132
7-24-79

Lb. 2892

MASTER

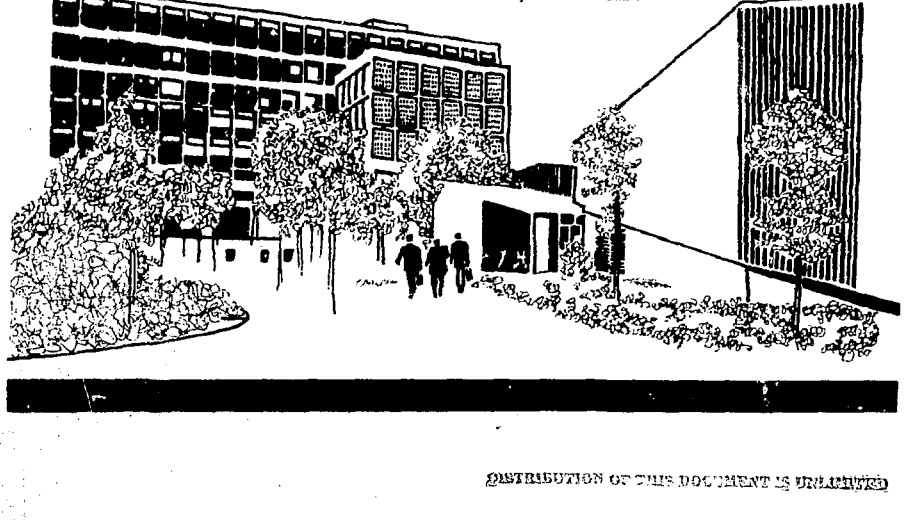
UCRL-52492

THE ANIMAL CODE

I. R. Lindemuth

February 28, 1979

Work performed under the auspices of the U.S. Department of Energy by the UCLLL under contract number W-7405-ENG-48.



Distribution Category
UC-34

LAWRENCE LIVERMORE LABORATORY
University of California Livermore, California 94550

UCRL-52492

THE ANIMAL CODE

I. R. Lindemuth*

MS. date: February 28, 1979

*Los Alamos Scientific Laboratory,
Los Alamos, New Mexico 87544.

CONTENTS

THE ANIMAL CODE

ABSTRACT

This report describes ANIMAL, a two-dimensional Eulerian magnetohydrodynamic computer code. ANIMAL's physical model also appears. Formulated are temporal and spatial finite-difference equations in a manner that facilitates implementation of the algorithm. Outlined are the functions of the algorithm's FORTRAN subroutines and variables.

1. INTRODUCTION

This report describes the physical model and numerical methods in the computer code ANIMAL—*A New* (Alternating Direction) *Implicit Magnetohydrodynamic ALgorithm*. In addition, a description of the important FORTRAN variables used in the code is included and the method of memory allocation is described. This report is intended to introduce the code features to the new user or interested computational physicist. This report is by no means complete: it does not give every detail of the physical model necessary for implementation, nor does it give every finite-difference technique, nor is it a card-by-card code description. This report does not attempt to justify the physical model or the finite-difference techniques, nor does it relate much of the history of the code. Rather, this report tells the reader what ANIMAL is now, take-it-or-leave-it. Hopefully, this report does give a useful overview of the code as it exists at this writing. The various references cited are useful supplements to this report and the reader should also familiarize himself with these documents.

ANIMAL numerically solves a set of nonlinear, time-dependent, two-dimensional magnetohydrodynamic (MHD) partial differential equations. Basically speaking, ANIMAL is intended to solve the model equations in any coordinate system. To date, the Cartesian, cylindrical r-z, cylindrical r-ϕ, spherical, and toroidal coordinate systems have been implemented. The present version of ANIMAL is limited to a rectangular domain in whichever coordinate system is used. However, a version nearing the completion of development allows the construction of a finite-difference grid that consists of a number of rectangular domains connected on at least one side with another domain, and the coordinate system of each domain can be different; with such a capability, more complex physical systems can be modeled.

ANIMAL is a generalized Eulerian code. Either the finite-difference grid remains fixed in space or the grid can move in one of the coordinate directions while retaining the orthogonality of the grid. In a Eulerian code, mass—as well as momentum, energy, and magnetic flux—is allowed to move from one cell to other cells, and the finite-difference grid remains orthogonal. Historically, nearly all major multidimensional MHD codes have been of the Eulerian (fixed-grid) or generalized Eulerian (moving-orthogonal-grid) type. One alternative is the Lagrangian approach where the finite-difference grid is fixed to the plasma, so that the mass within a cell remains fixed; momentum, energy, and magnetic flux do, however, move from cell to cell. It is doubtful that a Lagrangian code could handle the strongly two-dimensional, shearing motion observed in the calculations performed on ANIMAL, since, as is well known, Lagrangian codes tend to lose accuracy and stability when the mesh becomes strongly distorted. Intermediate to the generalized Eulerian and Lagrangian approaches is the generalized mesh method in which the mesh moves in a Lagrangian manner until sufficient distortion is encountered, at which time, mass is allowed to flow from cell to cell. The nearly Lagrangian motion of the generalized mesh method retains to a large degree the Lagrangian advantages of interface resolution and accurate convective transport in the absence of shear. However, when motion becomes strongly two-dimensional, the mesh in the generalized mesh method will become considerably distorted, and it is unclear to this author just how well physical processes such as strong diffusion can be handled numerically on a nonorthogonal mesh. Without a doubt, Eulerian codes have been used to simulate situations where the multi-dimensional motion that occurs is stronger than any computations reported to date using either Lagrangian or generalized mesh methods. In addition, the physical models that have been incorporated in the Eulerian codes are more complete.

The physical model used in ANIMAL includes thermal conduction, resistive diffusion, radiation, and ionization in addition to the inclusion of the Lorentz $\bar{J} \times \bar{B}$ force in a fluid description of a plasma. The charged-particle transport coefficients are "classical" and the atomic processes are based on local thermodynamic equilibrium. The plasma equations are coupled self-consistently to electrical circuit equations.

The major limitation of ANIMAL at present is that only the velocity components in the plane of the calculation and the magnetic field normal to the plane are considered. In addition, ions and electrons are assumed to be in thermal equilibrium, a limitation that can be eliminated in a quite straightforward manner. A simple Ohm's law including only resistive diffusion is normally used, although the transverse thermoelectric effect is an option included in the code.

Alternating-direction implicit (ADI) finite-difference methods are used in ANIMAL. The ADI method of temporal differencing allows the use of a timestep larger than could be used with less sophisticated, explicit finite-difference techniques. As Roberts and Potter[1] have discussed, the timestep restrictions encountered with an explicit method because of convective and diffusive transport can become very severe in magnetohydrodynamics. In principle, the ADI method allows for a timestep dependent on time-rates-of-change of quantities, rather than on Courant–Friedrichs–Lewy[2] conditions; thus, for example, when simulating a quiescent gas with a very large sound speed, the timestep can be much larger than the time required for a sound wave to travel across one zone. In addition, ADI permits the formal accuracy of the finite-difference methods to be rigorously second-order accurate with respect to the timestep. And, finally, ADI permits the inclusion of all physical processes and the inclusion of both dimensions, simultaneously, without the need to resort to fractional steps, i.e., "splitting," and the resultant inaccuracies.

In ANIMAL, the ADI method of temporal differencing has been combined with spatial-difference equations in such a manner that energy conservation depends only on timestep size and is independent of spatial-zone size, even though a nonconservation form of energy equation is used.[3] For isolated systems, exact energy conservation to within 0.01% is generally expected.

The treatment of boundaries in ANIMAL is unique.[4] ANIMAL incorporates previously unconsidered aspects of plasma/wall interaction. Situations such as wall contact, separation, and return are relevant in ANIMAL.

ANIMAL is intended to be a flexible and versatile tool. The numerous built-in options for geometries, boundary conditions, initial conditions, physics (and even for difference methods) permit classes of problems never run on the code previously to be attempted. For this flexibility the user must pay a price. Relatively little of the code is "hardwired" and the user must experiment with things like zone size, timestep control parameters, artificial-viscosity parameters, and difference technique to prevent pathologies and to ensure accuracy. The code is not "idiot-proof," and not all combinations of input data will work. The code must be considered on a "user-beware" basis, and the user should not take for granted that everything will perform as advertised for every conceivable case, since not every conceivable case has been checked out on the code.

During execution, ANIMAL produces a minimal amount of readily readable output. Normally, this output is sufficient only to verify that a problem has been correctly generated and has encountered no pathologies during the course of the computation. Only through the use of an extensive post-processor, briefly described in this document, is extensive problem analysis possible. At this writing, the post-processor provides "snapshot" computer graphics for approximately 55 spatially dependent quantities, e.g., density, and approximately 90 time-dependent quantities, e.g., average density.

Published applications of ANIMAL include a study of the interaction of a hot, magnetized plasma with a cold wall,[5] a study of z-pinch plasma under liner implosion conditions,[6] and a study of the MHD behavior of thermonuclear fuel in an advanced relativistic electron-beam target using the fuel preheat-and-magnetothermoinsulation principle.[7] ANIMAL was developed primarily to calculate the Krakatoa toroidal-pinch experiment.[8] An important accomplishment of ANIMAL is its ability to calculate the preionization phase of this experiment.[9] ANIMAL has also been used to study sausage instabilities of a plasma column, the run down stage of Lawrence Livermore Laboratory's (LLL's) plasma focus, Rayleigh-Taylor instabilities of a plasma decelerated by a magnetic field in connection with LLL's Baseball-II laser-target plasma production,[10] and studies of laser/plasma experiments.[11] ANIMAL calculations have suggested a possible diagnostic for use in liner implosion studies.[12]

Since ANIMAL is primarily a "pinch" code intended to calculate very dynamic plasma behavior, its capabilities should be considered in context with other pinch-like calculations appearing in the literature. The pioneering one-dimensional computer code of Hain *et al.*[13] was applied to the study of theta and screw pinches, and various versions of their code have been used similarly at laboratories throughout the world. Duchs,[14] Freeman and Lane,[15] Schneider,[16] and Lindemuth and Killeen[17] computed various two-dimensional features of theta pinches. The code of Lindemuth and Killeen was a predecessor to the current ANIMAL code. Other pinch-like two-dimensional simulations were of the plasma-focus discharge by Potter,[18] a plasmoid interacting with a solenoidal magnetic field by Freeman,[19] the trapping and thermalization of a laser-produced plasma by Lindemuth and Killeen,[17] and the simulations of plasma flow in channels by Brushlinsky.[20] More recently, Hofmann[21] has simulated the dynamics of a belt pinch and Lui and Chu[22] have simulated the dynamics of a toroidal screw pinch in which all three components of the magnetic field are important. Brackbill[23] has done a three-dimensional ideal MHD computation of kink instabilities.

2. PHYSICAL MODEL

An MHD model based on local thermodynamic equilibrium is incorporated into ANIMAL. The basic model equations are

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \bar{v}) = 0, \tag{1}$$

$$\frac{\partial (\rho \bar{v})}{\partial t} + \nabla \cdot (\rho \bar{v}\, \bar{v}) + \nabla p + \frac{1}{\mu_0} \bar{B} \times (\nabla \times \bar{B}) = 0, \tag{2}$$

$$\frac{\partial (\rho \epsilon)}{\partial t} + \nabla \cdot (\rho \bar{v} \epsilon) + p \nabla \cdot \bar{v} - \nabla \cdot \left[K \nabla T + \frac{T \bar{\beta}}{\mu_0} \times (\nabla \times \bar{B}) \right]$$

$$- (\nabla \times \bar{B}) \left[\frac{\eta}{\mu_0^2} (\nabla \times \bar{B}) - \frac{\bar{\beta}}{\mu_0} \times \nabla T \right] + \dot{\epsilon}_{RAD} = 0, \tag{3}$$

and

$$\frac{\partial \bar{B}}{\partial t} - \nabla \times (\bar{v} \times \bar{B}) + \nabla \times \left[\frac{\eta}{\mu_0} (\nabla \times \bar{B}) - \bar{\beta} \times \nabla T \right] = 0. \tag{4}$$

In Eqs. (1) to (4), ρ is the density, $\bar{v}$ is the fluid velocity, p the pressure, $\bar{B}$ the magnetic field, ϵ the specific internal energy of the fluid, K the thermal conductivity, T the temperature in joules, η the electrical resistivity, $\dot{\epsilon}_{RAD}$ a radiative energy loss, and μ_0 the free-space permeability; mks units are used throughout. Equation (1) is the continuity equation. Equation (2) is the equation of motion; the fourth term is the Lorentz force, $\bar{J} \times \bar{B}$, where the current density $\bar{J}$ has been eliminated through the use of Ampere's law with the usual neglect of displacement current. Equation (3) is the internal energy equation; the fourth term is the divergence of the heat-flow vector and the first part of the next-to-last term represents ohmic heating, ηJ^2. Equation (4) is Faraday's law, based on a simple Ohm's law, $\bar{E} = -\bar{v} \times \bar{B} + \eta \bar{J} - \bar{\beta} \times \nabla T$, where the vector $\bar{\beta}$ is the "transverse" thermoelectric coefficient multiplied by a unit vector in the $\bar{B}$ direction.

For implementation into ANIMAL, the vector-model Eqs. (1) to (4) must be written out into component form. The geometric versatility of ANIMAL is attained by writing the component equations in their general orthogonal, curvilinear coordinate form and making a coordinate transformation from the usual (x_1, x_3, t) two-dimensional coordinate system to a "fixed" coordinate system (ξ_1, ξ_3, t). It is required that the transformation must satisfy $\frac{\partial x_1}{\partial \xi_3} = \frac{\partial x_3}{\partial \xi_1} = 0$. Ignoring the azimuthal component of Eq. (4), the component forms of Eqs. (1) to (4) then become

$$\frac{\partial}{\partial t} (h_1 h_2 h_3 x_{11} x_{33} \rho) + \frac{\partial}{\partial \xi_1} \left(x_{33} h_2 h_3 \rho \left[v_1 - v_1^G \right] \right) + \frac{\partial}{\partial \xi_3} \left(x_{11} h_1 h_2 \rho \left[v_3 - v_3^G \right] \right) = 0, \tag{5}$$

$$\frac{\partial}{\partial t} (h_1 h_2 h_3 x_{11} x_{33} \rho v_1) + \frac{\partial}{\partial \xi_1} \left(x_{33} h_2 h_3 \rho v_1 \left[v_1 - v_1^G \right] \right)$$

$$+ \frac{\partial}{\partial \xi_3} \left(x_{11} h_1 h_2 \rho v_1 \left[v_3 - v_3^G \right] \right) + h_2 \rho v_3 (v_1 h_{13} x_{11} - v_3 h_3, x_{33})$$

$$+ h_2 h_3 x_{33} \frac{\partial p}{\partial \xi_1} + \frac{1}{\mu_0} h_3 x_{33} B_2 \frac{\partial}{\partial \xi_1} (h_2 B_2) = 0, \tag{6}$$

$$\frac{\partial}{\partial t}\left(h_1 h_2 h_3 x_{11} x_{33}\,\rho v_3\right)+\frac{\partial}{\partial \xi_1}\left(x_{33} h_2 h_3\,\rho v_3\left[v_1-v_1^G\right]\right)$$
$$+\frac{\partial}{\partial \xi_3}\left(x_{11} h_1 h_2\,\rho v_3\left[v_3-v_3^G\right]\right)+h_2\,\rho v_1\left(v_3 h_{31} x_{33}-v_1 h_{13} x_{11}\right)$$
$$+h_1 h_2 x_{11}\frac{\partial p}{\partial \xi_3}+\frac{1}{\mu_0} h_1 x_{11} B_2\frac{\partial}{\partial \xi_3}\left(h_2 B_2\right)=0, \tag{7}$$

$$\frac{\partial}{\partial t}\left(h_1 h_2 h_3 x_{11} x_{33}\,\rho\epsilon\right)+\frac{\partial}{\partial \xi_1}\left(x_{33} h_2 h_3\,\rho\epsilon\left[v_1-v_1^G\right]\right)$$
$$+\frac{\partial}{\partial \xi_3}\left(x_{11} h_1 h_2\,\rho\epsilon\left[v_3-v_3^G\right]\right)+p\frac{\partial}{\partial \xi_1}\left(h_2 h_3 x_{33} v_1\right)$$
$$+p\frac{\partial}{\partial \xi_3}\left(h_1 h_2 x_{11} v_3\right)-\frac{\partial}{\partial \xi_1}\left[\frac{h_2 h_3 x_{33}}{h_1 x_{11}}\left(K\frac{\partial T}{\partial \xi_1}+\frac{\beta T|B_2|}{\mu_0 h_2 B_2}\frac{\partial}{\partial \xi_1}\left[h_2 B_2\right]\right)\right]$$
$$-\frac{\partial}{\partial \xi_3}\left[\frac{h_1 h_2 x_{11}}{h_3 x_{33}}\left(K\frac{\partial T}{\partial \xi_3}+\frac{\beta T|B_2|}{\mu_0 h_2 B_2}\frac{\partial}{\partial \xi_3}\left[h_2 B_2\right]\right)\right]$$
$$-\frac{h_3 x_{33}}{h_1 h_2 x_{11}\mu_0}\frac{\partial}{\partial \xi_1}\left(h_2 B_2\right)\left[\frac{\eta}{\mu_0}\frac{\partial}{\partial \xi_1}\left(h_2 B_2\right)+\frac{h_2\beta|B_2|}{B_2}\frac{\partial T}{\partial \xi_1}\right]$$
$$-\frac{h_1 x_{11}}{h_3 h_2 x_{33}\mu_0}\frac{\partial}{\partial \xi_3}\left(h_2 B_2\right)\left[\frac{\eta}{\mu_0}\frac{\partial}{\partial \xi_3}\left(h_2 B_2\right)+\frac{h_2\beta|B_2|}{B_2}\frac{\partial T}{\partial \xi_3}\right]$$
$$+\dot{\epsilon}_{RAD}=0, \tag{8}$$

and

$$\frac{\partial}{\partial t}\left(h_1 h_3 x_{11} x_{33} B_2\right)+\frac{\partial}{\partial \xi_1}\left[h_3 x_{33} B_2\left(v_1-v_1^G\right)\right]+\frac{\partial}{\partial \xi_3}\left[h_1 x_{11} B_2\left(v_3-v_3^G\right)\right]$$
$$-\frac{\partial}{\partial \xi_3}\left[\frac{h_1 x_{11}}{h_3 x_{33}}\left(\frac{\eta}{\mu_0 h_2}\frac{\partial}{\partial \xi_3}\left(h_2 B_2\right)+\beta\frac{|B_2|}{B_2}\frac{\partial T}{\partial \xi_3}\right)\right]$$
$$-\frac{\partial}{\partial \xi_1}\left[\frac{h_3 x_{33}}{h_1 x_{11}}\left(\frac{\eta}{\mu_0 h_2}\frac{\partial}{\partial \xi_1}\left(h_2 B_2\right)+\beta\frac{|B_2|}{B_2}\frac{\partial T}{\partial \xi_1}\right)\right]=0. \tag{9}$$

In Eqs. (5) to (9), the definitions $x_{11}=\frac{\partial x_1}{\partial \xi_1}$, $x_{33}=\frac{\partial x_3}{\partial \xi_3}$, $h_{13}=\frac{\partial h_1}{\partial \xi_3}$, $h_{31}=\frac{\partial h_3}{\partial \xi_1}$, $v_1^G=h_1\frac{\partial x_1}{\partial t}$, and $v_3^G=h_3\frac{\partial x_3}{\partial t}$ have been introduced for convenience. The notations v_1 and v_3 refer to the velocity components in the ξ_1 and ξ_3 direction, respectively, and B_2 is the component of magnetic field normal to the ξ_1–ξ_3 plane.

The component Eqs. (5) to (9) are written essentially in the form in which they are differenced. It is very important to note that only first and second spatial derivatives of the dependent variables ρ, v_1, v_3, ϵ, and B_2 are present and that mixed derivatives of the form $\frac{\partial^2}{\partial \xi_1 \partial \xi_3}$ do not appear. Any additional Braginskii[24] physics effects not included in Eqs. (1) to (4) generally involve the mixed second derivative.

To be complete, the model specified in Eqs. (1) to (4) must be supplemented by equations of state relating the pressure, temperature, thermal conductivity, resistivity, and radiative energy loss to the density and specific internal energy, and, for the thermal conductivity, to the magnetic field. It is here that an approximation to local thermodynamic equilibrium is used. The ionization fraction f_i for deuterium at a temperature T is computed from

$$C_1 f_i^2 + f_i = 1, \tag{10}$$

where

$$C_1 = \frac{\rho}{m_i} \left(\frac{h^2}{2\pi m_e T} \right)^{3/2} \exp\left(\frac{I_1}{T} \right) . \tag{11}$$

Equation (10) is the Saha equation; the density dependence of the ionization potential I_1 is taken into account. A similar expression is used for the dissociation fraction f_d. The fraction dissociated and fraction ionized given the total number of particles, so the pressure, assuming an ideal gas, is

$$p = \frac{1}{2} \rho T(1 + f_d + 2f_i)/m_i \quad , \tag{12}$$

where T is in joules and all other quantities are in standard mks units. Similarly, the specific internal energy is

$$\epsilon = \frac{1}{2m_i} \left[T(2.5 + \frac{1}{2} f_d + 3f_i) + 2f_i \epsilon_i + \epsilon_d f_d \right] \quad , \tag{13}$$

where ϵ_i and ϵ_d are the ionization and dissociation energies, respectively, and where the 2.5 takes into account the rotational and vibrational degrees of freedom of the deuterium molecule.

