO TO 430
42250 422 CONTINUE
42300C Mesh ok
42350 GO TO 500
43000C
43001C* 4.3 Reduce displacement factor
43025 430 CONTINUE
43050 ZFAC=ZFAC*ALPHA
43100 PRINT," ZFAC REDUCED"
43150 PRINT,"   J =",I,"  DR* =",ZDRS,"  DR =",ZDR
43200 GO TO 411
49999C
50000C--------------------
50001C*5. Adiabatically transform variables to new mesh
50002C
50025 500 CONTINUE
50050 DO 511 J=1,NINT
50075C
50100C Cell centres, (*2)
50150 ZRS=RS(J+1)+RS(J); ZR=R(J+1)+R(J)
50200C Intervals
50250 ZDRS=RS(J+1)-RS(J); ZDR=R(J+1)-R(J)
50300C Compression factors
50350 ZFACR=ZR/ZRS; ZFACD=ZDR/ZDRS; ZFACA=ZFACR*ZFACD
50400C
50450C Transform variables
50500 XP(J)=XP(J)*ZFACA**GAMMA
50550 BT(J)=BT(J)*ZFACD
50600 BZ(J)=BZ(J)*ZFACA
50650 511 CONTINUE
59999C
60000C--------------------
60001C*6. Form convergence parameter
60002C
60025 IF(JIT.NE.1) DLAST=DELTA
60050 DELTA=ZO
60100 DO 601 J=2,NINT
60150 DELTA=DMAX1(DELTA,DABS(XSI(J))/(RS(J+1)-RS(J-1)))
60200 601 CONTINUE
60250 IF(JIT.NE.1) DLAMDA=DELTA/(DLAST**2)
69999C
```

7. Test output during iteration loop

```
        if $(m = 1)$ print $m, \delta$
        if $(m \neq 1)$ print $m, \delta, \lambda$
```

8. Reset mesh

```
        $R_j = R_j^*$,   $(j = 1,NWALL)$
```

9. End of iteration loop

```
    9.1 Test for convergence
        if $\delta < \varepsilon_c$ go to 920
        next iteration
        911 continue
        convergence failure:  go to 930
    9.2 Iterations converged
        920 continue
        print message;  print $R, P, B_\theta, B_z$
        stop
    9.3 Convergence failure
        930 continue
        print message
        stop
```

```
70000C-------------------
70001C*7. Test output during iteration loop
70002C
70050 IF(JIT.EQ.1) PRINT,JIT,DELTA
70100 IF(JIT.NE.1) PRINT,JIT,DELTA,DLAMDA
79999C
80000C------------------
80001C*8. Reset mesh
80002C
80050 DO 801 J=1,NWALL
80100 R(J)=RS(J)
80150 801 CONTINUE
89999C
90000C-------------------
90001C*9. End of iteration loop
91000C
91001C* 9.1 Test for convergence
91050 IF(DELTA.LT.EPSC) GO TO 920
91075C Next iteration
91100 911 CONTINUE
91125C Convergence failure
91150 GO TO 930
92000C
92001C* 9.2 Iterations converged
92025 920 CONTINUE
92035 PRINT 9400
92050 PRINT,"  ITERATION CONVERGES"
92100 PRINT 9400
92150 PRINT 9401,R; PRINT 9402,XP; PRINT 9403,BT; PRINT 9404,BZ
92200 STOP
93000C
93001C* 9.3 Convergence failure
93050 930 CONTINUE
93075 PRINT 9400
93100 PRINT,"  NO CONVERGENCE"
93150 STOP
94000C
94001C* 9.4 Format statements
94050 9400 FORMAT(1X)
94100 9401 FORMAT(1X,"R  =",11F6.2)
94150 9402 FORMAT(1X,"P  =",11F6.2)
94200 9403 FORMAT(1X,"BT =",11F6.2)
94250 9404 FORMAT(1X,"BZ =",11F6.2)
94300 9405 FORMAT(12X,"M",13X,"X",23X,"LAMBDA")
99999 END
```

ON-LINE PROGRAMMING TECHNIQUES

It has been found very convenient in practice to develop and test the critical sections of new programs on-line using time-sharing services available via the dialled telephone network. Such services usually have a much simpler Job Control or Command Language than ordinary in-house computer systems, they provide an immediate turnround and concise diagnostics for compilations and short test runs, often on a 7-day/24-hour basis, and they can be used from anywhere (e.g. from home). The General Electric Mark III International Information Service discussed in this Appendix has the further advantage that it can be used for collaboration between computational physicists in different countries.

On-line test runs are normally carried out with limited mesh sizes and a small number of timesteps in order to limit output as in the present example. Once the program or subroutine has been checked it can subsequently be transferred to the local in-house system for more extensive tests and for production runs, usually either via a paper tape or magnetic tape cassette or by hand repunching, although transfer via cards, 7/9 track magnetic tape or direct computer connection is also feasible.

Several techniques have been worked out to improve convenience and to reduce costs. Where paper tapes are mentioned magnetic cassettes could also be used. Mark III publications [10,11,12] should be consulted for further details. Corresponding techniques often apply to other time-sharing services in somewhat different forms.

Paper tape

When more than just a few Fortran statements are to be read into a file these should normally be prepunched off-line, e.g. using a teletype equipped with paper tape reader and punch. This reduces telephone and connect charges by a considerable factor. Each line ends with the characters

CR (carriage return), LF (line feed), Rubout (B.1)

and a short leader and trailer consisting of a number of rubout characters should also be punched at the beginning and end of the tape. The command

TAPE (B.2)

informs the system that paper tape input is required.

Fortran format

Each Fortran statement or comment begins with a line number of 1 - 5 digits and can therefore be distinguished from a system command which begins with a letter: there is no specific 'command mode' or 'edit mode'. Similarly, a comment has a 'C' immediately following the line number with no intervening blank space and can therefore be distinguished from a statement. A continuation line has an ampersand & immediately following the line number. These conventions which are illustrated in Appendix A have the advantage of limiting the number of key-punch depressions needed and also the time taken to generate a listing. The amount of paper output can be reduced and the format improved by punching several statements on one line separated by a semi-colon.

Editing

Although a powerful context editor is available [12] the simplified editing facilities provided by the input routine are usually sufficient for most work:

(a) Lines can be typed in any order: they are then automatically rearranged numerically.

(b) A line is replaced by retyping it with the same line number.

(c) A line is removed by typing its line number, followed immediately by CR (or the sequence (B.1) when using paper tape).

This allows a program or subroutine to be developed or corrected in any convenient order; e.g. the comment headings can be typed first and the statements filled in later.

OLYMPUS conventions

The OLYMPUS card conventions [4] have been modified for on-line work. In particular the line numbers are correlated with the section numbering and the headings are shortened by removing most of the blanks. The conventions used on Mark III are illustrated in Appendix A.

1. Sections. As an example, #6 begins with