The thermal conductivity used is the sum of the electron and ion thermal conductivities perpendicular to a magnetic field as given by Braginskii.[24] Neutral-particle thermal conductivity, formulated by Stevens,[25] can be used as an option. The effect of neutral particles on charged-particle thermal conduction has not been taken into account.

The enhanced radiation due to line emission can be an important energy-loss mechanism. The radiative loss rate $\dot{\epsilon}_{RAD}$ of Eq. (3) is

$$\dot{\epsilon}_{RAD} = C_2 \, G(\rho,T) \, \rho^2 \, T^{1/2} , \tag{14}$$

where C_2 is a constant and $G(\rho,T)$ is a density- and temperature-dependent "total emission factor," which is a sum of contributions from Griem's rates for bremsstrahlung, free–bound radiation, and bound–bound radiation.[26] When $G(\rho,T)$ is equal to unity, $\dot{\epsilon}_{RAD}$ equals the bremsstrahlung rate of Spitzer.[27]

The electrical resistivity, which represents the collisional loss of momentum and gain of thermal energy by electrons, is due to collisions with both ions and neutrals. The resistivity is taken to be

$$\eta = \eta_B + \eta_N , \tag{15}$$

where η_B is the Braginskii[25] perpendicular resistivity due to ion–electron collisions and where η_N, the resistivity due to electron–neutral particle collisions, is given by

$$\eta_N = C_3 \, T^{1/2} \, (1 - f_i)/f_i , \tag{16}$$

where C_3 is a constant.

The transverse thermoelectric coefficient β is also taken from Braginskii.[24]

3. TEMPORAL DIFFERENCING—GENERAL FORMALISM

The component Eqs. (5) to (9) can be written in the form

$$\frac{\partial \bar{T}(\bar{U})}{\partial T} + \bar{X}\left(\bar{U}, \frac{\partial}{\partial \xi_1}\right) + \bar{Y}\left(\bar{U}, \frac{\partial}{\partial \xi_3}\right) = 0, \tag{17}$$

where $\bar{X}$, $\bar{Y}$, $\bar{T}$, and $\bar{U}$ are five component vectors, and

$$\bar{U} = (\rho, v_1, v_3, e, B_2). \tag{18}$$

Only derivatives with respect to one orthogonal coordinate ξ_1 appear in $\bar{X}$ and only derivatives with respect to the second coordinate ξ_3 appear in $\bar{Y}$. The two alternately used finite-difference equations in ANIMAL can be considered to have the form

$$\frac{\bar{T}\left(\bar{U}^{n+1}_{j,k}\right) - \bar{T}\left(\bar{U}^{n}_{j,k}\right)}{t^{n+1} - t^{n}} + \tilde{X}\left[\left(\bar{U}^{n+1}_{j-1,k}\right), \left(\bar{U}^{n+1}_{j,k}\right), \left(\bar{U}^{n+1}_{j+1,k}\right)\right] + \tilde{Y}\left[\left(\bar{U}^{n}_{j,k-1}\right), \left(\bar{U}^{n}_{j,k}\right), \left(\bar{U}^{n}_{j,k+1}\right)\right] = 0 \tag{19}$$

$$\frac{\bar{T}\left(\bar{U}^{n+2}_{j,k}\right) - \bar{T}\left(\bar{U}^{n+1}_{j,k}\right)}{t^{n+2} - t^{n+1}} + \tilde{X}\left[\left(\bar{U}^{n+1}_{j-1,k}\right), \left(\bar{U}^{n+1}_{j,k}\right), \left(\bar{U}^{n+1}_{j+1,k}\right)\right] + \tilde{Y}\left[\left(\bar{U}^{n+2}_{j,k-1}\right), \left(\bar{U}^{n+2}_{j,k}\right), \left(\bar{U}^{n+2}_{j,k+1}\right)\right] = 0, \tag{20}$$

where $\bar{U}^{n}_{j,k}$ designates values at time t^n and spatial coordinates $(\xi_1)_j$, $(\xi_3)_k$. In Eqs. (19) and (20), $\tilde{X}$ and $\tilde{Y}$ are spatial finite-difference approximations to $\bar{X}$ and $\bar{Y}$, respectively. Equations (19) and (20) show the standard ADI coupling between unknown quantities. In Eq. (19), which is used to advance the calculations from t^n to t^{n+1}, the unknowns are the values $\bar{U}^{n+1}_{j,k}$ along a line of constant k; the quantities at $k+1$ and $k-1$ are known quantities since they have the superscript n. In Eq. (20), which is used to advance the calculations from t^{n+1} to t^{n+2}, the unknowns are the values $U^{n+2}_{j,k}$ along a line of constant j; the quantities at $j+1$ and $j-1$ are known quantities since they have the superscript $n+1$. Equations (19) and (20) are in general nonlinear functions of the unknown quantities and therefore cannot be solved directly. To solve Eqs. (19) and (20), ANIMAL uses essentially a Newton-Raphson method as given by Pennington,[28] among others. Application of the Newton-Raphson method to Eq. (19) gives an equation of the form

$$(\bar{\bar{A}}_1)^{n+1,\ell}_{j,k} \cdot \bar{U}^{n+1,\ell+1}_{j,k} + (\bar{\bar{B}}_1)^{n+1,\ell}_{j,k} \cdot \bar{U}^{n+1,\ell+1}_{j+1,k} + (\bar{\bar{C}}_1)^{n+1,\ell}_{j,k} \cdot \bar{U}^{n+1,\ell+1}_{j-1,k} = (\bar{V}_1)^{n+1,\ell}_{j,k}, \tag{21}$$

where the additional superscripts ℓ and $\ell+1$ indicate the iteration number and where $\bar{\bar{A}}$, $\bar{\bar{B}}$, and $\bar{\bar{C}}$ are matrices.

The calculations reported by Lindemuth and Killeen[17] can be considered as using Eq. (21) for $\ell = 0$ only. As shown by Lindemuth and Killeen, Eq. (21) for $\ell = 0$ gives an approximation formally second-order accurate with respect to the timestep $\Delta t = t^{n+1} - t^n$. Repeated application of Eq. (21) until convergence is achieved does not increase the formal accuracy of the solution, and Eqs. (19) and (20) are, when used together, still of second-order accuracy with respect to the timestep. However, the basic reason for the success of ADI is that the errors introduced on one timestep are cancelled on the following timestep. This apparently requires the two approximations to the $\bar{X}$ of Eq. (17) to have the same values, as indicated in Eqs. (19) and (20). If Eq. (21) is not iterated to some sort of convergence, the net effect is to use a somewhat different value for $\bar{X}$ in (20) than is used in (19). Experience during the code development process has shown that failure to iterate introduces unwanted, nonphysical effects that affect the calculations unless the timestep is reduced considerably.

Note that Eq. (19) or (20), considered alone, is an approximation to the complete physical system, Eq. (17). ANIMAL, as its predecessor,[17] does not use fractional timestep or splitting procedures, whereby one physical process—or one dimension—is treated as if the others were not present. Experience during code development has shown instances where the coupling between physical processes or dimensions was sufficiently strong that a fractional timestep method would have required a considerably reduced timestep to maintain accuracy. For example, situations have been observed where the energy increase due to Ohmic heating was balanced by the heat loss due to thermal conduction, so that no net change occurred, and yet either process by itself would have led to a drastic change in the net energy.

Equation (21) is appropriate only when $1 < j < J$ and $1 < k < K$. ANIMAL casts boundary conditions in the form

$$\bar{U}^{n+1,\varrho+1}_{1,k} = (\bar{\bar{E}}_1)^{n+1,\varrho}_{1,k} \cdot \bar{U}^{n+1,\varrho+1}_{2,k} + (\bar{\bar{H}}_1)^{n+1,\varrho}_{1,k} \cdot \bar{U}^{n+1,\varrho+1}_{3,k} + (\bar{F}_1)^{n+1,\varrho}_{1,k} , \tag{22}$$

$$\bar{U}^{n+1,\varrho+1}_{J,k} = (\bar{\bar{E}}_1)^{n+1,\varrho}_{J,k} \cdot \bar{U}^{n+1,\varrho+1}_{J-1,k} + (\bar{\bar{H}}_1)^{n+1,\varrho}_{J,k} \cdot \bar{U}^{n+1,\varrho+1}_{J-2,k} + (\bar{F}_1)^{n+1,\varrho}_{J,k} , \tag{23}$$

$$\bar{U}^{n+1,\varrho+1}_{j,1} = (\bar{\bar{E}}_3)^{n+1,\varrho}_{j,1} \cdot \bar{U}^{n+1,\varrho+1}_{j,2} + (\bar{\bar{H}}_3)^{n+1,\varrho}_{j,1} \cdot \bar{U}^{n+1,\varrho+1}_{j,3} + (\bar{F}_3)^{n+1,\varrho}_{j,1} , \tag{24}$$

and

$$\bar{U}^{n+1,\varrho+1}_{j,K} = (\bar{\bar{E}}_3)^{n+1,\varrho}_{j,K} \cdot \bar{U}^{n+1,\varrho+1}_{j,K-1} + (\bar{\bar{H}}_3)^{n+1,\varrho}_{j,K} \cdot \bar{U}^{n+1,\varrho+1}_{j,K-2} + (\bar{F}_3)^{n+1,\varrho}_{j,K} . \tag{25}$$

Equations (21) to (23) form a set of linear, simultaneous, "tridiagonal" algebraic equations in the unknown quantities $\bar{U}^{n+1,\varrho+1}_{j,k}$ for $1 \leq j \leq J$ along a line of constant k, $1 < k < K$. The method of solution involves calculating $\bar{\bar{E}}$'s and $\bar{F}$'s such that

$$\bar{U}^{n+1,\varrho+1}_{j,k} = \bar{\bar{E}}^{n+1,\varrho}_{j,k} \cdot \bar{U}^{n+1,\varrho+1}_{j+1,k} + \bar{F}^{n+1,\varrho}_{j,k} . \tag{26}$$

Substitution of Eq. (26) into Eq. (21) leads to the result that

$$\bar{\bar{E}}^{n+1,\varrho}_{2,k} = -\left[(\bar{\bar{A}}_1)^{n+1,\varrho}_{2,k} + (\bar{\bar{C}}_1)^{n+1,\varrho}_{2,k} \cdot (\bar{\bar{E}}_1)^{n+1,\varrho}_{1,k} \right]^{-1} \cdot \left[(\bar{\bar{B}}_1)^{n+1,\varrho}_{2,k} + (\bar{\bar{C}}_1)^{n+1,\varrho}_{2,k} \cdot (\bar{\bar{H}}_1)^{n+1,\varrho}_{1,k} \right] , \tag{27}$$

$$\bar{F}^{n+1,\varrho}_{2,k} = \left[(\bar{\bar{A}}_1)^{n+1,\varrho}_{2,k} + (\bar{\bar{C}}_1)^{n+1,\varrho}_{2,k} \cdot (\bar{\bar{E}}_1)^{n+1,\varrho}_{1,k} \right]^{-1} \cdot \left[(\bar{V}_1)^{n+1,\varrho}_{2,k} - (\bar{\bar{C}}_1)^{n+1,\varrho}_{2,k} \cdot (\bar{F}_1)^{n+1,\varrho}_{1,k} \right] , \tag{28}$$

$$\bar{\bar{E}}^{n+1,\varrho}_{j,k} = -\left[(\bar{\bar{A}}_1)^{n+1,\varrho}_{j,k} + (\bar{\bar{C}}_1)^{n+1,\varrho}_{j,k} \cdot \bar{\bar{E}}^{n+1,\varrho}_{j-1,k} \right]^{-1} \cdot (\bar{\bar{B}})^{n+1,\varrho}_{j,k} , \quad 2 < j < J , \tag{29}$$

and

$$\bar{F}^{n+1,\varrho}_{j,k} = \left[(\bar{\bar{A}}_1)^{n+1,\varrho}_{j,k} + (\bar{\bar{C}}_1)^{n+1,\varrho}_{j,k} \cdot \bar{\bar{E}}^{n+1,\varrho}_{j-1,k} \right]^{-1} \cdot \left[(\bar{V}_1)^{n+1,\varrho}_{j,k} - (\bar{\bar{C}}_1)^{n+1,\varrho}_{j,k} \cdot \bar{F}^{n+1,\varrho}_{j-1,k} \right] , \quad 2 < j < J . \tag{30}$$

Using Eqs. (29) and (30) for J−1 and J−2 in Eq. (26) and substituting into Eq. (23) leads to

$$\bar{U}^{n+1,\varrho+1}_{J,k} = \left\{ \bar{\bar{I}} - \left[(\bar{\bar{E}}_1)^{n+1,\varrho}_{J,k} + (\bar{\bar{H}}_1)^{n+1,\varrho}_{J,k} \cdot \bar{\bar{E}}^{n+1,\varrho}_{J-2,k} \right] \cdot \bar{\bar{E}}^{n+1,\varrho}_{J-1,k} \right\}^{-1} \cdot$$

$$\left\{ (\bar{F}_1)^{n+1,\varrho}_{J,k} + \left[(\bar{\bar{H}}_1)^{n+1,\varrho}_{J,k} \cdot \bar{\bar{E}}^{n+1,\varrho}_{J-2,k} + (\bar{E}_1)^{n+1,\varrho}_{J,k} \right] \cdot \bar{F}^{n+1,\varrho}_{J-1,k} \right.$$

$$\left. + (\bar{\bar{H}}_1)^{n+1,\varrho}_{J,k} \cdot \bar{F}^{n+1,\varrho}_{J-2,k} \right\} . \qquad (31)$$

Thus the solution procedure along a line of constant k is to set and store the boundary condition Eq. (22), calculate $\bar{\bar{A}}$, $\bar{\bar{B}}$, $\bar{\bar{C}}$, and $\bar{V}$ of Eq. (21) for j = 2, calculate and store $\bar{\bar{E}}_2$ and $\bar{F}_2$ from Eqs. (27) and (28), repetitively calculate $\bar{\bar{A}}$, $\bar{\bar{B}}$, $\bar{\bar{C}}$, and $\bar{V}$ of Eq. (21), and calculate and store the $\bar{\bar{E}}$'s and $\bar{F}$'s of Eqs. (29) and (30) for $2 < j < J$. $\bar{U}_j$ is then calculated from Eq. (31) and all other $\bar{U}_j$'s are calculated in decreasing order of j from Eq. (26). Note that it is not necessary to store $\bar{\bar{A}}$, $\bar{\bar{B}}$, $\bar{\bar{C}}$, and $\bar{V}$ for each j as long as the boundary conditions have the form given in Eqs. (22) and (23). Also, note that each k line is computed independently.

Equations (21) to (31), as a method of solving Eq. (19), give the basic ANIMAL algorithm for advancing the calculations from time t^n to t^{n+1}. The algorithm begins by setting $\bar{U}^{n+1,0}_{j,k} = \bar{U}^{n}_{j,k}$. Then for k = 2 (or K−1) Eqs. (21) to (31) are applied repetitively until $\left| \left(\bar{U}^{n+1,\varrho+1}_{j,k} - \bar{U}^{n+1,\varrho}_{j,k} \right) / \bar{U}^{n+1,\varrho}_{j,k} \right| < \delta$, where δ is typically 5×10^{-4}. Then each successive k is advanced similarly. When all k such that $2 \leq k \leq K-1$ have been advanced, the boundary conditions Eqs. (24) and (25) are used to set values at k = 1 and k = K.

To advance the calculation from time t^{n+1} to t^{n+2}, the Newton-Raphson method is applied to Eq. (20). Boundary conditions have the same form as in Eqs. (22) to (24), with the superscript n+1 replaced by n+2. The equations corresponding to Eqs. (21) and (26) are

$$(\bar{\bar{A}}_3)^{n+2,\varrho}_{j,k} \cdot \bar{U}^{n+2,\varrho+1}_{j,k} + (\bar{\bar{B}}_3)^{n+2,\varrho}_{j,k} \cdot \bar{U}^{n+2,\varrho+1}_{j,k+1} + (\bar{\bar{C}}_3)^{n+2,\varrho}_{j,k} \cdot \bar{U}^{n+2,\varrho+1}_{j,k-1} = (\bar{V}_3)^{n+2,\varrho}_{j,k} \qquad (32)$$

and

$$\bar{U}^{n+2,\varrho+1}_{j,k} = \bar{\bar{E}}^{n+2,\varrho}_{j,k} \cdot \bar{U}^{n+2,\varrho+1}_{j,k+1} + \bar{F}^{n+2,\varrho}_{j,k} , \qquad (33)$$

respectively. The derivation of expressions for $\bar{\bar{E}}^{n+2,\varrho}_{j,2}$, $\bar{F}^{n+2,\varrho}_{j,2}$, $\bar{\bar{E}}^{n+2,\varrho}_{j,k}$, $F^{n+2,\varrho}_{j,k}$, $\bar{U}^{n+2,\varrho}_{j,K}$ corresponding to Eqs. (27) to (31) is straightforward. Thus, the algorithms for solving both Eqs. (19) and (20) are identical except for the method of establishing the coefficients $\bar{\bar{E}}$, $\bar{\bar{H}}$, and $\bar{F}$ of Eqs. (22) to (25) and the coefficients $\bar{\bar{A}}$, $\bar{\bar{B}}$, $\bar{\bar{C}}$, and $\bar{V}$ of Eqs. (21) and (32). (Rigorously speaking, $\bar{F}$ and $\bar{V}$ are not "coefficients.") The only other difference in the two algorithms is where the computed results, Eqs. (26) and (33), are stored. The similarity in the two algorithms is used to minimize the coding in ANIMAL.

As formulated by Eqs. (19) to (33), ANIMAL's basic algorithm is quite general and need not be restricted to a five-component solution vector $\bar{U}$ as indicated in Eq. (18). ANIMAL is in fact set up to calculate subsets of the model equations. The following subsets can be selected in addition to Eq. (18):

(1) $\bar{U} = (\rho, \epsilon, B)$ in one-dimension, i.e., one-dimensional diffusive transport.
(2) $\bar{U} = (\rho, \epsilon, B)$ in two dimensions.
(3) $\bar{U} = (\rho, v_1, \epsilon)$, i.e., one-dimensional hydrodynamics.
(4) $\bar{U} = (\rho, v_1, v_3, \epsilon)$, i.e., two-dimensional hydrodynamics.
(5) $\bar{U} = (\rho, v_1, \epsilon, B_2)$, i.e., one-dimensional MHD.

For Eq. (18) and each of the subsets a variety of physics options are available; e.g., $\bar{U} = (\rho, v_1, \epsilon)$ can be ideal one-dimensional hydrodynamics if the thermal conductivity and radiation are set to zero. In addition, because of the generality, the ANIMAL algorithm is set up to handle as many as ten variables in anticipation of the addition of more dependent variables. For example, most of the structure to handle additional magnetic field components B_1 and B_3 is already in the code (ANIMAL's predecessor[17] did in fact calculate B_1 and B_3); what is missing is merely coding to determine the appropriate coefficients, and this would be a relatively minor fraction of the entire coding.

It is important to note that for the one dimensional subsets, $\bar{Y}$ of Eqs. (19) and (20) are identically zero. Hence, in one-dimensional calculations, ANIMAL uses a fully *implicit* method [Eq. (19)] to advance from t^n to t^{n+1} and then ANIMAL uses a fully *explicit* method [Eq. (20)] to advance from t^{n+1} to t^{n+2}. By combining Eqs. (19) and (20), one can see that the one-dimensional difference equations relating U^{n+2} to U^n, for n even, appear to be Crank-Nicholson,[2] whereas those relating U^{n+3} to U^{n+1} appear to be "leapfrog."[2]

4. SPATIAL DIFFERENCING—GENERAL FORMALISM

The preceding section outlined the temporal-differencing techniques in ANIMAL. The spatial-difference equations in Eqs. (19) and (20) are formulated by integrating the model equations over the "control volume," or "mesh cell," or "zone," in the ξ_1–ξ_3 plane (shown schematically in Fig. 1). The following integrals of geometric quantities are relevant:

$$V^n_{j,k} = \int_{(\xi_1)_{j-1/2}}^{(\xi_1)_{j+1/2}} \int_{(\xi_3)_{k-1/2}}^{(\xi_3)_{k+1/2}} (h_1 h_2 h_3 x_{11} x_{33})^n \, d\xi_1 \, d\xi_3 \, . \tag{34}$$

$$S^n_{j+1/2,k} = \int_{(\xi_3)_{k-1/2}}^{(\xi_3)_{k+1/2}} (h_2 h_3 x_{32})^n_{j+1/2} \, d\xi_3 \, . \tag{35}$$

$$S^n_{j,k+1/2} = \int_{(\xi_1)_{j-1/2}}^{(\xi_1)_{j+1/2}} (h_1 h_2 x_{11})^n_{k+1/2} \, d\xi_1 \, . \tag{36}$$

$$L^n_{j,k} = \int_{(\xi_1)_{j-1/2}}^{(\xi_1)_{j+1/2}} \int_{(\xi_3)_{k-1/2}}^{(\xi_3)_{k+1/2}} \left(\frac{h_1 h_3 x_{11} x_{33}}{h_2} \right)^n d\xi_1 \, d\xi_3 \, . \tag{37}$$

$$D^n_{j+1/2,k} = \int_{(\xi_3)_{k-1/2}}^{(\xi_3)_{k+1/2}} \left(\frac{h_3 x_{33}}{h_2} \right)^n_{j+1/2} d\xi_3 \, . \tag{38}$$

$$D^n_{j,k+1/2} = \int_{(\xi_1)_{j-1/2}}^{(\xi_1)_{j+1/2}} \left(\frac{h_1 x_{11}}{h_2} \right)^n_{k+1/2} d\xi_1 \, . \tag{39}$$

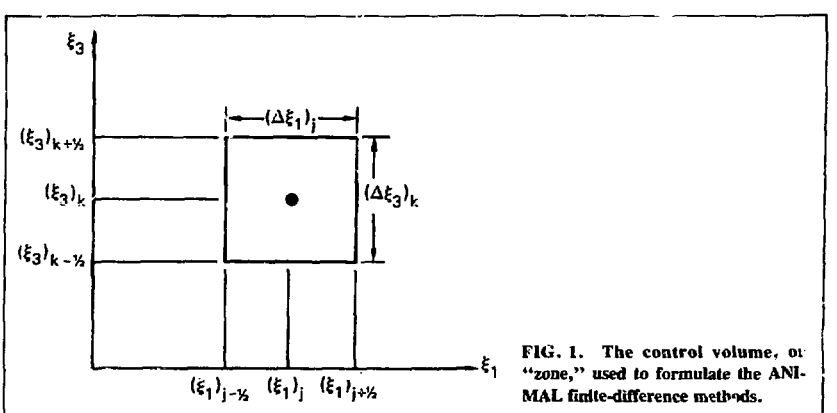


FIG. 1. The control volume, or "zone," used to formulate the ANIMAL finite-difference methods.

In Eq. (34), $V^n_{j,k}$ can be identified as the two-dimensional volume of the zone; the true volume is $V^n_{j,k}\,\Delta\xi_2$, where, for axisymmetric problems, $\Delta\xi_2 = 2\pi$. Similarly, $S^n_{j+1/2,k}$ and $S^n_{k,k+1/2}$ can be interpreted as the areas of the cell interfaces at $j+1/2$ and $k+1/2$, respectively. The quantities given in Eqs. (37) to (39) are relevant because it has been found to be advantageous to difference the magnetic field as if h_2B_2 was a dependent variable. $L^n_{j,k}$ can be interpreted as the "inductance" of the zone.

Equations (5) and (9) are totally in "conservation" form and the difference approximations exactly (except for machine roundoff) conserve mass and magnetic flux (the reader should be familiar with Ref. 3). Thus, the first component of Eq. (19), corresponding to Eq. (5), is differenced as

$$\frac{V^{n+1}_{j,k}\,\rho^{n+1}_{j,k} - V^n_{j,k}\,\rho^n_{j,k}}{t^{n+1} - t^n} + S^{n+1}_{j+1/2,k}\left[\rho(v_1 - v_1^G)\right]^{n+1}_{j+1/2,k}$$

$$- S^{n+1}_{j-1/2,k}\left[\rho(v_1 - v_1^G)\right]^{n+1}_{j-1/2,k}$$

$$+ S^n_{j,k+1/2}\left[\rho(v_3 - v_3^G)\right]^{n}_{j,k+1/2}$$

$$- S^n_{j,k-1/2}\left[\rho(v_3 - v_3^G)\right]^{n}_{j,k-1/2} = 0\,, \tag{40}$$

where the method for evaluating the dependent variables in the spatial differences has yet to be specified.

Many of the terms in Eqs. (6) to (8) are also "conservative" and represent *fluxes* of momentum or energy. However, terms that cannot be integrated once exactly in the double integration over $d\xi_1$ and $d\xi_3$ are "nonconservative" and represent what this author refers to as *forces*. An example of a *force* is the pressure gradient term of Eq. (6), which in fact is a real force. Force terms involving spatial derivatives of dependent variables are called *nonlocal* forces. These terms are integrated by assuming the force varies linearly over the direction in which the derivative is taken and by assuming the dependent variables are averages in the other direction, e.g.,

$$\int_{(\xi_1)_{j-1/2}}^{(\xi_1)_{j+1/2}} \int_{(\xi_3)_{k-1/2}}^{(\xi_3)_{k+1/2}} \left(h_2h_3x_{33}\,\frac{\partial p}{\partial \xi_1}\right)^{n+1} d\xi_1\,d\xi_3 =$$

$$\left((\xi_1)_{j+1/2} - (\xi_1)_{j-1/2}\right) \int_{(\xi_3)_{k-1/2}}^{(\xi_3)_{k+1/2}} \left[\frac{1}{2}\left(h_2h_3x_{33}\,\frac{\partial p}{\partial \xi_1}\right)^{n+1}_{j+1/2}\right.$$

$$\left. + \frac{1}{2}\left(h_2h_3x_{33}\,\frac{\partial p}{\partial \xi_1}\right)^{n+1}_{j-1/2}\right] d\xi_3 =$$

$$\left((\xi_1)_{j+1/2} - (\xi_1)_{j-1/2}\right)\left[\frac{1}{2}\,S^n_{j+1/2,k}\left(\frac{\partial p}{\partial \xi_1}\right)^{n+1}_{j+1/2,k}\right.$$

$$\left. + \frac{1}{2}\,S^n_{j-1/2,k}\left(\frac{\partial p}{\partial \xi_1}\right)^{n+1}_{j-1/2,k}\right]. \tag{41}$$

The treatment of nonlocal forces as implied by Eq. (41) is necessary to retain in the difference equations the "subconservation" properties of differential equations even though the model Eqs. (5) to (9) are not completely in "conservation" form.[3] Forces not involving spatial derivatives of dependent variables are termed *local* forces, e.g., the term $h_2\rho v_3 v_1 h_{13} x_{11}$ of Eq. (6).

From the form of Eq. (40) it should be apparent that, for fluxes, each zone on either side of an interface receives the same contribution from quantities on either side of the interface except for a change in sign (i.e., fluxes are antisymmetric about the interface). Similarly, from the form of Eq. (41), it should be apparent that, for nonlocal forces, each zone on either side of an interface receives the same contribution from quantities on either side of the interface with the same sign (i.e., nonlocal forces are symmetric about the interface). Thus, the coefficients of Eq. (21) can be written as

$$(\bar{\bar{A}}_1)^{n+1,\varrho}_{j,k} = \bar{\bar{\varrho}}^{n+1,\varrho}_{j,k} + (\bar{\bar{r}}_1)^{n+1,\varrho}_{j,k} + (\bar{\bar{a}}_1)^{n+1,\varrho}_{j+1/2,k} + (\bar{\bar{c}}_1)^{n+1,\varrho}_{j+1/2,k}$$

$$- (\bar{\bar{b}}_1)^{n+1,\varrho}_{j-1/2,k} + (\bar{\bar{d}}_1)^{n+1,\varrho}_{j-1/2,k} , \tag{42}$$

$$(\bar{\bar{B}}_1)^{n+1,\varrho}_{j,k} = (\bar{\bar{b}}_1)^{n+1,\varrho}_{j+1/2,k} + (\bar{\bar{d}}_1)^{n+1,\varrho}_{j+1/2,k} , \tag{43}$$

$$(\bar{\bar{C}}_1)^{n+1,\varrho}_{j,k} = -(\bar{\bar{a}}_1)^{n+1,\varrho}_{j-1/2,k} + (\bar{\bar{c}}_1)^{n+1,\varrho}_{j-1/2,k} , \tag{44}$$

and

$$(\bar{V}_1)^{n+1,\varrho}_{j,k} = \bar{m}^{n+1,\varrho}_{j,k} + (\bar{s}_1)^{n+1,\varrho}_{j,k} + (\bar{v}_1)^{n+1,\varrho}_{j+1/2,k} + (\bar{w}_1)^{n+1,\varrho}_{j+1/2,k}$$

$$- (\bar{v}_1)^{n+1,\varrho}_{j-1/2,k} + (\bar{w}_1)^{n+1,\varrho}_{j-1/2,k} - (\bar{g}^c_3)^n_{j,k}$$

$$- (\bar{g}^a_3)^n_{j,k+1/2} - (\bar{g}^s_3)^n_{j,k+1/2} + (\bar{g}^a_3)^n_{j,k-1/2} - (\bar{g}^s_3)^n_{j,k-1/2} . \tag{45}$$

In Eqs. (42) to (45), the various superscripts and subscripts imply a functional dependence for any quantity Q given by

$$Q^{n+1,\varrho}_{j,k} = Q\left(\bar{U}^{n+1,\varrho}_{j,k}\right) , \tag{46}$$

$$Q^{n+1,\varrho}_{j+1/2,k} = Q\left(\bar{U}^{n+1,\varrho}_{j,k} , \bar{U}^{n+1,\varrho}_{j+1,k}\right) , \tag{47}$$

and

$$Q^{n}_{j,k+1/2} = Q\left(\bar{U}^{n}_{j,k} , \bar{U}^{n}_{j,k+1}\right) . \tag{48}$$

Eqs. (42) to (45) result from

$$(\bar{f}^a_1)^{n+1,\varrho+1}_{j+1/2,k} = (\bar{\bar{b}}_1)^{n+1,\varrho}_{j+1/2,k} \cdot \bar{U}^{n+1,\varrho+1}_{j+1,k} + (\bar{\bar{a}}_1)^{n+1,\varrho}_{j+1/2,k} \cdot \bar{U}^{n+1,\varrho+1}_{j,k} - (\bar{v}_1)^{n+1,\varrho}_{j+1/2,k} , \tag{49}$$

$$(\bar{f}^s_1)^{n+1,\varrho+1}_{j+1/2,k} = (\bar{\bar{d}}_1)^{n+1,\varrho}_{j+1/2,k} \cdot \bar{U}^{n+1,\varrho+1}_{j+1,k} + (\bar{\bar{c}}_1)^{n+1,\varrho}_{j+1/2,k} \cdot \bar{U}^{n+1,\varrho+1}_{j,k} - (\bar{w}_1)^{n+1,\varrho}_{j+1/2,k} , \tag{50}$$

$$(\bar{f}^c_1)^{n+1,\varrho+1}_{j,k} = (\bar{\bar{r}}_1)^{n+1,\varrho}_{j,k} \cdot \bar{U}^{n+1,\varrho+1}_{j,k} - (\bar{s}_1)^{n+1,\varrho}_{j,k} , \tag{51}$$

$$\left\{ \frac{\bar{T}^{n+1}_{j,k} - \bar{T}^{n}_{j,k}}{t^{n+1} - t^{n}} \right\}^{n+1,\ell+1} = \bar{\bar{\ell}}^{n+1,\ell}_{j,k} \cdot \bar{U}^{n+1,\ell+1}_{j,k} - \bar{m}^{n+1,\ell}_{j,k} \quad , \tag{52}$$

and

$$\tilde{Y}^{n}_{j,k} = (\bar{g}^{c}_{3})^{n}_{j,k} + (\bar{g}^{a}_{3})^{n}_{j,k+1/2} + (\bar{g}^{s}_{3})^{n}_{j,k+1/2} - (\bar{g}^{a}_{3})^{n}_{j,k-1/2} + (\bar{g}^{s}_{3})^{n}_{j,k-1/2} \quad , \tag{53}$$

where $\bar{f}^{a}_{1}$ represents all the antisymmetic fluxes, $\bar{f}^{s}_{1}$ represents all the symmetric nonlocal forces, and $\bar{f}^{c}_{1}$ represents the local centered forces appearing in $\tilde{X}^{n+1}_{j,k}$ of Eq. (19). Thus, Eq. (49) is the Newton-Raphson linearization of the forces, and hence

$$(\bar{\bar{b}}_{1})^{n+1,\ell}_{j+1/2,k} = \left[\frac{\partial(\bar{f}^{a}_{1})_{j+1/2,k}}{\partial \bar{U}_{j+1,k}} \right]^{n+1,\ell} \quad , \tag{54}$$

$$(\bar{\bar{a}}_{1})^{n+1,\ell}_{j+1/2,k} = \left[\frac{\partial(\bar{f}^{a}_{1})_{j+1/2,k}}{\partial \bar{U}_{j,k}} \right]^{n+1,\ell} \quad , \tag{55}$$

and

$$(\bar{v}_{1})^{n+1,\ell}_{j+1/2} = -(\bar{f}^{a}_{1})^{n+1,\ell}_{j+1/2,k} + (\bar{\bar{b}}_{1})^{n+1,\ell}_{j+1/2,k} \cdot \bar{U}^{n+1,\ell}_{j+1,k} + (\bar{\bar{a}}_{1})^{n+1,\ell}_{j+1/2,k} \cdot \bar{U}^{n+1,\ell}_{j,k} \quad , \tag{56}$$

i.e.; $\bar{\bar{b}}_{1}$ and $\bar{\bar{a}}_{1}$ are "Jacobian" matrices. Note that in Eq. (56), $(\bar{f}^{a}_{1})^{n+1,\ell}_{j+1/2,k}$ represents an *explicit* flux, which is only a function of $\bar{U}^{n+1,\ell}$. Throughout this report, the convention is adapted that any quantity with an iteration subscript of the form $\ell+1$ is a function of both $\bar{U}^{n+1,\ell+1}$ and $\bar{U}^{n+1,\ell}$. The functional dependence implied by superscripts on any quantity Q is

$$Q^{n+1,\ell+1} = Q(\bar{U}^{n+1,\ell+1}, \bar{U}^{n+1,\ell}) \quad , \tag{57}$$

$$Q^{n+1,\ell} = Q(\bar{U}^{n+1,\ell}) \quad , \tag{58}$$

and

$$Q^{n} = Q(\bar{U}^{n}) \quad . \tag{59}$$

All of the coefficients appearing in Eqs. (50) to (52) are also determined as a Newton-Raphson linearization. In Eq. (53), $\bar{g}^{a}_{3}$, $\bar{g}^{s}_{3}$, and $\bar{g}^{c}_{3}$ represent the fluxes, nonlocal forces, and local forces, respectively, appearing in $\tilde{Y}^{n}_{j,k}$ which appears *explicitly* in Eq. (19).

In a similar fashion, the coefficients in Eq. (32) can be written as

$$(\bar{\bar{A}}_{3})^{n+2,\ell}_{j,k} = \bar{\bar{\ell}}^{n+2,\ell}_{j,k} + (\bar{\bar{r}}_{3})^{n+2,\ell}_{j,k} + (\bar{\bar{a}}_{3})^{n+2,\ell}_{j,k+1/2} + (\bar{\bar{c}}_{3})^{n+2,\ell}_{j,k+1/2} - (\bar{\bar{h}}_{3})^{n+2,\ell}_{j,k-1/2} + (\bar{\bar{d}}_{3})^{n+2,\ell}_{j,k-1/2} \quad , \tag{60}$$

$$(\bar{\bar{B}}_3)^{n+2,\varrho}_{j,k} = (\bar{\bar{b}}_3)^{n+2,\varrho}_{j,k+1/2} + (\bar{\bar{d}}_3)^{n+2,\varrho}_{j,k+1/2} \quad , \tag{61}$$

$$(\bar{\bar{C}}_3)^{n+2,\varrho}_{j,k} = -(\bar{\bar{a}}_3)^{n+2,\varrho}_{j,k-1/2} + (\bar{\bar{c}}_3)^{n+2,\varrho}_{j,k-1/2} \quad , \tag{62}$$

and

$$\begin{aligned}(\bar{V}_3)^{n+2,\varrho}_{j,k} = {} & \bar{m}^{n+2,\varrho}_{j,k} + (\bar{s}_3)^{n+2,\varrho}_{j,k} + (\bar{v}_3)^{n+2,\varrho}_{j,k+1/2} + (\bar{w}_3)^{n+2,\varrho}_{j,k+1/2} \\ & - (\bar{v}_3)^{n+2,\varrho}_{j,k-1/2} + (\bar{w}_3)^{n+2,\varrho}_{j,k-1/2} - (\bar{f}^{\,c}_1)^{n+1}_{j,k} \\ & - (\bar{f}^{\,a}_1)^{n+1}_{j+1/2,k} - (\bar{f}^{\,s}_1)^{n+1}_{j+1/2,k} + (\bar{f}^{\,a}_1)^{n+1}_{j-1/2,k} - (\bar{f}^{\,s}_1)^{n+1}_{j-1/2,k} \quad ,\end{aligned} \tag{63}$$

where, analogous to Eq. (49),

$$(\bar{\bar{g}}^{\,a}_3)^{n+2,\varrho+1}_{j,k+1/2} = (\bar{\bar{b}}_3)^{n+2,\varrho}_{j,k+1/2} \cdot \bar{U}^{n+2,\varrho+1}_{j,k+1} + (\bar{\bar{a}}_3)^{n+2,\varrho}_{j,k+1/2} \cdot \bar{U}^{n+2,\varrho+1}_{j,k} - (\bar{v}_3)^{n+2,\varrho}_{j,k+1/2} \tag{64}$$

is the Newton-Raphson linearization of the fluxes $\bar{g}^a_3$. Similar linearizations for $\bar{g}^s_3$ and $\bar{g}^c_3$, analogous to Eqs. (50) and (51), are also used in Eqs. (50) to (63); and Eq. (52), with superscript n+1 replaced by n+2, is also used.

The coefficients $\bar{A}$, $\bar{\bar{B}}$, $\bar{\bar{C}}$, and $\bar{V}$ of Eqs. (21) and (32) have been broken up into the form given by Eqs. (42) to (45) and Eqs. (60) to (63), respectively, to facilitate implementation of the algorithm into actual FORTRAN coding. Thus, for example, prior to the forward sweep along line k indicated by Eqs. (21) to (31), the "explicit" fluxes $(\bar{g}^a_1)^n_{j,k+1/2}$ and "explicit" forces $(\bar{g}^s_1)^n_{j,k+1/2}$ are computed for all $1 < j < J$ and stored. They are then recalled as required in the forward sweep to contribute to $\bar{V}_1$ as indicated in Eq. (45). When the iterations along line k converge, the algorithm is then applied to line $k' = k+1$. The explicit fluxes and forces at the $k'+1/2$ interfaces are again calculated and stored, but the fluxes and forces at $k'-1/2$ need not be recomputed, for they are merely the previously computed and stored values for $k+1/2$. Hence, when advancing time from t^n to t^{n+1}, there is no coding in which the mass flux represented by the last term on the left side of Eq. (40) is explicitly calculated (except at boundaries, to be described later); there is only coding corresponding to the fourth term of Eq. (40). In a similar fashion, the coefficients $\bar{\bar{a}}$, $\bar{\bar{b}}$, $\bar{v}$, $\bar{\bar{c}}$, $\bar{\bar{d}}$, and $\bar{w}$, which appear in Eqs. (42) to (45), are computed only for a $j+1/2$ interface and are then stored where they are used for the $j'-1/2$ interface when $j' = j+1$. In essence, then, only one-half the flux and nonlocal force-difference equations are actually coded, and this minimizes the possibility of error.

Just as there is symmetry (or antisymmetry) about an interface in the algorithm, there is also a symmetry in the model equations with respect to the interchange of the variables ξ_1 and ξ_3 (and subscripts 1 and 3). And there is a symmetry in the corresponding difference equations when ξ_1 and ξ_3 are interchanged and the time superscripts n+1 and n+2 are interchanged. This symmetry is a result of retaining the scale factors h_1, h_2, and h_3 in the model equations and in the corresponding difference equations. Thus, the same coding used to calculate $(\bar{\bar{g}}^a_3)^n_{j,k+1/2}$ in Eq. (45) is used to calculate $(\bar{\bar{f}}^a_1)^{n+1}_{j+1/2,k}$ in Eq. (63). Similarly, the same coding used to calculate the coefficients $(\bar{\bar{a}}_1)^{n+1,\varrho}_{j+1/2,k}$, $(\bar{\bar{b}}_1)^{n+1,\varrho}_{j+1/2,k}$, and $(\bar{v}_1)^{n+1,\varrho}_{j+1/2,k}$ of Eq. (49) is used to calculate the coefficients $(\bar{\bar{a}}_3)^{n+2,\varrho}_{j,k+1/2}$, $(\bar{\bar{b}}_3)^{n+2,\varrho}_{j,k+1/2}$, and $(\bar{v}_3)^{n+2,\varrho}_{j,k+1/2}$ of Eq. (64), and so forth.

Consequently, to understand nearly the entire ANIMAL coding, it is sufficient to understand only the procedure for advancing from t^n to t^{n+1}, i.e., the procedure for solving Eq. (19). In addition, when more physical effects are to be added to the code, or when finite-difference techniques are to be changed, it is nearly sufficient to make the appropriate modifications to the coding $\bar{g}^s_1$, $\bar{g}^a_1$, and $\bar{g}^c_1$ of Eq. (45) and to the coding for the coefficients of Eqs. (49) to (52); the implied modifications to Eqs. (60) to (64) are automatically taken into account. There are, of course, sections of the coding that must differentiate between ξ_1 and ξ_3 and between $t^n \to t^{n+1}$ and $t^{n+1} \to t^{n+2}$, but these sections are relatively small and relatively static.