```
60000C --- (20 dashes) ---                    (B.3)
```

and terminates with

```
69999C                                        (B.4)
```

Line 60001 has:

(a) C* in columns 6 & 7

(b) 6. in columns 7 & 8

(c) Blank in column 10

(d) Heading beginning in column 11.

Line 60002 is a 'blank comment' similar to (B.4). Subsequent lines in this section (until #6.1) start with '60'. They would normally be punched in units of 50 starting from 60050 to allow for subsequent insertions. Where #6.1

immediately follows the heading of #6, line 60002 is omitted.

2. Subsections. As an example, #3.4 begins with 34000C, followed by line 34001 which has

(a) C* in columns 6 & 7

(b) Blank in column 8

(c) 3.4 in columns 9 - 11

(d) Blank in column 12

(e) Heading beginning in column 13.

Subsequent lines in this section start with '34' and are normally punched in units of 50 starting from 34050.

3. Preservation of line numbers. Many on-line systems allow lines to be automatically renumbered but this facility is not used for OLYMPUS files because it would disturb the conventions 1&2. An advantage of not renumbering program and subroutine files is that old listings remain useful for some time if they are corrected by hand; this reduces costs and saves time by limiting the amount of output needed.

4. Insertion of COMMON. The OLYMPUS COMMON blocks [4,5] labelled [C1.1] - [C6.9] occupy the 4-digit line numbers 1100 - 6999 and are contained in a single file. They can then be automatically inserted into a subroutine using the EDIT WEAVE facility [12], e.g.

```
(i)    EDI WEA P7C2S1; P7COM  )
(ii)   SAVE TEMP              )
(iii)  COM                    )        (B.5)
(iv)   SAVE STEPON.           )
```

Here file P7C2S1 contains the source file of subroutine ⟨2.1⟩ STEPON of program P7, while file P7COM contains all the COMMON blocks. Command (i) merges them together to form a 'current' file which is then saved in a new temporary file TEMP, compiled, and the object module saved as STEPON. It can then be called in automatically by the loader. However, COMMON is not used in program EQUIL shown in Appendix A.

Compilation and Testing

A program or subroutine can be compiled using the COM command. Fig. 5 illustrates some of the error diagnostics that are available: here a short subroutine has been compiled with COMMON omitted. The use of line-numbers combined with the OLYMPUS numbering system makes it very easy to locate and correct errors.

A source-language or binary program can be run using the RUN command,

and again the diagnostics enable errors to be located using the line number.

Diagnostic output

The PRINT statement whose application is illustrated in Appendix A already supplies some of the facilities normally provided by the OLYMPUS diagnostic routines [5]. Other OLYMPUS facilities could be extended for on-line use but have not yet been introduced. In developing a major new program it is often convenient to test the logic with the physics edited out and replaced by diagnostic statements indicating the flow of control, indexing etc. This is currently being carried out for a 3DMHD code.