In the linearized difference equations, Eqs. (21) and (32), the contribution of any of the fluxes or forces appearing in the model equations can be identified as a single number only if the fluxes and forces appear *explicitly*; i.e., only if the fluxes and forces appear in $\bar{Y}$ or in $\bar{X}$ of Eq. (20). Thus, for example, when working with Eq. (21) to advance from t^n to t^{n+1}, the value for the mass fluxes corresponding to the last two terms on the left side of Eq. (40) can readily be obtained as a single number. On the other hand, the values for the mass fluxes corresponding to the second and third terms of Eq. (40) cannot readily be obtained, for these fluxes appear *implicitly* in Eq. (21); i.e., the actual value for the mass fluxes can only be obtained after Eq. (21) is solved. The fact that the implicit fluxes and forces actually do have a single value, computable only after solution of the implicit equations, seems to be a fact not usually considered by people working with implicit equations. In ANIMAL, however, this fact is used as a debugging diagnostic. For example, suppose the iteration procedure using Eq. (21) converges to within a factor δ after $L+1$ iterations. If the coefficients $\bar{\bar{b}}_1$, $\bar{\bar{a}}_1$, and $\bar{v}_1$ of Eqs. (54) to (56) have been saved, one can compute the flux values using Eq. (49), i.e.,

$$(\bar{f}_1)^{n+1,L+1}_{j+1/2,k} = (\bar{\bar{b}}_1)^{n+1,L}_{j+1/2,k} \cdot \bar{U}^{n+1,L+1}_{j+1,k} + (\bar{\bar{a}}_1)^{n+1,L}_{j+1/2,k} \cdot \bar{U}^{n+1,L+1}_{j,k} - (\bar{v}_1)^{n+1,L}_{j+1/2,k} \quad . \tag{65}$$

Since the iterations converge, the identification $\bar{U}^{n+1}_{j,k} = \bar{U}^{n+1,L+1}_{j,k}$ is made. The explicit flux $(\bar{f}^a_1)^{n+1}_{j+1/2,k}$, which appears in Eq. (63), can then be computed. If the coefficients $\bar{\bar{b}}_1$, $\bar{\bar{a}}_1$, and $\bar{v}_1$ of Eq. (65), used to advance from t^n to t^{n+1}, were correctly computed and if the explicit computation of $(\bar{f}^a_1)^{n+1}_{j+1/2,k}$, which is to be used to advance from t^{n+1} to to t^{n+2} was also correct, then

$$(\bar{f}^a_1)^{n+1}_{j+1/2,k} = (\bar{f}^a_1)^{n+1,L+1}_{j+1/2,k} + 0(\delta^2) \quad , \tag{66}$$

a consequence of the fact that the Newton–Raphson method is second order. In ANIMAL, the capability for performing the check implied by Eq. (66) is built in for checking all implicit fluxes and forces. If the difference is not $0(\delta^2)$, then the existence of an error is implied (of course, sometimes the error is in the coding doing the checking).

The symmetry of the algorithm provides another useful check used in ANIMAL. In the algorithm, when the coefficients $\bar{\bar{A}}$, $\bar{\bar{B}}$, $\bar{\bar{C}}$, and $\bar{V}$ of Eqs. (42) to (45) are being assembled, $\bar{\bar{\ell}}^{n+1,\ell}_{j,k}$ and $\bar{m}^{n+1,\ell}_{j,k}$, corresponding to the time derivatives as indicated in Eq. (52), are the last contributions to be incorporated. The algorithm can be intercepted on the first iteration prior to the incorporation of ℓ and $\bar{m}$, and the coefficients are modified so the actual coefficients used in the tridiagonal solver have the form

$$(\bar{\bar{A}}'_1)^{n+1,0}_{j,k} = \bar{\bar{\ell}}^{n+1,0}_{j,k} \quad , \tag{67}$$

$$(\bar{\bar{B}}'_1)^{n+1,0}_{j,k} = (\bar{\bar{C}}'_1)^{n+1,0}_{j,k} = 0 \quad , \tag{68}$$

and

$$(\bar{V}')^{n+1,0}_{j,k} = (\bar{V}_1)^{n+1,0}_{j,k} - \left[(\bar{\bar{A}}_1)^{n+1,0}_{j,k} - \bar{\bar{\ell}}^{n+1,0}_{j,k}\right] \cdot \bar{U}^{n+1,0}_{j,k} - (\bar{\bar{B}}_1)^{n+1,0}_{j,k} \cdot \bar{U}^{n+1,0}_{j+1,k} - (\bar{\bar{C}}_1)^{n+1,0}_{j,k} \cdot \bar{U}^{n+1,0}_{j-1,k} \quad , \tag{69}$$

where the unprimed coefficients $\bar{\bar{A}}$, $\bar{\bar{B}}$, $\bar{\bar{C}}$, and $\bar{V}$ are computed according to Eqs. (42) to (45). Using the primed coefficients in Eq. (21) is equivalent to doing an explicit calculation, since for example, using Eq. (56),

$$(\bar{\bar{a}}_1)^{n+1,0}_{j+1/2,k} \cdot \bar{U}^{n+1,0}_{j,k} + (\bar{\bar{b}}_1)^{n+1,0}_{j+1/2,k} \cdot \bar{U}^{n+1,0}_{j+1,k} - (\bar{v}_1)^{n+1,0}_{j+1/2,k} = (\bar{f}^a_1)^{n+1,0}_{j+1/2,k} = (\bar{f}^a_1)^{n}_{j+1/2,k} \quad . \tag{70}$$

In a similar manner the algorithm normally used to advance from t^{n+1} to t^{n+2} can be used instead to advance from t^n to t^{n+1}, be intercepted on the first iteration, and be made to effectively perform an explicit calculation by modifying the coefficients analogously to Eqs. (67) to (69). Thus, there are two distinct ways to calculate an explicit $\bar{U}^{n+1,1}$; and the resultant values should be identical, or errors in the coding are implied. This type of check is normally performed along with the checks implied by Eq. (66) whenever modifications to ANIMAL are made. These checks are really like a single parity check in that they verify that the number of errors is even, not odd, and hopefully that the even number is zero.

As indicated previously, there is a single section in ANIMAL that calculates implicit forces and fluxes associated with an *interior* interface and a single section in the code that calculates explicit forces and fluxes associated with an *interior* interface. The forms used for fluxes and forces across a *boundary* interface, however, are sufficiently different that separate sections of coding for both left (or lower) and right (or upper) boundary fluxes are required. This is a result of the fact that, in ANIMAL, interior dependent variables $\bar{U}_{j,k}$ for $2 \leq j \leq J-1$ and $2 \leq k \leq K-1$ are interpreted to be "zone-centered," or average values. Each interior point $[(\xi_1)_j, (\xi_3)_k]$ is located a full cell-width away from its interior neighbors. On the other hand, boundary values are interpreted as being boundary *interface* values, not boundary cell, or "half cell," values. Since the boundary values are interface values, they are located a half cell-width away from the first interior values, as indicated in Fig. 2. Boundary values are not included in any volume integrations, so, for example, the total mass within the domain is

$$M^n = \sum_{j=2}^{J-1} \sum_{k=2}^{K-1} V^n_{j,k} \rho^n_{j,k} \quad . \tag{71}$$

The above described interpretation of boundary and interior values facilitates the incorporation of unique boundary conditions[4] and facilitates the maintenance of conservation properties.[3]

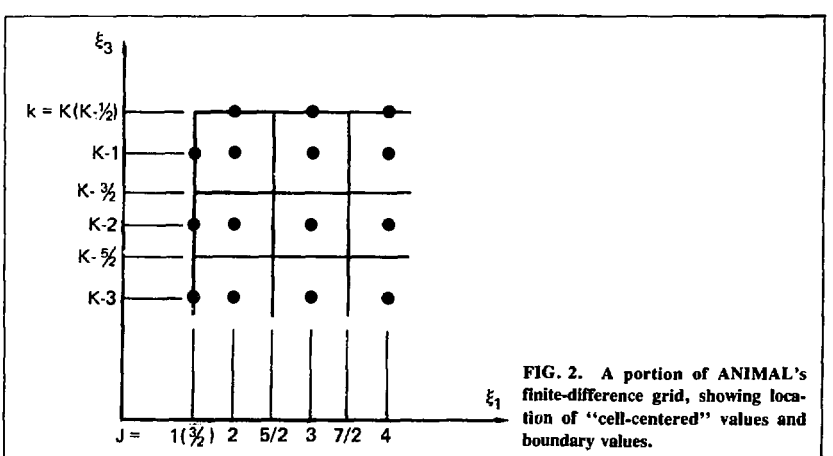


FIG. 2. A portion of ANIMAL's finite-difference grid, showing location of "cell-centered" values and boundary values.

5. BASIC DIFFERENCE EQUATIONS FOR TIME DERIVATIVES

The time derivatives appearing in the model equations are treated in a nearly straightforward manner. The time derivatives in Eqs. (5) and (9) are linear in the dependent variables and hence need no linearization. On the other hand, the time derivatives in Eqs. (6) to (8) are nonlinear. In spite of earlier claims in this report, the time derivatives *are not* linearized by straightforward application of the Newton-Raphson procedure. ANIMAL's procedure is a result of the fact that its predecessor[17] did not iterate to convergence, but rather accepted the first iterate $\bar{U}^{n+1,1}$ as the final value. As shown by Lindemuth and Killeen,[17] when the two alternately used approximations to the spatial derivatives are considered as a whole, even with only one iteration, they are overall-accurate to $0(\Delta t^2)$. For the time derivatives to be overall $0(\Delta t^2)$, the linearization process can not introduce any $0(\Delta t)$ error other than that introduced by writing

$$\left[\frac{\partial(\rho\epsilon)}{\partial t}\right]^{n+1} = \frac{\rho^{n+1}\,\epsilon^{n+1} - \rho^{n}\,\epsilon^{n}}{\Delta t} + 0(\Delta t) \quad . \tag{72}$$

Newton-Raphson linearization of the first term on the right of Eq. (72) leads to

$$\left(\frac{\rho^{n+1}\,\epsilon^{n+1} - \rho^{n}\,\epsilon^{n}}{\Delta t}\right)^{n+1,1} = \epsilon^{n}\,\frac{\rho^{n+1,1} - \rho^{n}}{\Delta t} - \rho^{n}\,\frac{\epsilon^{n+1,1} - \epsilon^{n}}{\Delta t} \quad . \tag{73}$$

A truncation error analysis of Eq. (73) shows that additional $0(\Delta t)$ errors are introduced by the linearization. These errors are not canceled when advancing from t^{n+1} to t^{n+2}. Hence the overall approximation, if only one iteration were used, would be still only $0(\Delta t)$. This fact has been realized independently by Briley and MacDonald,[29] who essentially use the same linearization algorithm to solve the Navier–Stokes equations as Lindemuth and Killeen[17] used in magnetohydrodynamics. On the other hand, a truncation error analysis of the ANIMAL method given below shows that the only $0(\Delta t)$ error introduced is that indicated in Eq. (72), so even if only one iteration is performed, ANIMAL's treatment of the nonlinear time derivatives is still overall correct to $0(\Delta t^2)$. Since ANIMAL now iterates to convergence, it is not clear whether or not this treatment is necessary or useful. The ANIMAL treatment uses a pseudo-explicit approximation to $\rho_{j,k}^{n+1}$ based on both $\bar{U}^{n,\varrho}$ and $\bar{U}^{n}$. Consider the full difference approximation for the continuity equation to have the form

$$\frac{V_{j,k}^{n+1}\,\rho_{j,k}^{n+1} - V_{j,k}^{n}\,\rho_{j,k}^{n}}{t^{n+1} - t^{n}} + X_1(\bar{U}^{n+1}) + Y_1(\bar{U}^{n}) = 0 \quad , \tag{74}$$

which corresponds to the first component of Eq. (19). The pseudo-explicit $\tilde{\rho}_{j,k}^{n+1,\varrho}$ is then calculated from

$$\frac{V_{j,k}^{n+1}\,\tilde{\rho}_{j,k}^{n+1,\varrho} - V_{j,k}^{n}\,\rho_{j,k}^{n}}{t^{n+1} - t^{n}} + X_1(\bar{U}^{n+1,\varrho}) + Y_1(\bar{U}^{n}) = 0 \quad . \tag{75}$$

Prior to the introduction of the basic time-derivative difference equations, it is convenient to introduce a change in notation. As defined in Section 2, the velocities v_1^G and v_3^G are the "grid" velocities, i.e., the velocities of the moving coordinates x_1 and x_3 with respect to the "fixed" coordinates ξ_1 and ξ_3, respectively. As shown in Eq. (40), it is actually the *relative* velocity $v_1 - v_1^G$ that determines the convection of mass (momentum, energy, magnetic flux). Hence, in ANIMAL, the dependent variables are the relative velocities, rather than the total velocities, which are given by

$$v_1^R = v_1 - v_1^G \tag{76}$$

and

$$v_3^R = v_3 - v_3^G \quad . \tag{77}$$

It is also convenient to introduce a relative, or "plasma," magnetic field such that

$$B_2^P = B_2 - B_2^E \quad , \tag{78}$$

such that the electrical current flowing in the plasma is

$$\bar{J}^P = \frac{1}{\mu_0} \nabla \times \bar{B}^P \quad . \tag{79}$$

Usually, $\nabla \times \bar{B}^E = 0$ in the region of solution; i.e., $\bar{B}^E$ represents the magnetic field due to currents external to the domain. However, in calculations of an advanced relativistic electron-beam target,[7] $\bar{B}^E$ represented the magnetic field due to the electron beam. In the electron-beam target, $\nabla \times \bar{B}^E \neq 0$ in the region of solution, since the electron beam represented an additional current flowing through the target; however, the component of current that led to plasma forces and plasma ohmic heating was still given by Eq. (79).

In all cases, v_1^G, v_3^G, and B_2^E are assumed to be known functions of time and space (i.e., they are specified prior to the initiation of a computation). The quantities v_1^G, v_3^G, and B_2^E can be always identically zero, in which case, v_1^R, v_3^R, and B_2^P represent the actual velocities and magnetic field, respectively.

In the presentation below, the term in the model equation is indicated on the left side of the "approximately equal" sign and the difference equations that define the coefficients corresponding to Eq. (52) appear on the right side. The double-integration signs are shorthand for the integral of $d\xi_1\, d\xi_3$ from $(\xi_1)_{j-1/2}$ to $(\xi_1)_{j+1/2}$ and from $(\xi_3)_{k-1/2}$ to $(\xi_3)_{k+1/2}$.

The basic time-derivative difference equations used in ANIMAL are

$$\frac{\partial}{\partial t} \iint h_1 h_2 h_3 x_{11} x_{13}\, \rho\, d\xi_1\, d\xi_3 \approx \frac{V_{j,k}^{n+1}\, \rho_{j,k}^{n+1,l+1} - V_{j,k}^{n}\, \rho_{j,k}^{n}}{t^{n+1} - t^{n}} \quad , \tag{80}$$

$$\begin{aligned} \frac{\partial}{\partial t} \iint & h_1 h_2 h_3 x_{11} x_{13}\, \rho \left(v_1^R + v_1^G \right) d\xi_1\, d\xi_3 \\ & \approx V_{j,k}^{n+1}\, \tilde{\rho}_{j,k}^{n+1,\ell} \frac{\left(v_1^R\right)_{j,k}^{n+1,\ell+1} + \left(v_1^G\right)_{j,k}^{n+1} - \left(v_1^R\right)_{j,k}^{n} - \left(v_1^G\right)_{j,k}^{n}}{t^{n+1} - t^{n}} \\ & \quad + \left[\left(v_1^R\right)_{j,k}^{n} + \left(v_1^G\right)_{j,k}^{n} \right] \frac{V_{j,k}^{n+1}\, \rho_{j,k}^{n+1,\ell+1} - V_{j,k}^{n}\, \rho_{j,k}^{n}}{t^{n+1} - t^{n}} \quad , \end{aligned} \tag{81}$$

$$\begin{aligned} \frac{\partial}{\partial t} \iint & h_1 h_2 h_3 x_{11} x_{33}\, \rho \left(v_3^R + v_3^G \right) d\xi_1\, d\xi_3 \\ & \approx V_{j,k}^{n+1}\, \tilde{\rho}_{j,k}^{n+1,\ell} \frac{\left(v_3^R\right)_{j,k}^{n+1,\ell+1} + \left(v_3^G\right)_{j,k}^{n+1} - \left(v_3^R\right)_{j,k}^{n} - \left(v_3^G\right)_{j,k}^{n}}{t^{n+1} - t^{n}} \\ & \quad + \left[\left(v_3^R\right)_{j,k}^{n} + \left(v_3^G\right)_{j,k}^{n} \right] \frac{V_{j,k}^{n+1}\, \rho_{j,k}^{n+1,\ell+1} - V_{j,k}^{n}\, \rho_{j,k}^{n}}{t^{n+1} - t^{n}} \quad , \end{aligned} \tag{82}$$

$$\frac{\partial}{\partial t}\iint h_1 h_2 h_3 x_{11} x_{33}\,\rho\epsilon\,d\xi_1\,d\xi_3$$

$$\approx V_{j,k}^{n+1}\,\tilde{\rho}_{j,k}^{n+1,\ell}\,\frac{\epsilon_{j,k}^{n+1,\ell+1}-\epsilon_{j,k}^{n}}{t^{n+1}-t^{n}} + \epsilon_{j,k}^{n}\,\frac{V_{j,k}^{n+1}\,\rho_{j,k}^{n+1,\ell+1}-V_{j,k}^{n}\,\rho_{j,k}^{n}}{t^{n+1}-t^{n}}\ , \tag{83}$$

and

$$\frac{\partial}{\partial t}\iint h_1 h_3 x_{11} x_{33}\left(B_2^P + B_2^E\right)\,d\xi_1\,d\xi_3 \approx$$

$$\frac{L_{j,k}^{n+1}\left(h_2\right)_{j,k}^{n+1}\left[\left(B_2^P\right)_{j,k}^{n+1,\ell+1}+\left(B_2^E\right)_{j,k}^{n+1}\right]-L_{j,k}^{n}\left(h_2\right)_{j,k}^{n}\left[\left(B_2^P\right)_{j,k}^{n}+\left(B_2^E\right)_{j,k}^{n}\right]}{t^{n+1}-t^{n}}\ . \tag{84}$$

Expressions identical to Eqs. (80) through (84), with superscript n+1 replaced by n+2 and superscript n replaced by n + 1, are used to advance from t^{n+1} to t^{n+2}. Except that in Eqs. (81) to (83), $\tilde{\rho}_{j,k}^{n+1,\ell}$, calculated according to Eq. (75), is replaced by $\tilde{\rho}_{j,k}^{n+2,\ell}$, calculated according to

$$\frac{V_{j,k}^{n+2}\,\tilde{\rho}_{j,k}^{n+2,\ell}-V_{j,k}^{n+1}\,\rho_{j,k}^{n+1}}{t^{n+2}-t^{n+1}} + X_1(\bar{U}^{n+1}) + Y_1(\bar{U}^{n+2,\ell}) = 0\ , \tag{85}$$

which corresponds to the first component of Eq. (20).

6. VELOCITY FRACTIONAL-STEP TIME-DIFFERENCING

In ANIMAL, there is actually a choice of three different methods of time-differencing, the method given in the previous section and the ones to be outlined in this section and the next. From the basic time-differencing formalism of Section 3, it should be apparent that in Eqs. (19) and (20), only $(v_1)^{n+1}$, and not $(v_1)^n$ or $(v_1)^{n+2}$ appear in the two successive approximations for the model Eqs. (5), (8), and (9); $(v_1)^n$ and $(v_1)^{n+2}$ appear only in the approximations for the model Eqs. (6) and (7). Similarly, only $(v_3)^n$ and $(v_3)^{n+2}$ appear in the approximations for Eqs. (5), (8), and (9); $(v_3)^{n+1}$ appears only in the approximations for Eqs. (6) and (7). Under certain circumstances, the velocities . . . $(v_1)^{n-2}$, $(v_1)^n$, $(v_1)^{n+2}$ . . . oscillate about the relatively constant values . . . $(v_1)^{n-1}$, $(v_1)^{n+1}$ . . . . The exact origin of these oscillations is unclear, although occasionally they can be attributed to improper timestep control or insufficient iterating convergence. The oscillations are driven by the pressure gradiant and magnetic forces. They are generally related to the "explicit–implicit" character of ADI and high "Courant" numbers; they are short wavelength oscillations, which have no business being there. These oscillations are not unstable (or at worst very weakly unstable), but they sometimes wreak havoc with the timestep control. Therefore, the "velocity fractional step" approach to time-differencing was developed.

Normally,

$$\bar{U}^{n+1} = \left[\rho^{n+1}\ ,\ (v_1)^{n+1}\ ,\ (v_3)^{n+1}\ ,\ \epsilon^{n+1}\ ,\ (B_2)^{n+1}\right] \tag{86}$$

and v_1 and v_3 are advanced by equations of the form

$$\frac{V^{n+1}_{j,k}\ \rho^{n+1}_{j,k}\ (v_1)^{n+1}_{j,k} - V^{n}_{j,k}\ \rho^{n}_{j,k}\ (v_1)^{n}_{j,k}}{t^{n+1} - t^{n}} + X_2(\bar{U}^{n+1}) + Y_2(\bar{U}^{n}) = 0 \tag{87}$$

and

$$\frac{V^{n+2}_{j,k}\ \rho^{n+2}_{j,k}\ (v_3)^{n+2}_{j,k} - V^{n+1}_{j,k}\ \rho^{n+1}_{j,k}\ (v_3)^{n+1}_{j,k}}{t^{n+2} - t^{n+1}} + X_4(\bar{U}^{n+1}) + Y_4(\bar{U}^{n+2}) = 0\ . \tag{88}$$

In the velocity fractional-step approach, new dependent variable vectors are defined as

$$\bar{U}^{n} = \left[\rho^{n}\ ,\ (v_1)^{n-1}\ ,\ (v_3)^{n}\ ,\ \epsilon^{n}\ ,\ (B_2)^{n}\right]\ , \tag{89}$$

$$\bar{U}^{n+1} = \left[\rho^{n+1}\ ,\ (v_1)^{n+1}\ ,\ (v_3)^{n}\ ,\ \epsilon^{n+1}\ ,\ (B_2)^{n+1}\right]\ , \tag{90}$$

$$\bar{U}^{n+2} = \left[\rho^{n+2}\ ,\ (v_1)^{n+1}\ ,\ (v_3)^{n+2}\ ,\ \epsilon^{n+2}\ ,\ (B_2)^{n+2}\right]\ ; \tag{91}$$

and Eqs. (87) and (88) are replaced by

$$V^{n}_{j,k}\ \rho^{n}_{j,k}\ \frac{(v_1)^{n+1}_{j,k} - (v_1)^{n-1}_{j,k}}{t^{n+1} - t^{n-1}} + X_2(\bar{U}^{n+1}) - (v_1)^{n+1}_{j,k}\ X_1(\bar{U}^{n+1}) + Y_2(\bar{U}^{n}) - (v_1)^{n-1}_{j,k}\ Y_1(\bar{U}^{n}) = 0 \tag{92}$$

and

$$V^{n+1}_{j,k}\ \rho^{n+1}_{j,k}\ \frac{(v_3)^{n+2}_{j,k} - (v_3)^{n}_{j,k}}{t^{n+2} - t^{n}} + X_4(\bar{U}^{n+1}) - (v_3)^{n}_{j,k}\ X_1(\bar{U}^{n+1}) + Y_4(\bar{U}^{n+2}) - (v_3)^{n+2}_{j,k}\ Y_1(\bar{U}^{n+2}) = 0\ . \tag{93}$$

In essence, as shown by Eqs. (92) and (93), the velocity components are advanced every other timestep, always by an equation implicit in the direction of the velocity component and explicit in the direction perpendicular to the component. The possibility of instability in the explicit direction is introduced, but since the only process in the perpendicular direction is convection, instability can be eliminated by the use of "upwind" differencing for the convective term in the perpendicular direction. As can be shown when the actual form of X and Y are specified, Eqs. (87) and (88) exactly conserve Cartesian momentum, whereas Eqs. (92) and (93) conserve momentum only to $0(\Delta t)$. Where direct one- and two-dimensional comparisons between the normal method and the velocity fractional-step method are made, the apparent inaccuracies introduced by the latter are tolerable, and often the computation time is significantly decreased as a result of using increased timesteps. Perhaps unacceptable inaccuracy is introduced in strong shock cases, where the conservation properties of the difference equations are important. In one-dimensional problems, the inaccuracies introduced by the velocity fractional-step method are not as severe as those introduced by a fully implicit method, i.e., using Eq. (19) successively. In a sense, the velocity fractional-step method is not really a fractional-step method, since Eqs. (92) and (93) involve all physics and both directions.

Implementation of the velocity fractional-step method in ANIMAL was a quite minor task. All coding to calculate the coefficients of Eq. (21) [or Eq. (32)] was left intact. When the coding was executed, the quantity called $(v_1)^n$ was actually $(v_1)^{n-1}$. At the point where time derivatives are incorporated into the coefficients, the coefficients for the v_1 (v_3) equation were appropriately modified, and the coefficients for the v_3 (v_1) equation were zeroed. The algorithm still went through the motions of using 5×5 matrices, but the v_3 equation merely led to the result $(v_3)^{n+1} = (v_3)^n$. It would be more difficult (but advantageous from an execution time point of view) to make use of the fact that really only 4×4 matrices, as implied by Eqs. (90) and (91), are required.

7. IMPLICIT TIME-DIFFERENCING WITH $\beta > 1/2$

It was indicated in Section 6 that one pathology observed in the operation of ANIMAL was the oscillation of the values $(v_1)^{n-2}$, $(v_1)^n$, $(v_1)^{n+2}$ . . . about the values $(v_1)^{n-1}$, $(v_1)^{n+1}$, $(v_1)^{n+3}$ for n-even. Similar oscillations have also been observed in the internal energy and magnetic field. The oscillations in internal energy are driven by the thermal-diffusion terms and the oscillations in magnetic field are driven by the resistive-diffusion terms. As with the velocity oscillation, the exact origin of the oscillations is unclear, although occasionally they can be attributed to improper timestep control or insufficient iteration convergence. The oscillations are generally related to the "explicit–implicit" character of ADI and high "Courant" numbers. It can be shown that such oscillatory behavior occurs for short wavelengths at high "Courant" numbers for the basic ADI differencing. What is not clear is the origin of the short wavelengths, which have no business being there.

The implicitness of finite-difference equations is usually discussed in terms of an implicitness parameter β (not to be confused with the transverse thermoelectric coefficient β of section 2). For fully explicit difference equations, $\beta = 0$. And for fully implicit equations, $\beta = 1$. The basic alternating difference Eqs. (19) and (20) are equations with $\beta = 1/2$, at least for a fixed timestep. However, it seems plausible that a varying timestep could effectively lead to one dimension having an effective β slightly less than, or slightly more than, 1/2. If the effective β is slightly less than 1/2, one might expect conditional stability.

In this section, a method for doing alternating direction implicit calculations with $\beta > 1/2$ is presented. The intent of the method is to introduce additional implicitness beyond that of the method in Section 6 so that the amplitude of short-wavelength numerical oscillations is reduced.

As pointed out recently by Briley and MacDonald,[30] the basic alternating-direction, implicit finite-difference Eqs. (19) and (20) can be considered a special case of the general Douglas–Gunn[31] alternating-direction methods. The general Douglas–Gunn equations as applied to Eq. (17) can be considered to have the form

$$\frac{\bar{T}^* - \bar{T}^n}{t^* - t^n} + \beta\tilde{X}^* + (1 - \beta)\tilde{X}^n + \tilde{Y}^n = 0 \tag{94}$$

and

$$\frac{\bar{T}^{n+2} - \bar{T}^n}{t^{n+2} - t^n} + \beta\tilde{X}^* + (1-\beta)\tilde{X}^n + \beta\tilde{Y}^{n+2} + (1-\beta)\tilde{Y}^n = 0 \ . \tag{95}$$

Define t^{n+1} as

$$t^{n+1} = t^n + \beta(t^* - t^n) \ . \tag{96}$$

Then, for *linear* $\bar{T}$ and $\tilde{X}$,

$$\bar{T}^{n+1} = \bar{T}^n + \beta(\bar{T}^* - \bar{T}^n) \tag{97}$$

and

$$\tilde{X}^{n+1} = \beta\tilde{X}^* + (1 - \beta)\tilde{X}^n \ . \tag{98}$$

So Eq. (94) becomes

$$\frac{\bar{T}^{n+1} - \bar{T}^n}{t^{n+1} - t^n} + \tilde{X}^{n+1} + \tilde{Y}^n = 0 \ . \tag{99}$$

Using Eq. (99) to eliminate $\tilde{Y}^n$ from Eq. (95) leads to

$$\frac{\bar{T}^{n+2} - T^{n+1}}{t^{n+2} - t^{n+1}} + \frac{t^{n+2} - t^n}{t^{n+2} - t^{n+1}}\beta(\tilde{X}^{n+1} + \tilde{Y}^{n+2}) - \frac{\bar{T}^{n+1} - \bar{T}^n}{t^{n+1} - t^n}\left[1 - \beta\frac{(t^{n+2} - t^n)}{t^{n+2} - t^{n+1}}\right] = 0 \tag{100}$$

For $t^* = t^{n+2}$,

$$t^{n+2} - t^{n+1} = (1 - \beta)\,(t^{n+2} - t^n)\ , \tag{101}$$

and Eq. (100) becomes

$$\frac{\bar{T}^{n+2} - \bar{T}^{n+1}}{t^{n+2} - t^{n+1}} + \frac{\beta}{(1-\beta)}\,(\tilde{X}^{n+1} + \tilde{Y}^{n+2}) - \frac{\bar{T}^{n+1} - \bar{T}^n}{t^{n+1} - t^n}\left(\frac{1 - 2\beta}{1 - \beta}\right) = 0\ . \tag{102}$$

Equation (99) is identical in form to Eq. (19) and, for $\beta = 1/2$, Eq. (102) is identical in form to Eq. (20). Equations (99) and (102) are equivalent to the Douglas–Gunn Eqs. (94) and (95) for linear $\bar{T}$ and $\tilde{X}$ and for $t^* = t^{n+2}$.

The symmetrical form of Eqs. (19) and (20) has been pointed out previously. However, Eqs. (99) and (102) are symmetrical only if $\beta = 1/2$. Intuitively, it seems that the Douglas–Gunn algorithm for $\beta \neq 1/2$ might lead to possibly unacceptable asymmetries in symmetrical problems, e.g., a spherical expansion in r–z coordinates. When it was decided to give ANIMAL the capability for time-differencing with $\beta \neq 1/2$, a symmetrical two-step algorithm was derived.

Equation (99) is a consistent approximation to the original differential equation, Eq. (19). Hence, the Douglas–Gunn equations can be applied to $\bar{U}^{n+1}$ to calculate $\bar{U}^{n+2}$ and $\bar{U}^{n+3}$. The appropriate difference equations are

$$\frac{\bar{T}^{n+2} - \bar{T}^{n+1}}{t^{n+2} - t^{n+1}} + \tilde{X}^{n+1} + \tilde{Y}^{n+2} = 0 \tag{103}$$

and

$$\frac{\bar{T}^{n+3} - \bar{T}^{n+2}}{t^{n+3} - t^{n+2}} + \frac{\beta}{(1-\beta)}\,(\tilde{X}^{n+3} + \tilde{Y}^{n+2}) - \frac{\bar{T}^{n+2} - \bar{T}^{n+1}}{t^{n+2} - t^{n+1}}\left(\frac{1 - 2\beta}{1 - \beta}\right)\ . \tag{104}$$

Equations (103) and (104) still have the same type of asymmetry as Eqs. (99) and (102), except that the two dimensions are interchanged. However, a symmetrical set of equations can be derived by averaging Eq. (99) with Eq. (104) for $t^n \rightarrow t^{n+1}$ and by averaging Eqs. (102) and (103). The resultant set of equations is

$$(2 - 2\beta)\,\frac{\bar{T}^{n+1} - \bar{T}^n}{t^{n+1} - t^n} + \tilde{X}^{n+1} + \tilde{Y}^n - (1 - 2\beta)\,\frac{\bar{T}^n - \bar{T}^{n-1}}{t^n - t^{n-1}} = 0 \tag{105}$$

and

$$(2 - 2\beta)\,\frac{\bar{T}^{n+2} - \bar{T}^{n+1}}{t^{n+2} - t^{n+1}} + \tilde{X}^{n+1} + \tilde{Y}^{n+2} - (1 - 2\beta)\,\frac{\bar{T}^{n+1} - \bar{T}^n}{t^{n+1} - t^n} = 0\ . \tag{106}$$

Equations (105) and (106) have the same symmetry as Eqs. (19) and (20) and reduce to Eqs. (19) and (20) if $\beta = 1/2$. Equation (105) can be rewritten as

$$\frac{\bar{T}^{n+1} - \bar{T}^n}{t^{n+1} - t^n} + \tilde{X}^{n+1} + \tilde{Y}^n + (1 - 2\beta)\left[\frac{\bar{T}^{n+1} - \bar{T}^n}{t^{n+1} - t^n} - \frac{\bar{T}^n - \bar{T}^{n-1}}{t^n - t^{n-1}}\right] = 0\ . \tag{107}$$

Equation (107) is identical to Eq. (19) except for the last term. The last term of Eq. (107) is an approximation to $(1 - 2\beta)\,\Delta t\,(\partial^2\vec{T}/\partial t^2)$.

Equations (105) and (106) have been implemented in ANIMAL. As implemented $\vec{T}$, $\vec{X}$ and $\vec{Y}$ are nonlinear, even though Eqs. (105) and (106) were derived from Eqs. (94) and (95) with an assumption of linearity. Implementation of Eqs. (105) and (106) was relatively trivial. To implement the first terms of Eqs. (105) and (106), the timestep was merely divided by $2 - 2\beta$ when the coefficients for time derivatives were computed. The terms involving $\vec{X}$ and $\vec{Y}$ are identical to Eqs. (19) and (20), and no changes involving these complicated spatial-derivative terms were necessary. The last terms are "known," or explicit, quantities that are computed directly and included in the $\vec{V}$'s of Eqs. (21) and (32).

For some one-dimensional problems, it appears that some of the inaccuracies encountered in the velocity fractional-step method are not encountered using $\beta > 1/2$. Note that Eqs. (105) and (106) guarantee the conservation of mass, momentum, and magnetic flux, but an additional nonconservation of energy of order $\Delta t\,(1 - 2\beta)$ is introduced. At this writing, the usefulness of using $\beta > 1/2$ has not been fully evaluated.

8. SPATIAL-DIFFERENCE EQUATIONS

It may, or may not be, apparent to the reader that a complete difference equation for any of the model equations, or any simplifications of them, has yet to be written in this document. Even Eqs. (40) and (41) are not complete, since, for example, the method of evaluating the interface quantity $\left[\rho\left(v_l - v_l^G\right)\right]^{n+1}_{j+1/2,k}$ has yet to be specified. This author hopes, at this time, that the reader understands the general formalism about how implicit quantities such as $\rho\left[\left(v_i - v_i^G\right)\right]^{n+1}_{j,k}$ are implemented in the code as a set of coefficients as implied by Eqs. (42) to (64). This author also hopes that the reader understands the symmetries between the two coordinates ξ_1 and ξ_3, both in the model equations and in the numerical algorithm.

The purpose of Section 4 was to formulate an algorithm in which only one interface implicitly treated and only one interface explicitly treated need be considered. When these two are properly considered the calculations necessary for all interior interfaces are automatically taken into account. (As previously indicated, boundary interfaces must be considered separately.) In this section it is only necessary to specify the exact form of the fluxes and forces associated with the, say, j+1/2,k interface. The reader should be able to figure out the appropriate Newton–Raphson coefficients and also be able to extend the form to the j,k+1/2 interface, and so forth, so that the complete difference equation can be written down if so desired. Note that it is *never* necessary to actually write down the complete difference equation for implementation of new fluxes, forces, or difference methods in ANIMAL. Consequently, a complete difference equation is never written out in all its gory detail in this document; Eq. (40) is as close to a complete difference equation as this author ever cares to write.

In specifying the forces and fluxes, several quantities need be defined. First of all,

$$(\xi_1)_j = \frac{(\xi_1)_{j-1/2} + (\xi_1)_{j+1/2}}{2} \tag{108}$$

and

$$(\xi_3)_k = \frac{(\xi_3)_{k-1/2} + (\xi_3)_{k+1/2}}{2} , \tag{109}$$

which implies that the locations of the zone-centers are derived from the locations of zone-interfaces. The zone-centers are always located half-way between the interfaces, but when nonuniform zoning is used, the interfaces are not located half-way between the centers. The following geometric quantities are relevant:

$$(\Delta\xi_1)_j = (\xi_1)_{j+1/2} - (\xi_1)_{j-1/2} , \tag{110}$$

$$(\Delta\xi_3)_k = (\xi_3)_{k+1/2} - (\xi_3)_{k-1/2} , \tag{111}$$

$$(\Delta\xi_1)_{j+1/2} = (\xi_1)_{j+1} - (\xi_1)_j , \tag{112}$$

and

$$(\Delta\xi_3)_{k+1/2} = (\xi_3)_{k+1} - (\xi_3)_k \quad , \tag{113}$$

where the first two quantities are zone dimensions and the latter two are distances between zone-centers. Fluxes and nonlocal forces are expressed in terms of the average, difference, and sign of any quantity Q, defined as

$$\bar{Q}_{j+1/2,k} = \frac{Q_{j+1} + Q_j}{2} \quad , \tag{114}$$

$$\delta Q_{j+1/2,k} = \frac{Q_{j+1} - Q_j}{2} \quad , \tag{115}$$

$$s(\bar{Q}_{j+1/2,k}) = \begin{cases} 1 & \text{if } \bar{Q}_{j+1/2,k} \geq 0 \\ -1 & \text{if } \bar{Q}_{j+1/2,k} < 0 \end{cases} \quad , \tag{116}$$

and

$$s(\delta Q_{j+1/2,k}) = \begin{cases} 1 & \text{if } \delta Q_{j+1/2,k} \geq 0 \\ -1 & \text{if } \delta Q_{j-1/2,k} < 0 \end{cases} \quad . \tag{117}$$

In the notation of Eq. (114), an overline refers to an average, not a vector as in previous sections.

The mass fluxes of Eq. (5) are expressed by

$$\int h_2 h_3 x_{33} \rho \left(v_1 - v_1^G\right) d\xi_3 \approx (\bar{f}_M)_+ = S_+ \left(\bar{v}_1^R\right)_+ \left\{ \bar{\rho}_+ - s \left[\left(\bar{v}_1^R\right)_+\right] s(\delta\rho_+) (\delta\rho_+)^2 / \bar{\rho}_+ \right\} \quad , \tag{118}$$

where the subscript $j+1/2,k$ has been replaced by a + for convenience and the quantity S_+ is defined by Eq. (35). The second term in Eq. (118) is a second-order (in $\Delta\xi$) "mass diffusion" term, which is included to reduce "numerical dispersion" and to assure "positivity." The use of a second-order mass diffusion of the form given in Eq. (118) is a unique feature of ANIMAL and minimizes the numerical diffusion traditionally associated with Eulerian codes. At the first interior interfaces, $j = \frac{5}{2}$, $J - \frac{3}{2}$, $k = \frac{5}{2}$, $K - \frac{3}{2}$, the form used is

$$\int h_2 h_3 x_{33} \rho \left(v_1 - v_1^G\right) d\xi_3 \approx (\bar{f}_M)_+ = S_+ \left(\bar{v}_1^R\right)_+ \left\{ \bar{\rho}_+ - s \left[\left(\bar{v}_1^R\right)_+\right] \delta\rho_+ \right\} \quad , \tag{119}$$

which is a first-order (in $\Delta\xi$) donor cell, or upwind, convection introduced to prevent clearly nonphysical effects in the vicinity of boundaries.

For momentum convection in Eqs. (6) and (7), ANIMAL uses

$$\int h_2 h_3 x_{33} \rho v_1 \left(v_1 - v_1^G\right) d\xi_3 \approx (\bar{f}_M)_+ \left[\left(\bar{v}_1^R\right)_+ + \left(\bar{v}_1^G\right)_+\right] \tag{120}$$

and

$$\int h_2 h_3 x_{33} \rho v_3 \left(v_1 - v_1^G\right) d\xi_3 \approx (\bar{f}_M)_+ \left[\left(\bar{v}_3^R\right)_+ + \left(\bar{v}_3^G\right)_+\right] \quad , \tag{121}$$

where $(\bar{f}_m)_+$ is defined by Eq. (118) or (119). The form given in Eqs. (120) and (121) assures that a subconservation property is maintained.[3]

The internal-energy convection of Eq. (8) in ANIMAL is written as

$$\int h_2 h_3 x_{33}\, \rho\epsilon \left(v_1 - v_1^G\right) d\xi_3 \approx (\bar{f}_M)_+ \left\{ \bar{\epsilon}_+ - s \left[\left(\bar{v}_1^R\right)_+\right] s\,(\delta\epsilon_+)\,(\delta\epsilon_+)^2 \,/\, \bar{\epsilon}_+ \right\} , \qquad (122)$$

where a second-order (in $\Delta\xi$) "diffusion" term is added to maintain "positivity." At the first interior interfaces,

$$\int h_2 h_3 x_{33}\, \rho\epsilon \left(v_1 - v_1^G\right) d\xi_3 \approx (\bar{f}_M)_+ \left\{ \bar{\epsilon}_+ - s \left[\left(\bar{v}_1^R\right)_+\right] \delta\epsilon_+ \right\} . \qquad (123)$$

The magnetic flux convection of Eq. (9) is written as

$$\int h_3 x_{33}\, B_2 \left(v_1 - v_1^G\right) d\xi_3 \approx D_+ \left(\bar{v}_1^R\right)_+ \left[\left(\overline{h_2 B_2^P}\right)_+ + \left(\overline{h_2 B_2^E}\right)_+\right] , \qquad (124)$$

where D_+ is defined in Eq. (30). Equation (124), coupled with the magnetic force term to be given in Eq. (126), guarantees the maintenance of a subconservation property.[3]

The pressure-gradient nonlocal force term in Eq. (6) is written as

$$\text{“}\frac{1}{2}\text{”} \iint h_2 h_3 x_{33} \frac{\partial p}{\partial \xi_1} d\xi_1\, d\xi_3 \approx S_+ \delta p_+ , \qquad (125)$$

where the factor “$\frac{1}{2}$” implies a corresponding symmetric contribution from the j−1/2,k interface.

The magnetic nonlocal force of Eq. (6) is

$$\text{“}\frac{1}{2}\text{”} \frac{1}{\mu_0} \iint h_3 x_{33} B_2 \frac{\partial}{\partial \xi_1} (h_2 B_2)\, d\xi_1\, d\xi_3 \approx D_+ \left[\left(\overline{h_2 B_2^P}\right)_+ + \left(\overline{h_2 B_2^E}\right)_+\right] \delta \left(h_2 B_2^P\right)_+ . \qquad (126)$$

From Eq. (41), one might be tempted to multiply Eqs. (125) and (126) by $\Delta\xi_j/\Delta\xi_{j+1/2}$ when nonuniform zoning is used. However, doing so would destroy the appropriate subconservation property. In ANIMAL, as should be apparent from Eqs. (118) to (126), there is no attempt in the fluxes and forces to weight quantities at j and j+1 in a manner related to their distances from the j+1/2 interface, which are unequal when nonuniform zoning is used. On the nonuniformly zoned shock-tube test problem suggested by Le Blanc,[32] ANIMAL displays inaccuracies similar to those displayed by various Lagrangian and Eulerian codes.

The local "force" terms $h_2\rho v_3 v_1 h_{13} x_{11}$ and $h_2\rho v_3^2 h_{31} x_{33}$ of Eq. (6) are evaluated at a cell center merely by multiplying by $(\Delta\xi_1)_j(\Delta\xi_3)_k$ to take into account the volume integration and using the obvious cell values. Both terms are included in X of Eqs. (19) and (20); i.e., they are treated implicitly for $t^n \rightarrow t^{n+1}$ and explicitly for $t^{n+1} \rightarrow t^{n+2}$.

The pressure-gradient and magnetic nonlocal forces and the local forces of Eq. (7) are treated analogously to those of Eq. (6) as described above.

The compressional-work nonlocal force term of Eq. (8) can be rewritten as

$$P \frac{\partial}{\partial \xi_1} (h_2 h_3 x_{33} v_1) = \frac{\partial}{\partial \xi_1} (h_2 h_3 x_{33} v_1 p) - h_2 h_3 x_{33} v_1 \frac{\partial p}{\partial \xi_1} . \qquad (127)$$

The first term on the right side of Eq. (127) leads to a flux and the last term leads to a nonlocal force. The difference equations for the latter and former are

$$\text{“}\frac{1}{2}\text{”} \iint h_2 h_3 x_{33} v_1 \frac{\partial p}{\partial \xi_1} d\xi_1\, d\xi_3 \approx S_+ \left[\left(\bar{v}_1^R\right)_+ + \left(\bar{v}_1^G\right)_+\right] \delta p_+ \qquad (128)$$

and

$$\int h_2 h_3 x_{33} v_1 p \, d\xi_3 \approx S_+ \left\{ \left[\left(\bar{v}_1^R \right)_+ + \left(\bar{v}_1^G \right)_+ \right] \bar{p}_+ + C_p \left[\delta \left(v_1^R \right)_+ + \delta \left(v_1^G \right)_+ \right] \delta p_+ \right\} , \tag{129}$$

where

$$C_p = \begin{cases} C_d \text{ if } \left[\left(\bar{v}_1^R \right) + \left(\bar{v}_1^G \right) \right]_+ \delta p_+ < 0 \\ 0 \text{ otherwise} \end{cases} \tag{130}$$

and C_d is a constant that is normally unity but can be altered at execution time. The term of Eq. (130) involving C_p is a second-order energy flux added to partially offset the effect of Eq. (128) at a contact discontinuity at the leading edge of a rarefaction, e.g., the Riemann shock-tube problem suggested by Trigger.[33]

The thermal-conduction flux of Eq. (8) is approximated by

$$\int \frac{h_2 h_3 x_{33}}{h_1 x_{11}} K \frac{\partial T}{\partial \xi_1} d\xi_3 \approx \frac{2 S_+ \bar{K}_+ \delta T_+}{(\overline{h_1 x_{11}})_+ (\Delta \xi_1)_+} , \tag{131}$$

where the factor of 2 is required because of the definition of δT_+ as given by Eq. (115); note that a simple average is used for the thermal conductivity.

Similarly, the resistive-diffusion flux of Eq. (9) is given by

$$\frac{1}{\mu_0} \int \frac{h_3 x_{33}}{h_2 h_1 x_{11}} \eta \frac{\partial \left(h_2 B_2^P \right)}{\partial \xi_1} d\xi_3 \approx \frac{2 D_+ \bar{\eta}_+ \delta \left(h_2 B_2^P \right)_+}{\mu_0 (\overline{h_1 x_{11}})_+ (\Delta \xi_1)_+} \tag{132}$$

and the corresponding ohmic-heating nonlocal force in Eq. (8) is

$$\text{“}\frac{1}{2}\text{”} \iint \frac{h_3 x_{33}}{\mu_0^2 h_1 h_2 x_{11}} \eta \left[\frac{\partial}{\partial \xi_1} \left(h_2 B_2^P \right) \right]^2 d\xi_1 \, d\xi_3 \approx (\bar{f}_\eta)_+ =$$

$$\frac{2 D_+ \bar{\eta}_+ \delta \left(h_2 B_2^P \right)_+ \delta \left(h_2 B_2^P \right)_+}{\mu_0^2 (\overline{h_1 x_{11}})_+ (\Delta \xi_1)_+} . \tag{133}$$

As written in Eq. (133), an equal amount of energy is deposited in the two zones on either side of an interface as a result of a gradient in $h_2 B_2^P$ between the two zones. If the densities of the two zones adjacent to the interface are considerably different, then the resulting temperature change (or, more correctly, specific internal energy change) is also considerably different. Early in the development of ANIMAL it was decided to require that ohmic heating lead to an equal temperature change rather than an equal energy change; this is accomplished by adding a second-order energy flux, i.e.,

$$-\int \frac{1}{4} \frac{\Delta \xi_1 \eta h_3 x_{33}}{\mu_0^2 h_1 h_2 x_{11} \rho} \frac{\partial \rho}{\partial \xi_1} \left[\frac{\partial}{\partial \xi_1} \left(h_2 B_2^P \right) \right]^2 d\xi_3 \approx - (\bar{f}_\eta)_+ \delta \rho_+ / \bar{\rho}_+ , \tag{134}$$

where $(\bar{f}_\eta)_+$ is defined by Eq. (133).

Since Eq. (134) is an internal energy flux, it does not alter the subconservation properties of Eqs. (132) and (133). At this time, it is not clear whether or not the use of Eq. (134) is necessary.

As indicated in Section 2, the "transverse" thermoelectric effect is incorporated into the code. The difference equation for the "transverse" thermoelectric flux in Eq. (9) is

$$\int \frac{h_3 x_{33}\,\beta}{h_1 x_{11}} \frac{|B_2|}{B_2} \frac{\partial T}{\partial \xi_1}\, d\xi_3 \approx \frac{2D_+ (\overline{h_2})_+ \delta T_+ \left(\overline{\beta s (B_2^P + B_2^E)}\right)_+}{(\overline{h_1 x_{11}})_+ (\Delta\xi_1)_+} \tag{135}$$

and the corresponding *force* in Eq. (8) is

$$\text{“}\frac{1}{2}\text{”} \iint \frac{h_3 \beta X_{33}}{h_1 x_{11} \mu_0} \frac{|B_2|}{B_2} \frac{\partial T}{\partial \xi_1} \frac{\partial (h_2 B_2)}{\partial \xi_1}\, d\xi_1\, d\xi_3 \approx$$

$$\frac{2D_+ (\overline{h_2})_+ \delta T_+ \left[\overline{s\left(B_2^P + B_2^E\right)\beta}\right]_+ \delta\left(h_2 B_2^P\right)_+}{\mu_0 (\overline{h_1 x_{11}})_+ (\Delta\xi_1)_+} \tag{136}$$

The "transverse" thermoelectric energy *flux*, which appears in Eq. (8), is

$$\int \frac{h_3 x_{33}\,\beta\, T}{\mu_0 h_1 x_{11}} \frac{|B_2|}{B_2} \frac{\partial}{\partial \xi_1}\left(h_2 B_2^P\right) d\xi_3 \approx$$

$$\frac{2D_+ (\overline{h_2})_+ \overline{T}_+ \left[\overline{s\left(B_2^P + B_2^E\right)\beta}\right]_+ \delta\left(h_2 B_2^P\right)_+}{\mu_0 (\overline{h_1 x_{11}})_+ (\Delta\xi_1)_+} \tag{137}$$

The radiative-energy-loss local force of Eq. (8) is incorporated simply by using zone-center values and multiplying by $\Delta\xi_1\ \Delta\xi_3$. The radiative loss is always treated implicitly.

To this point, spatial-difference equations for all terms appearing in the model Eqs. (5) to (9) have been presented. In the absence of shocks and steep gradients, the difference equations as presented so far are adequate. However, to treat shocks and to help minimize certain pathologies, it is generally necessary to incorporate an *artificial viscosity*. In ANIMAL, artificial viscosities are incorporated by added *fluxes* to the equations of motion and adding *forces* to the internal energy equation to account for the kinetic energy dissipated. The artificial viscosities included in ANIMAL can be considered to be difference approximations to partial differential equations of the form

$$\frac{\partial}{\partial t}(h_1 h_2 h_3 \rho v_1) + \ldots + \frac{\partial}{\partial \xi_1}(h_2 h_3 x_{33} q_{11}) + \frac{\partial}{\partial \xi_3}(h_1 h_2 x_{11} q_{13}) = 0\ , \tag{138}$$

$$\frac{\partial}{\partial t}(h_1 h_2 h_3 \rho v_3) + \ldots + \frac{\partial}{\partial \xi_1}(h_2 h_3 x_{33} q_{31}) + \frac{\partial}{\partial \xi_3}(h_1 h_2 x_{11} q_{33}) = 0\ , \tag{139}$$

and

$$\frac{\partial}{\partial t}(h_1 h_2 h_3 \rho \epsilon) + \ldots + h_2 h_3 x_{33} q_{11} \frac{\partial v_1}{\partial \xi_1} + h_1 h_2 x_{11} q_{13} \frac{\partial v_1}{\partial \xi_3}$$

$$+ h_2 h_3 x_{33} q_{31} \frac{\partial v_3}{\partial \xi_1} + h_1 h_2 x_{11} q_{33} \frac{\partial v_3}{\partial \xi_3} = 0\ . \tag{140}$$

Since all the added terms in Eqs. (136) and (137) are in flux form, they do not represent merely a modification to the pressure-gradient terms. Nor do the added terms include all terms resulting from taking the divergence of a pressure tensor in orthogonal curvilinear coordinates. Furthermore, in general, $q_{31} \neq q_{13}$. So the added terms are not symmetric as would be the components of a pressure tensor. Thus, it is hard to make a precise physical analogy between Eqs. (138) to (140) and real viscous effects, except that the terms do dissipate kinetic energy and do increase entropy, just as real viscous effects do.

It is perhaps better to consider the terms of Eqs. (138), (139), and (140) as truncation-error modifications. From a truncation-error point of view, if the various q's are formally of $0(\Delta\xi)^m$, with m greater than zero, then the complete difference equations are still consistent approximations to the differential Eqs. (5) to (9).

Because of the symmetry of Eqs. (138) to (140) with respect to the coordinates ξ_1 and ξ_3, it is sufficient merely to consider the fluxes and forces associated with the j+1/2,k interface, just as has been done throughout this section. Thus one can write the momentum fluxes as

$$\int h_2 h_3 x_{33} q_{11} \, d\xi_3 \approx S_+ (\bar{q}_{11})_+ \tag{141}$$

and

$$\int h_2 h_3 x_{33} q_{31} \, d\xi_3 \approx S_+ (\bar{q}_{31})_+ \tag{142}$$

and the energy forces as

$$\text{“}\frac{1}{2}\text{”} \iint h_2 h_3 x_{33} q_{11} \frac{\partial v_1}{\partial \xi_1} \, d\xi_1 \, d\xi_3 \approx S_+ (\bar{q}_{11})_+ \delta \left(v_1^R + v_1^G \right)_+ \tag{143}$$

and

$$\text{“}\frac{1}{2}\text{”} \iint h_2 h_3 x_{33} q_{31} \frac{\partial v_3}{\partial \xi_1} \, d\xi_1 \, d\xi_3 \approx S_+ (\bar{q}_{31})_+ \delta \left(v_3^R + v_3^G \right)_+ \, . \tag{144}$$

The shock heating indicated by Eqs. (143) and (144) can give steep temperature gradients if the density gradient between zones is steep, so an energy *flux* is also added:

$$-\int \frac{h_2 h_3 x_{33}}{4\rho} \Delta\xi_1 \left(q_{11} \frac{\partial v_1}{\partial \xi_1} + q_{13} \frac{\partial v_3}{\partial \xi_1} \right) \frac{\partial \rho}{\partial \xi_1} d\xi_3 \approx$$

$$- \delta\rho_+ S_+ \left[(\bar{q}_{11})_+ \delta \left(v_1^R + v_1^G \right)_+ + (\bar{q}_{13})_+ \delta \left(v_3^R + v_3^G \right)_+ \right] \Big/ \bar{\rho}_+ \, . \tag{145}$$

For shocks moving from a high-density region into a low-density region, use of Eq. (145) may not be wise since the energy should be deposited in the low-density region.

Artificial viscosities of several different forms have been incorporated into ANIMAL. Thus,

$$(\bar{q}_{11})_+ = \bar{p}_+ \delta \left(v_1^R + v_1^G \right)_+ \left[(\bar{q}_{11}^a)_+ + (\bar{q}_{11}^c)_+ + (\bar{q}_{11}^e)_+ \right] + (\bar{q}_{11}^b)_+ \, . \tag{146}$$

The terms of Eq. (146) are given by

$$\left(\bar{q}_{11}^a \right)_+ = \begin{cases} C_q^a \delta \left(v_1^R + v_1^G \right)_+ \, , & \text{if } \delta \left(v_1^R + v_1^G \right)_+ < 0 \\ 0, \text{ otherwise} \end{cases} \, , \tag{147}$$

$$\left(\bar{q}^{c}_{11}\right)_{+} = - C^{c}_{q} \left(t^{n+1} - t^{n}\right) s\left(\delta\rho_{+}\right) \delta\rho_{+} \left[\bar{P}_{+} + D_{+} \left[\left(\overline{h_2 B^{P}_{2}}\right) + \left(\overline{h_2 B^{E}_{2}}\right)^{+} \right]^{2} / S_{+} \right] \Big/ \left[\left(\overline{h_1 x_{11}}\right)_{+} \Delta\xi_1 \left(\bar{\rho}_{+}\right)^{2} \right] , \tag{148}$$

$$\left(\bar{q}^{e}_{11}\right)_{+} = \begin{cases} - s\left(\left(\bar{f}_{M}\right)_{+}\right) \left(\bar{f}_{M}\right)_{+} / \left(\bar{\rho}_{+} S_{+}\right) \text{ if } j = 5/2, J - 3/2 \\ 0, \text{ otherwise} \end{cases} \tag{149}$$

where C^{a}_{q} and C^{c}_{q} are constants, normally 8 and 0, respectively, which can be specified at execution time. In Eq. (149), $\bar{q}^{a}_{11}$ is similar to the standard von Neumann–Richtmyer artificial viscosity,[2] and, for Cartesian coordinates, it is in fact identical. Considered in combination with Eq. (120), Eq. (149), which uses $\bar{f}_{M}$ as defined in Eq. (118), leads to first-order "upwind" convection at the first interior interface.

As defined in Eq. (148), $\bar{q}^{c}_{11}$ is a *magnetic-field-dependent artificial viscosity* introduced in an attempt to minimize pathologies. The difference equations for the magnetic and pressure forces, as given in Eqs. (125) and (126), deposit an equal amount of *momentum* in each zone adjacent to an interface. When there is a large density variation across the interface, a large velocity gradient can be introduced. Even when the velocity gradient introduced is negative, Eq. (146) is often insufficient if the thermal and magnetic energy densities are much greater than the kinetic energy density. Note that $\bar{q}^{c}_{11}$ depends on the density gradient. Note also that the first factor in brackets is essentially the magnetoacoustic velocity.

A more straightforward approach to the problem of forces across a steep density gradient is to deposit momentum in the two adjacent zones proportional to the density of the zone. This can be accomplished for the pressure gradient by specifying $\bar{q}^{b}_{11}$ in Eq. (146) to be

$$\bar{q}^{b}_{11} = -C^{b}_{q} \delta\rho_{+} \delta p_{+} / \bar{\rho}_{+} , \tag{150}$$

where C^{b}_{q} is a constant, normally 0, which can be set at execution. From Eq. (144) one can see that q^{b}_{11} does not always dissipate kinetic energy. A modified momentum flux dependent on the magnetic force must also be used in conjunction with Eq. (150). The modified flux is

$$\bar{r}^{b}_{11} = -C^{b}_{q} \delta\rho_{+} D_{+} \left[\left(\overline{h_2 B^{P}_{2}}\right)_{+} + \left(\overline{h_2 B^{E}_{2}}\right)_{+} \right] \delta\left(h_2 B^{P}_{2}\right)_{+} / \bar{\rho}_{+} . \tag{151}$$

When incorporating Eqs. (150) and (151) it was decided to interpret them as modified forces. A modified pressure gradient should modify the internal energy, as indicated by Eq. (144). However, a modified magnetic force should lead to a modification of the magnetic energy, not the internal energy. Since Eq. (151) can be interpreted as an approximation to

$$\frac{\partial}{\partial t} \left(h_1 h_2 h_3 x_{11} x_{33} \rho v_1\right) + \ldots + \frac{\partial}{\partial \xi_1} \left(r_{11}\right) = 0 , \tag{152}$$

an equation of the form

$$\frac{\partial}{\partial t} \left(h_1 h_3 B_2\right) + \ldots - \frac{\partial}{\partial \xi_1} \left[r_{11} \frac{\partial v_1}{\partial \xi_1} \Big/ \frac{\partial\left(h_2 B_2\right)}{\partial \xi_1} \right] = 0 \tag{153}$$

is required to ensure energy conservation. Hence, to the difference equation for Eq. (9) must be added a flux

$$-\int \left[r_{11} \frac{\partial v_1}{\partial \xi_1} \Big/ \frac{\partial\left(h_2 B_2\right)}{\partial \xi_1} \right] d\xi_3 \approx C^{b}_{q} \delta\rho_{+} D_{+} \left[\left(\overline{h_2 B^{P}_{2}}\right)_{+} + \left(\overline{h_2 B^{E}_{2}}\right)_{+} \right] \delta\left(v^{I}_{1} + v^{G}_{1}\right)_{+} / \bar{\rho}_{+} . \tag{154}$$

This author has not run many problems using Eqs. (150) to (154); some have been satisfactory and some have been obviously unsatisfactory. At this writing, this author recommends that C_q^b always be set to zero. It is possible, however, that the basic idea of modifying the magnetic convection flux (rather than an internal energy force) when a modified magnetic force is used is a valid one and should be pursued further.

Analogous to Eq. (146), the "shear" viscosity is written as

$$(\bar{q}_{31})_+ = \bar{\rho}_+ \delta \left(v_3^R + v_3^G\right)_+ \left(\bar{q}_{31}^a + \bar{q}_{31}^c + \bar{q}_{31}^e\right) , \tag{155}$$

where, generally, $\bar{q}_{31}^a = 0$, $\bar{q}_{31}^c = 0$, and

$$\bar{q}_{31}^e = \begin{cases} -s \left((\bar{\Gamma}_M)_+\right) (\bar{f}_M)_+ \Big/ (\bar{\rho}_+ S_+) \\ 0, \text{ otherwise} \end{cases} . \tag{156}$$

In earlier versions, $\bar{q}_{31}^a = q_{11}^a$ and $\bar{q}_{31}^c = \bar{q}_{11}^c$ were used, and it has yet to be resolved which method is most satisfactory. In Eq. (156), the first form is used when the velocity fractional-step method is used since Eq. (155) and the momentum convective term corresponding to Eq. (121) are always treated "explicitly," as indicated in Eq. (92).

The previous paragraph completes the specification of the fluxes and forces associated with interior zones. Boundary fluxes and forces at the j = J−1/2 boundary have essentially the same form if averages and differences are redefined as

$$\bar{Q}_{J-1/2,k} = Q_{J,k} \tag{157}$$

and

$$\delta Q_{J-1/2,k} = Q_{J,k} - Q_{J-1,k} . \tag{158}$$

The upwind forms for convection given in Eqs. (119), (123), (149), and (156) are used if $(v_1^R)_J > 0$. If $(v_1^R)_J \neq 0$, the forces have the form given in Eqs. (125) and (126). Otherwise the forces are zero (see Ref. 4). The compressional work is as given by Eqs. (128) and (129), except the term involving C_P is neglected. The heat flux, resistive diffusion, and ohmic heating is as given in Eqs. (131) to (133), respectively. The thermoelectric effect is treated as given in Eqs. (135) to (137). Only the viscosity $\bar{q}_{11}^q$, given in Eq. (147), is used. The boundary at the j = 3/2 interface is treated similarly, with the average and difference redefined as

$$\bar{Q}_{3/2,k} = Q_{1,k} \tag{159}$$

and

$$\delta Q_{3/2,k} = Q_{2,k} - Q_{1,k} . \tag{160}$$

9. IMPLEMENTATION OF THE ALGORITHM—INTRODUCTORY REMARKS

In the previous section of this document, the mathematical algorithm used in the ANIMAL code has been described. Implementation of the algorithm into an actual working computer code has required a major computer programming effort. The code is written for operation on the CDC-7600 computer. Most of the computer code has been written in FORTRAN, but some sections that consume a disproportionate fraction of the total computational time have been reprogrammed in the ASCENTF assembly language. The combined FORTRAN/ASCENTF source code is compiled by the CDC PUTT compiler, a simple, one-pass compiler introduced to LLL by a "third world" movement dissatisfied with the CHAT compiler developed at LLL. Since ANIMAL's predecessor[17] was originally developed for

a CDC-3400 computer, which used a CDC compiler, when ANIMAL was first implemented at LLL it was felt that use of the PUTT compiler would facilitate the conversion. Originally, it was anticipated that eventually ANIMAL would be compiled with CHAT. However, even after ANIMAL has been in existence for over six years, there is no obvious reason for attempting to use the CHAT compiler. Tests have indicated that only insignificant decreases in execution time could be made with CHAT using all of its optimization features.[34] On the other hand, it is clear that the CHAT compiler is much more cumbersome to use for the many compiles and recompiles necessary during code development. This author has found the PUTT compiler and its associated library and debug routines more than adequate, and, in fact, this author feels progress in the development of ANIMAL would not have been as rapid had the more accepted CHAT system been used. Recently, however, tests have indicated that the CDC FTN compiler will lead to a significantly faster executing code with an acceptable increase in compilation time.[35] It currently appears that conversion of ANIMAL to an FTN version will be advantageous when FTN-associated libraries and debug routines reach the same sort of sophistication and convenience currently available through PUTT.

The most recent operable version of ANIMAL is ANMAL07. The corresponding source, MALAD07, consists of roughly 13,000 FORTRAN lines. Consequently, a complete code description would be a monstrous undertaking. The intent of the following sections is to describe the important variables and subroutines in MALAD07 (the description is not entirely applicable to previous versions). The following sections are not by themselves, very useful; for the following sections to be completely useful, the reader must have available a source listing, an ANIMAL user's manual, and the previous sections of this report. A copy of the ANIMAL user's manual and a copy of the ANIMAL post-processor's user's manual are included on microfiche on the inside back cover of this report. However, a source listing is available only from the author by direct request.

To understand a section of the code, the user will want to refer to this report. However, a particular variable may not be described here. In such a case, this author recommends that the user use LLL's TRIX AC to "TS" the variable throughout the source listing to determine how it is defined in terms of variables described here.

The following sections will not describe outdated, superseded, or experimental coding in ANIMAL. Since the ANIMAL source listing has never really been cleaned up, there is inevitably unused or superseded coding still carried along. In addition, since ANIMAL is by no means a static code, there is within the source listing experimental coding for various techniques not yet proved and documented here or elsewhere.

In ANIMAL there are slightly less than 90 subroutines. These subroutines can generally be divided into several different classes. INITIALIZATION subroutines read input data that define a particular problem and then do various operations such as constant evaluation, table generation, and similar tasks that must be accomplished before the time-marching algorithm can be initiated. MISCELLANEOUS LOGIC AND CONTROL subroutines do various tasks that generally do not fit any of the other categories described here. PREPARATORY subroutines do a variety of manipulation and transfer tasks to put problem data in a usable form for the subroutines that define the coefficients. COEFFICIENT subroutines actually define the coefficients appearing in the linearized difference equations, Eqs. (21) and (32), and the linearized boundary equations, Eqs. (22) to (25). TRI-DIAGONAL SOLVER subroutines solve the linearized difference equations by doing the forward–backward sweeps implied by Eqs. (26) and (33). OUTPUT subroutines do the unfortunately many necessary chores to accurately provide BINARY output data on tape or disc for restart and post-processing and do the chores required to put a selected quantity of computed results into user-readable ASCII. MARKER PARTICLE subroutines initialize and "push" marker, or tracer, particles that follow in a Lagrangian manner the motion of fluid elements based on the velocity field calculated by the Eulerian code. DIAGNOSTIC subroutines are used for debugging purposes and perform a variety of functions including the checks indicated in Eqs. (66) to (70).

When additional physical effects or dependent variables are added to the code, the major changes to the coding occur in the COEFFICIENT subroutines. Relatively minor changes are made to a few INITIALIZATION and PREPARATORY subroutines, mainly those directly involving physics, but for the most part all subroutines except the COEFFICIENT routines remain essentially intact.

INITIALIZATION subroutines are

STARTUP	INITI
MATRIX	DTINIT
TCINIT	CKTINIT
EOSINIT	RBC
EPFROMT	SPL
MESH	FINDLW

MISCELLANEOUS LOGIC AND CONTROL subroutines are

XPAND	BNDRYSW
SPLINT	SETBDRY
CKTSET	ITCON
GRIDMOV	MAXVAR
SWTCH13	DTCNTRL
SET27PT	OUTK
INITPTR	ENDMAL
SETPTR	VOLINT1

PREPARATORY subroutines are

NEXTHHP	MXTZGR
NEXTRBBP	NXTZFLX
NEXTK	SETRGR
NEXTTC	SETSGN
EQNST	SETBC
TRANCO	SETRP

COEFFICIENT subroutines are

BCRZ	RFLX
ZFLX	RBCRFLX
UBCZFLX	LBCRFLX
DBCZFLX	MAT2

TRI-DIAGONAL SOLVER subroutines are

MAIN program
SUMBCE
TRIANG

OUTPUT subroutines are

RUNDATA	BUFFO
OUTPUT1	BSPVAR
PLOTR	SETDEN
NUMZBZ	STATUS
SCALFAC	WRBLNK
BBEXT	ENDFIL
ORDMES	IFAR
WOFORD	NEWTPE
GETIOC	BIDARZ
BODARZ	BIVAR
BOVAR	BUFFIN

MARKER PARTICLE subroutines are

MPINIT	INTERP1
MARKER	INTERP2
SEARCH	MPOUT
GETDXDT	

DIAGNOSTIC subroutines are

CHECK	CHKRBCR
SPECHK	CHKRFLX
CHKLBCR	

Prior to a description of the subroutine functions, the basic control variables and basic arrays in the coding are introduced and the allocation of LCM (large core memory) is described.

10. BASIC CONTROL VARIABLES

NDIM— DA(1); the *N*umber of *DIM*ensions of the problem; has values 1 or 2.

NV— DA(16); the *N*umber of dependent *V*ariables; has values 3, 4, or 5 depending on which set of variables is calculated; code can handle a maximum NV = 9.

NDR— DA(9); the number of mesh points in the ξ_1 direction; corresponds to the value J used in the text; note that the number of zones in the ξ_1 direction is NDR-2.

NDZ— DA(10); the number of mesh points in the ξ_3 direction; corresponds to the value K used in the text; note that the number of zones in the ξ_3 direction is NDZ-2.

IIV— DA(21); a ten-digit number that designates which dependent variables are being computed; each digit corresponds, in order, to a component of the "total" solution vector

$$\bar{U}^T = \left(\rho,\ v_1^R,\ v_2^R,\ v_3^R,\ \epsilon_e,\ \epsilon_i,\ B_1^P,\ B_2^P,\ B_3^P\right) \quad ; \tag{161}$$

if the digit that corresponds to a particular component is less than or equal to NV, then that variable is being computed; e.g., for $\bar{U} = \rho, v_1^R, \epsilon_e$, NV = 3 and IIV = 1245637890, so the first, second, and sixth digit are $\leq$ NV; FORTRAN names corresponding to the components of $\bar{U}^T$ are RO, V1, V2, V3, EE, EI, B1, B2, B3, respectively.

IIVV— DA(22); a ten-digit number that designates which dependent variable of the total solution vector corresponds to a particular component of the actual solution vector; the first NV digits correspond to each component of the actual solution vector; e.g., for $\bar{U} = (\rho, v_1, \epsilon_e)$, IIVV = 126345790; i.e., the third component of the solution vector $\bar{U}$ corresponds to the sixth component of the total solution vector $\bar{U}^T$.

IFSV— DA(8); has values 0 and 1; if 1, the velocity fractional-step method is used.

IBE— DA(24); if 1, magnetic field is broken up into plasma and "external" components as indicated by Eq. (78); otherwise $B_2^E = 0$ and $B_2^P = B_2$.

MOVGRID— DA(126); if 1, the moving-grid option is used and the velocity is broken up into two components as indicated by Eqs. (76) and (77); otherwise, $v_1^G = v_3^G = 0$ and $v_1^R = v_1$, $v_3^R = v_3$.

ICOORD— DA(124); designates the orthogonal coordinate system (ξ_1, ξ_3) being used: if −1, Cartesian; if 0, standard (r, z) cylindrical coordinates; if 1, spherical coordinates; if 2, toroidal coordinates; if 3, cylindrical (r, ϕ) coordinates.

NMP— DA(116); the number of marker particles used.

ISBC— DA(200); if 1, split-boundary conditions are used.

ISTEP— designates which direction is treated implicitly; if 1 or 3, the ξ_1 direction is treated implicitly; if 2 or 4, the ξ_3 direction is treated implicitly; the values 3 and 4 are not used; the values 3 and 4 were intended to solve the difference equations by a "backward–forward" method rather than the "forward–backward" methods implied by Eqs. (26) and (33).

LINSEQ— *LIN*e *SEQ*uence; designates the sequence in which the implicitly treated lines are computed; if 1, k-lines are computed in the order k = 2, 3, 4 . . . K-1 for ISTEP = 1 and j-lines are computed in the order j = 2,3,4 . . . J-1 for ISTEP = 2; if 2, k-lines are computed in the order k = K-1, K-2, K-3 . . . 4, 3, 2 for ISTEP = 1 and j-lines are computed in the order j = J-1, J-2, J-3 . . . 4, 3, 2 for ISTEP = 2; LINSEQ = 2 was introduced in anticipation of treating mixed second derivatives of the form $\frac{\partial^2}{\partial\xi_1\,\partial\xi_3}$.

MDSEQ— *M*ixed *D*erivative *SEQ*uence; controls LINSEQ for each pair of time steps; values for LINSEQ are tabulated below:

	MDSEQ =	1	2	3	4
ISTEP = 1		1	2	1	2
ISTEP = 2		1	2	2	1

MDSEQ was introduced in anticipation of treating mixed second derivatives of the form $\frac{\partial^2}{\partial\xi_1\,\partial\xi_3}$.

NP— the number of points along a k- or j-line; for ISTEP = 1, NP = NDR and for ISTEP = 2, NP = NDZ.

IPNT— timestep number at which to make next ASCII output.

TTOEDIT— time at which to make next ASCII output.

NWREC— timestep number at which to make next BINARY output.

TTODUMP—time at which to make next BINARY output.

TIME— DADT(1); the real, i.e., problem, time since the problem was begun.

NNDT— DADT(2); the number of timesteps required for the problem to reach a certain TIME.

DT— DADT(11); the timestep being used to advance the calculations.

DTOLD— DADT(10); the timestep size used to advance the calculations from the previous timestep NNDT-1 to the present.

JBCRR— DA(69); a ten-digit number specifying the boundary conditions on the j = 1(3/2) boundary; each digit, in order, corresponds to a component of the "total" solution vector $\bar{U}^T$; if split boundaries are used, JBCRR applies to the upper k-values.

JBCR— DA(6); a ten-digit number specifying the boundary conditions on the j = K(K-1/2) boundary; each digit, in order, corresponds to a component of the "total" solution vector $\bar{U}^T$; if split boundaries are used, JBCR applies to the upper k-values.

JBCZZ— DA(68); a ten-digit number specifying the boundary conditions on the k = 1(3/2) boundary; each digit, in order, corresponds to a component of the "total" solution vector $\bar{U}^T$; if split boundaries are used, JBCZZ applies to the upper j-values.

JBCZ— DA(70); a ten-digit number specifying the boundary conditions on the k = K(K-1/2) boundary; each digit, in order, corresponds to a component of the "total" solution vector $\bar{U}^T$; if split boundaries are used, JBCZ applies to the upper j-values.

JSBCRR— DA(192); a ten-digit number specifying the boundary conditions on the j = 1(3/2) boundary for lower k-values when split boundaries are used.

JSBCR— DA(193); a ten-digit number specifying the boundary conditions on the j = J(J-1/2) boundary for lower k-values when split boundaries are used.

JSBCZZ— DA(194); a ten-digit number specifying the boundary conditions on the k = 1(3/2) boundary for lower j-values when split boundaries are used.

JSBCZ— DA(195); a ten-digit number specifying the boundary conditions on the k = K(K-1/2) boundary for lower j-values when split boundaries are used.

NTC— the number of transport coefficients plus the number of derivatives of transport coefficients with respect to the dependent variables; this number is currently 19; the list of 19 is

$$\bar{K} = \left(\theta,\ \frac{\partial\theta}{\partial\epsilon},\ T,\ \frac{\partial T}{\partial\epsilon},\ f_i,\ \frac{\partial f_i}{\partial\epsilon},\ \eta,\ K,\ \frac{\partial\eta}{\partial\epsilon},\ \frac{\partial K}{\partial\epsilon},\ G,\ G_L,\ G_B,\right.$$

$$\left.\beta,\ \frac{\partial K}{\partial B},\ \frac{\partial K}{\partial\rho},\ \frac{\partial\eta}{\partial B},\ \frac{\partial\eta}{\partial\rho},\ \frac{\partial G}{\partial\epsilon}\right) , \qquad (162)$$

where $\bar{K}$ is a transport coefficient vector, θ is the pressure p divided by the density ρ, i.e., $\theta = p/\rho$, and G_L and G_B are the Lyman alpha-line and Balmer alpha-line emission factors used only for tally purposes; FORTRAN names corresponding to the components of $\bar{K}$ are: THET, DTHDE, TA, DTADE, FION, DFIDE, RESP, XKEP, DRSDE, DKEDE, GF, GLA, GHA, BETAT, DKEDB, DKEDR, DRSDB, DRSDR, DGFDE, respectively.

11. BASIC ARRAYS

DA— *DA*ta array used to store various information that does not change as a problem progresses; besides the basic control words described in the previous section, DA includes two 40-word subarrays, ADDA (*AD*ditional *DA*ta values) and IDA (additional data value descriptor words), which begin at DA(26) and DA(71), respectively, and into which are packed various problem-dependent data words subsequently printed in the ASCII output. The number of words used in ADDA and IDA is given by NADDA[DA(20)].

DADT— *DA*ta array used to store various information that changes with each timestep as a problem progresses; besides the basic control words described in the previous section, this array includes circuit parameters and energy tallies, such as

DADT(18)—total circuit-load magnetic flux.
DADT(19)—circuit-load voltage.
DADT(20)—electrical energy delivered to load by circuit.
DADT(21)—dL/dt.
DADT(22)—the circuit-load inductance L.
DADT(23)—the circuit source voltage.
DADT(24)—the circuit current at the advanced time.
DADT(25)—the circuit current at the present time.
DADT(26)—the circuit current at the previous timestep.
DADT(13)—the total heat loss to the surrounding walls.
DADT(15)—the total radiation loss.
DADT(91)—the total pdV heating.
DADT(92)—the total shock heating.
DADT(93)—the total ohmic heating.
DADT(94)—the total Balmer alpha-line radiation loss.
DADT(95)—the total Lyman alpha-line radiation loss.
DADT(98)—the "cutoff" density ρ_{co} (see Ref. 4).
DADT(100)—the grid velocity $\left(v_1^G\right)_{j,k}$ of the j = J(J-1/2) interface for all k lines.

In addition, DADT includes two 10-word subarrays VVARM1 and VVARM2, which begin at DADT(51) and DADT(61), respectively, in which are stored, at time t^n, the maximum fractional variation of the dependent variables when advancing from t^{n-1} to t^n and t^{n-2} to t^{n-1}, respectively. VVARM1 and VVARM2 are used in conjunction with the timestep control.

IVV, IV, IBCRR, IBCR, IBCZZ, IBCZ, ISBCRR, ISBCR, ISBCZZ, ISBCZ—ten-word arrays, each word of which has the same value as the corresponding digit of IIVV, IIV, JBCRR, JBCR, JBCZZ, JBCZ, JSBCRR, JSBCR, JSBCZZ, JSBCZ, respectively, and which are used for purposes similar to the purposes of the words from which they are derived.

A, B, C, V— arrays corresponding to the "coefficients" $\bar{\bar{A}}_1$, $\bar{\bar{B}}_1$, $\bar{\bar{C}}_1$, and $\bar{\bar{V}}_1$ of Eq. (21) and $\bar{\bar{A}}_3$, $\bar{\bar{B}}_3$, $\bar{\bar{C}}_3$, and $\bar{\bar{V}}_3$, of Eq. (32).

KCOM27— a 10 × 10 array of subscripts for the matrix arrays A, B, and C, which are used by the COEFFICIENT subroutines to assure that a computed coefficient is stored in the proper location. In the coefficient routines, the elements of KCOM27 are named

LRORO, LROV1, LROV2, LROV3, LROEE, LROEI, LROB1, LROB2, LROB3, LRODUM,
LV1RO, LV1V1, LV1V2, LV1V3, LV1EE, LV1EI, LV1B1, LV1B2, LV1B3, LV1DUM,
LV2RO, LV2V1 . . . ,
LV3RO, LV3V1 . . . ,
LEERO, LEEV1 . . . ,
LEIRO, LEIV1 . . . ,
LB1RO, . . . ,
LB2RO, . . . ,
LB3RO, . . . , etc.

For ISTEP = 1, a coefficient with the subscript LROV1 is a coefficient of the v_1^R velocity component (V1) in the density (RO) equation, i.e., the continuity equation. Similarly, for ISTEP = 1, a coefficient with the subscript LV1RO is a coefficient of the density (RO) in the v_1^R velocity-component (V1) equation. On the other hand, for ISTEP = 2, the subscript LROV1 refers to a coefficient of the v_3^R velocity component (V3) in the density (RO) equation. It is by proper definition of KCOM27 that ANIMAL is able to use the same coding for implicit fluxes in either the ξ_1 direction or the ξ_3 direction. If the capability to compute the B1 magnetic field component were being added to the code, coefficients with subscripts LB1RO, LB1V1 . . . , etc. would have to be defined. If the solution vector was $\bar{U} = (\rho, \epsilon, B_2)$, so that 3 × 3 matrices were used and V_1^R was not computed, LROV1 and LV1RO would have values greater than 9 = NV * NV.

LCOM27— an array of ten subscripts for the vector array V used by the COEFFICIENT subroutines to assure that a computed vector component is stored in the proper location. In the COEFFICIENT routines, the elements of LCOM27 are named LRO, LV1, LV2, LV3, LEE, LE1, LB1, LB2, LB3, LDUM. For ISTEP = 1, a coefficient vector component with subscript LV1 refers to a quantity included in the v_1^R component of the vectors $\bar{V}_1$ and $\bar{V}_3$ of Eqs. (21) and (32), respectively. On the other hand, for ISTEP = 2, the subscript LV1 refers to the v_3^R component of the vectors $\bar{V}_1$ and $\bar{V}_3$.

RZ— mesh array giving position of grid interfaces at t^n; RZ(M) is position of interface at j = M+1/2, i.e., $(\xi_1)_{M+1/2}$; RZ(NDR + M) is position of interface at k = M+1/2, i.e., $(\xi_3)_{M+1/2}$.

RZP— mesh array giving position of grid interfaces at t^{n+1}.

RR— mesh array giving ξ_1 location of grid (zone-center) values at time t^n, i.e., RR(M) = $(\xi_1)_M$.

RRP— mesh array giving ξ_1 location of grid (zone-center) values at time t^{n+1}.

ZZ— mesh array giving ξ_3 location of grid (zone-center) values, i.e., ZZ(M) = $(\xi_3)_M$; no corresponding ZZP is required since at this writing the grid does not move in the ξ_3 direction.

SINZ— mesh array in which values for the sine of the angles ξ_3 are stored for spherical, toroidal, and cylindrical r-ϕ coordinates.

VAR— the COMMON/I9/ variable name; VAR is the main small core memory (SCM) working storage array; the function of VAR is determined by various pointers and the particular part of the algorithm the code is executing.

12. LCM MEMORY ALLOCATION

Large core memory on the CDC-7600 is used for bulk storage. ANIMAL does not do any random accessing of LCM. All transfers of data between LCM and SCM are done by PUTT's block copy instructions SMALLIN and SMALLOUT. ANIMAL's LCM array is named simply XLCM and prior to execution has a length of 2. By issuing appropriate system calls, XLCM is expanded as required at execution time. Information is stored in LCM locations determined by the pointers described in this section. Every word of XLCM is used, so the total length of XLCM is the minimum required for execution. This allows the code to time-share as much as possible. The total length of XLCM is problem-dependent. Most of LCM memory allocation is done in subroutine MESH. The LCM pointers and the length of the block of memory they point to are

NTIME— length = NV * NDR * NDZ; all the dependent variables at all the mesh points at the time t^n when advancing from t^n to t^{n+1}; the ith component of the solution vector U at the (j, k) point is given by

$$(U_i)^n_{j,k} = \text{XLCM (NTIME + I + NV * (J - 1 + NDR * (K - 1)))} \tag{163}$$

NP1TIME— length = NV * NDR * NDZ; the block in which variables at time t^{n+1} are stored when an iteration along a k-line (for ISTEP = 1; a j-line for ISTEP = 2) has been completed; indexing is same as NTIME.

NM1TIME— length = NV * NDR * NDZ; all the dependent variables at all the mesh points at time t^{n-1} when advancing from t^n to t^{n+1}; if the calculation is successfully advanced to t^{n+1}, the values of the pointers NM1TIME, NTIME, NP1TIME are rotated to the values of NTIME, NP1TIME, NM1TIME, respectively; if it is necessary to recycle, the pointer NTIME is set to NM1TIME.

MPLCM— length = 2 * NMP; ξ_1 and ξ_3 coordinates of marker particles at time t^n.

MPLCMP— length = 2 * NMP; new coordinates of marker particles at time t^{n+1}.

MPLCMO— length = 2 * NMP; marker-particle coordinates at time t^{n-1}; when calculations advance, MPLCMO, MPLCM, MPLCMP rotate to MPLCM, MPLCMP, MPLCMO, respectively; when recycling, MPLCM is set to MPLCMO.

NAEOST— length = NEOST; the complete equation-of-state tables; consider an equation-of-state vector $\bar{S} = \left(\epsilon, \frac{\partial^2 f_i}{\partial \epsilon^2}, \frac{\partial^2 f_d}{\partial \epsilon^2}, f_i, f_d\right)$; a collection of sets of $\bar{S}$ for a series of density values stored in an array EOSRO are generated and stored in LCM.

NZERO— length = 1000; an array of zero's block-copied into SCM to zero out SCM arrays.

KKMMLCM—length = problem dependent; all the scale factors, external magnetic field values, grid velocity values, transport coefficients, and dependent variables for all j-values, $1 \le j \le J$, for $k = k' - 2$ when working along a line $k = k'$ (for ISTEP = 1; for ISTEP = 2, all k-values $1 \le k \le K$, for $j = j' - 2$ working along a line $j = j'$).

KKMLCM— same as KKMMLCM except at $k = k' - 1$.

KKLCM— same as KKMMLCM except at $k = k'$.

KKPLCM— same as KKMMLCM except at $k = k' + 1$.

KKPPLCM— used instead of KKMMLCM to store values at $k = k' + 2$ when LINSEQ = 2.

LCMKPZF— length = 2 * NV * NDR (for ISTEP = 1; length = 2 * NV * NDZ for ISTEP = 2); the explicit fluxes and forces associated with the $k = k' + 1/2$ ($j = j' + 1/2$) interface.

LCMKMZF— length = 2 * NV * NDR (for ISTEP = 1; length = 2 * NV * NDZ for ISTEP = 2); the explicit fluxes and forces associated with the $k = k' - 1/2$ ($j = j' - 1/2$) interface.

13. INITIALIZATION SUBROUTINES

General comments. ANIMAL begins execution in one of two modes, new problem generation or restart of a problem partially completed. Under restart, not all data cards necessary for generation are needed, so the flow path in each INITIALIZATION subroutine is different depending on the mode of execution. Most control information for restart is stored in a BINARY dump of the DA array; much of the information is obtained by searching the IDA and ADDA subarrays. Upon generation, coding in the INITIALIZATION subroutines involving the variables NADDA, IDA, and ADDA is intended to provide restart capability as well as ASCII output information.

STARTUP— controls the basic initialization process, whether for new-problem generation or for restart; reads file-name data, if any, from the execution line and then reads the OUTPUT card (see ANIMAL user's manual for input-card description); if a new problem is being generated, also reads the ID card.

MATRIX— establishes the solution vector $\bar{U}$; reads PHYSICS card and sets ICOORD. NV, DIM, IIV, IIVV, and arrays IV, IVV accordingly.

TCINIT— *INIT*ializes *T*ransport *C*oefficients; evaluates constants appearing in algebraic expressions for thermal conductivities, resistivity, and radiative-loss term; sets up "total emission factor" [$G(\rho,T)$ of Eq. (14)] tables; the coding associated with array variables TCMULT (*T*ransport *C*oefficient *MULT*iplier) and TCCON (*T*ransport *C*oefficient *CON*stant) implements the various multiplier/constant/zero options provided on the PHYSICS card; coding associated with array variables JDW and IDW loads the data word arrays IDA and ADDA to provide ASCII description of options used, etc.; reads any constant or multiplier data required by the options used; important variables are

UO— free-space permeability.

EO— free-space permittivity.

PLANCK—Planck's constant.

EMASS— ion mass.
ECHG— electronic charge.
TQ— temperature above which quantum effects become important in evaluation of Spitzer[27] Coulomb logarithm.
CNETE— constant appearing in Braginskii[24] expression for $n_e\tau_e$.
CNITI— constant appearing in Braginskii[24] expression for $n_i\tau_i$.
CWETE— constant appearing in Braginskii[24] expression for $\omega_e\tau_e$.
CWITI— constant appearing in Braginskii[24] expression for $\omega_i\tau_i$.
CLAM1— constant appearing in Spitzer[27] expression for Coulomb logarithm.
CRES2— constant C_3 of Eq. (16) giving functional form of neutral resistivity.
CRES1— constant appearing in Braginskii[24] expression for resistivity of fully ionized gas.
CKEP1— constant appearing in Braginskii[24] expression for electron thermal conductivity perpendicular to a magnetic field.
CKIP1— constant appearing in Braginskii[24] expression for ion thermal conductivity perpendicular to a magnetic field.
CTTE1— constant appearing in Braginskii[24] expression for transverse thermoelectric-effect coefficient.
CKN1— constant appearing in Stevens[25] expression for neutral-particle thermal conduction.
CLAM2— arbitrary constant added to argument of Coulomb logarithm to prevent argument from being less than 1.
CBREM— constant C_2 of Eq. (14) giving expression for radiative loss.
CSAHA— constant appearing in C_1 of Eq. (11).
GFLA— array; Lyman alpha-line emission table.
GFHA— array; Balmer alpha-line emission table.
GFF— array; total emission factor [$G(\rho,T)$ of Eq. (14)] table.
TRAD— array; temperature table corresponding to GFHA, GFLA, GFF.
D2GFDT— array; second derivative of ℓn(GFF) used in spline interpolation for GFF.

EOSINIT— *INIT*ializes *E*quation *O*f *S*tate tables if the default ideal gas EOS is not being used; creates a table for f_i and f_d [see Eq. (10)] as a function of density ρ and *specific internal energy* ϵ [in ANIMAL, ϵ and ρ are treated as the two independent thermodynamic variables, so $T = T(\rho, \epsilon)$; using ϵ instead of T makes the treatment of energy conservation and the treatment of the "phase change," which occurs during dissociation and ionization, more accurate]; for every density ρ = EOSRO (NRO), the EOS tables have NTAB(NRO) values of ϵ; for a particular density, values are stored in the EOS tables in the order . . .

$$\ell,\ \left(\frac{\partial^2 f_i}{\partial\epsilon^2}\right)_\ell,\ \left(\frac{\partial^2 f_d}{\partial\epsilon^2}\right)_\ell,\ (f_i)_\ell,\ (f_d)_\ell,\ \epsilon_{\ell+1},$$

$$\left(\frac{\partial^2 f_i}{\partial\epsilon^2}\right)_{\ell+1},\ \left(\frac{\partial^2 f_d}{\partial\epsilon^2}\right)_{\ell+1},\ (f_i)_{\ell+1},\ (f_d)_{\ell+1},\ \epsilon_{\ell+2}\ \cdots;$$

second derivatives of f_i and f_d are used with spline interpolation; the tables for ρ = EOSRO(NRO) begin at address NATAB(NRO); the tables are generated by starting at a low temperature and incrementing by DTEMP until f_d or f_i vary by a prescribed amount.

EPFROMT—used in conjunction with EOSINIT to give a specific internal energy ϵ = EI and the corresponding f_i = FION and f_d = FDIS for specified values of T and ρ = RO; a "reduced ionization potential" that depends on f_i is used, so an iterative method is used to calculate FION.

MESH— generates the finite-difference grid and establishes the LCM memory allocation; reads all GRID cards; sets up RZ, RZP, RR, RRP, ZZ, SINZ arrays; if a moving grid is used, reads "r vs t" tables; variables relevant to moving grid are

- X1MAX— table of r values.
- AC1T— table of time values corresponding to X1MAX.
- AC1— table of $\frac{\partial^2 r}{\partial t^2}$ values used with spline interpolation.
- NACC1— number of values in the X1MAX, AC1T, and AC1 tables.

INIT1— generates initial conditions for solution vector; basic function is to establish values for the NV *NDR*NDZ LCM words, which begin at pointer NTIME; reads INITIAL CONDITION cards, including perturbation cards, as required; if problem is a "link," searches binary records from earlier problem to get appropriate "link" values; "commented" coding is for a "rezoner," which is not completely operative due to memory allocation changes made after the rezoner was developed.

DTINIT— *INIT*ializes timestep *DT* controls; reads TIMESTEP CONTROL card.

CKTINIT— *INIT*ializes circuit *CKT* calculations; reads circuit cards including voltage or current tables as required; table variables are

- ZI— array; current or voltage table.
- ZITYM— time array corresponding to ZI.
- D2ZI— second derivative with respect to time used in spline interpolation.

RBC— initializes boundary conditions; reads all BOUNDARY CONDITION cards; sets up arrays ISBCR, ISBCZ, ISBCRR, ISBCZZ, IBCR, IBCZ, IBCRR, and IBCZZ; reads INSULATING WALL card; reads MAGNETIC FIELD BOUNDARY VALUE cards as required; table variables are

- ZI— magnetic field or current tables.
- ZITYM— time tables corresponding to ZI.
- D2ZI— second-derivative values used in spline interpolation.

SPL— generates second-derivative tables necessary for cubic spline interpolation.

FINDLW— *FIND*s *L*ast *W*rite on a BINARY record disc file or tape during restart.

14. MISCELLANEOUS LOGIC AND CONTROL SUBROUTINES

XPAND— e*XPANDS* length of LCM by issuing appropriate system calls.

SPLINT— performs a *SPL*ine *INT*erpolation.

CKTSET— advances the circuit equations forward in time; basic function is, given current I^n (ZIN) at time t^n, to calculate current I^{n+1} (ZINP1); integrates B_2 over entire grid to find total FLUX in grid; the circuit-calculation method depends on which source is actually used, as specified on CIRCUIT data cards; because two alternately used finite-difference schemes [Eqs. (21) and (32)] are used, ANIMAL actually does a circuit calculation every two timesteps. The ANIMAL circuit equations assume that the circuit load, i.e., the plasma region, is entirely surrounded by an "ideal" conductor except for an insulated "feed-through" slot between the "terminals" of the load. If the small ohmic voltage drop in the external conductors is neglected, the voltage appearing at the "terminals" of the apparatus can be written, using Faraday's law, as

$$V = \int_A^B \bar{E}\cdot d\bar{\ell} = -\int_B^A \bar{E}\cdot d\bar{\ell} = -\oint_C \bar{E}\cdot d\bar{\ell} = \iint_S \frac{\partial \bar{B}}{\partial t}\cdot d\bar{S} = \frac{d\Phi}{dt}\ , \tag{164}$$

where

$$\Phi = \iint_S \bar{B}\cdot d\bar{S}\ . \tag{165}$$

In Eq. (164), the first two integrals mean along a "path-independent" open path between the "terminals" A and B, which are external to the apparatus. The third integral of Eq. (164) is along a closed path, which includes the open path of the first two integrals and which also connects the terminals A and B by a path passing around the plasma region. The equating of the second integral to the third is valid if the ohmic voltage drop in the external conductors is neglected. Inclusion of the copper drop is straightforward. On the other hand, if the plasma region is not surrounded by conductor, such as in a direct z-pinch, a term dependent on plasma resistivities and current densities must also be included. The third integral in Eq. (164) is equated to the fourth integral in accordance with Faraday's law. Note that the total magnetic flux ϕ, as defined in Eq. (165), includes the magnetic flux in both the plasma region and the surrounding insulation. The load inductance of the apparatus is defined to be

$$L = \Phi / I \ ; \tag{166}$$

and, therefore, using Eq. (164),

$$V = L \frac{dI}{dt} + I \frac{dL}{dt} \ , \tag{167}$$

the usual relation for a time-varying inductance. For the capacitive source shown in Fig. 3, the circuit calculation is performed using the difference equations

$$V_C^n = V_C^{n-2} - (t^n - t^{n-2}) \frac{I_1^n + I_2^n}{C} \ , \tag{168}$$

$$V_C^n = R_E (I_2^n + I_1^n) + L_E \frac{I_1^{n+2} + I_2^{n+2} - I_1^n - I_2^n}{t^{n+2} - t^n} + L_S \frac{I_1^{n+2} - I_1^n}{t^{n+2} - t^n} + V_L^n \ , \tag{169}$$

and

$$R_T I_2^n + L_T \frac{I_2^{n+2} - I_2^n}{t^{n+2} - t^n} = L_S \frac{I_1^{n+2} - I_1^n}{t^{n+2} - t^n} + V_L^n \ , \tag{170}$$

where V_L is the voltage across L. Ideally, one would like to represent the voltage V_L, using Eq. (166), as

$$V_L^n = \frac{L^{n+2} I_1^{n+2} - L^n I_1^n}{t^{n+2} - t^n} \ . \tag{171}$$

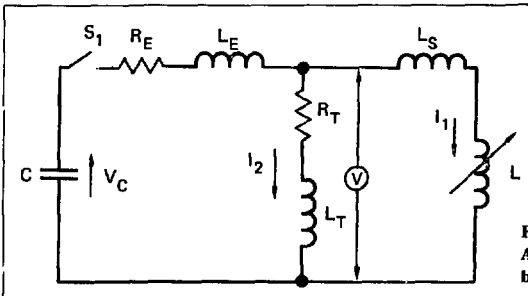


FIG. 3. The electrical circuit used in ANIMAL to provide magnetic field boundary conditions.

However, L^{n+2} can be determined from the plasma calculation only if I_1^{n+2} is known; i.e., Eq. (171) is nonlinear in the unknown, I_1^{n+2}. To avoid iterating between the plasma and circuit calculations, the following two forms have been used in ANIMAL:

$$V_L^n = L^n \frac{I_1^{n+2} - I_1^n}{t^{n+2} - t^n} + I_1^{n+2} \frac{L^n - L^{n-2}}{t^n - t^{n-2}} \tag{172}$$

and

$$V_L^n = L_V \frac{I_1^{n+2} - I_1^n}{t^{n+2} - t^n} + \frac{\phi_p^n - \phi_p^{n-2}}{t^n - t^{n-2}} \quad . \tag{173}$$

In Eq. (173), L_V is the vacuum, plasma-free inductance of the region of solution and ϕ_p is the magnetic flux due to plasma currents. ϕ_p is given by

$$\phi_p = \iint_S \bar{B}^p \cdot d\bar{S} \quad . \tag{174}$$

Of course, finite-difference approximations are used to evaluate the integrals in Eqs. (165) and (174), which are used to define the inductance L in accordance with Eq. (166). Originally, Eq. (172) was used, but in calculations in which the current approached zero, the inductance L became unmanageably large [see Eq. (166)]. In addition, during reverse-bias pinches, the inductance became unmanageably small as the total load flux passed through zero. Currently, it is believed that the use of Eq. (173) is superior. With the voltage V_L specified by Eq. (172) or (173), Eq. (168) is used to calculate V_C^n and then the linear, simultaneous Eqs. (169) and (170) are used to calculate I_1^{n+2} and I_2^{n+2}. With I_1^{n+2} determined, the plasma calculation can proceed.

GRIDMOV—determines $\left(v_i^G\right)^{n+1}$ and arrays RZP and RRP when a moving grid is used; assumes that the grid uniformly expands or contracts in the ξ_1 direction only.

SWTCH13—*SWiTCH*es ξ_1 and ξ_3 directions when ISTEP = 2; this is accomplished by interchanging appropriate values of the IVV, IV, IBCRR, IBCR, IBCZZ, IBCZ, ISBCRR, ISBCR, ISBCZZ, ISBCZ, and ISBCK arrays, which are located in COMMON/28/ (note that there are fixed arrays having the same names in COMMON/4/).

SET27PT—sets the LCOM27 and KCOM27 arrays using the IV and IVV arrays.

INITPTR—*INiT*ializes SCM *PoinTeR*s described below; called at the beginning of a timestep.

Along a particular k- or j-line, the scale factors $(h_1)^n$, $(h_1)^{n+1}$, $(h_2)^n$, $(h_2)^{n+1}$, $(h_3)^n$, $(h_3)^{n+1}$, the grid velocities $(v_1^G)^n$, $(v_1^G)^{n+1}$, $(v_3^G)^n$, $(V_3^G)^{n+1}$, the "external" magnetic field components $(B_1^E)^n$, $(B_1^E)^{n+1}$, $(B_2^E)^n$, $(B_2^E)^{n+1}$, $(B_3^E)^n$, $(B_3^E)^{n+1}$, and the transport "coefficients" and derivatives are needed in addition to the dependent variables $\bar{U}^n$. These quantities are computed line by line as required and then transferred as a block to the LCM locations KKMMLCM, KKMLCM, KKLCM, KKPLCM, etc. When brought back into the VAR array of SCM, these quantities are located by pointers:

KH1— for ISTEP = 1, $(h_1)_{j,k}^n$ along a k-line KK is located at VAR(KH1 + J); similarly, for ISTEP = 2, $(h_3)_{k,j}^n$ along a j-line KK is located at VAR(KH1 + J).

KH1P— pointer for $(h_1)_{j,k}^{n+1}$ or $(h_3)_{k,j}^{n+1}$, etc.

KH2— $(h_2)_{j,k}^n$ or $(h_2)_{k,j}^n$.

KH2P— $(h_2)_{j,k}^{n+1}$ or $(h_2)_{k,j}^{n+1}$.

KH3— $(h_3)_{j,k}^n$ or $(h_1)_{k,j}^n$.

KH3P— $(h_3)^{n+1}_{j,k}$ or $(h_1)^{n+1}_{k,j}$.

KVG1— $(v_1^G)^n_{j,k}$ or $(v_3^G)^n_{k,j}$.

KVG1P— $(v_1^G)^{n+1}_{j,k}$ or $(v_3^G)^{n+1}_{k,j}$.

KVG3— $(v_3^G)^n_{j,k}$ or $(v_1^G)^n_{k,j}$.

KVG3P— $(v_3^G)^{n+1}_{j,k}$ or $(v_3^G)^{n+1}_{k,j}$.

KB1E— $(B_1^E)^n_{j,k}$ or $(B_3^E)^n_{k,j}$.

KB1EP— $(B_1^E)^{n+1}_{j,k}$ or $(B_3^E)^{n+1}_{k,j}$.

KB2E— $(B_2^E)^n_{j,k}$ or $(B_2^E)^n_{k,j}$.

KB2EP— $(B_2^E)^{n+1}_{j,k}$ or $(B_2^E)^{n+1}_{k,j}$.

KB3E— $(B_3^E)^n_{j,k}$ or $(B_1^E)^n_{k,j}$.

KB3EP— $(B_3^E)^{n+1}_{j,k}$ or $(B_1^E)^{n+1}_{k,j}$.

KTC— pointer to the transport coefficients; for ISTEP = 1, the I-component of the transport-coefficient vector $\bar{K}^n_{j,k}$ of Eq. (162) is located at VAR(KTC + NTC * (J − 1) + I); similarly, for ISTEP = 2, the I-component of $\bar{K}^n_{k,j}$ is also located at VAR(KTC + NTC * (J − 1) + I).

KVAR— pointer to the dependent variables along a k- or j-line; the I-component of the dependent-variable vector $\bar{U}^n_{j,k}$ or $\bar{U}^n_{k,j}$ is located at VAR(KVAR + NV * (J − 1) + I).

KPH1, KPH1P, . . . KPB3EP, KPTC, KPVAR—pointers corresponding to KH1, KH1P, . . . KB3EP, KTC, KVAR for time t^n along k- or j-line KKP = KK + 1.

KMH1, KMH1P, . . . KMB3EP. KMTC, KMVAR—pointers corresponding to KH1, KH1P, . . . KB3EP, KTC, KVAR for time t^n along k- or j-line KKM = KK − 1.

NXTH1, NXTH1P, . . . NXTB3EP, NXTTC, NXTVAR—pointers corresponding to KH1, KH1P, . . . KB3EP, KTC, KVAR for time t^n along the *next* k- or j-line required; i.e., NXTK = KKP = KK + 1 if LINSEQ = 1, and NXTK = KKM = KK − 1 if LINSEQ = 2.

KZH1, KZH1P, . . . KZB3EP, KZTC, KZVAR—pointers corresponding to KH1, KH1P, . . . KB3EP, KTC, KVAR for time t^n for quantities associated with the k- or j-line KKZ and which are to be used to calculate the explicit fluxes across the KKZ + 1/2 interfaces; for LINSEQ = 1, KKZ = KK, and for LINSEP = 2, KKZ = KKM = KK − 1.

KPZH1, KPZH1P, . . . KPZB3EP, KPZTC, KPZVAR—pointers corresponding to KZH1, KZH1P, . . . KZB3EP, KZTC, KZVAR for quantities associated with the k- or j-line KKPZ = KKZ + 1 and which are to be used to calculate the explicit fluxes across the KKPZ − 1/2 = KKZ + 1/2 interfaces.

KRH1, KRH1P, . . . KRB3EP, KRTC, KRVAR—pointers corresponding to KH1, KH1P, . . . KB3EP, KTC, KVAR for time t^{n+1}, corresponding to the iterate along a k- or j-line KKR = KK; for $\ell = 0$, KRH1 = KH1, KRH1P = KH1P . . . for $\ell \neq 0$, KRH1 = KRH1P = KH1P; KRTC refers to the transport-coefficient vector $\bar{K} = \bar{K}(\bar{U}^{n+1,\ell})$ and KRVAR refers to $\bar{U}^{n+1,\ell}$.

Note that pointers for B_1^E and B_3^E are included even though ANIMAL cannot currently handle the field components B_1 and B_3. This is an example showing that much of the formalism for handling B_1 and B_3 is still retained in the code.
Other SCM pointers are

NAU— the pointer for the "new" variables $\bar{U}^{n+1,\,\ell+1}$ along a k- or j-line KK; the I-component is located at VAR(NAU + NV * (J − 1) + I).

NAE— the pointer for the matrices $\bar{\bar{E}}$ of Eqs. (22) to (26) and (33); for ISTEP = 1, the NV * NV elements of the matrix $\bar{\bar{E}}^{n+1,\ell}_{j,k}$ are located beginning at VAR (NAE + NV * NV * (J − 1) + 1).

NAF— the pointer for the vector $\bar{F}$ of Eqs. (22) to (26) and (33); for ISTEP = 1, the NV elements of the vector $\bar{F}^{n+1,\ell}_{j,k}$ are located beginning at VAR(NAF + NV * (J − 1) + 1).

LOCADD— the pointer for the NRGR = 10 grid quantities associated with zone-centers and implicitly treated interfaces; for ISTEP = 1, the NRGR quantities are

$$\bar{R}^n_{j,k} = \left[V^n_{j,k},\ V^{n+1}_{j,k},\ (h_2L)^n_{j,k},\ (h_2L)^{n+1}_{j,k},\ (\Delta\xi_1)_j\ (\Delta\xi_3)_k,\right.$$

$$\left.(h_{13})^n_k,\ (h_{31})^{n+1}_j,\ S^{n+1}_{j+1/2,k},\ D^{n+1}_{j+1/2,k},\ (\Delta\xi_1)_{j+1/2}\right], \tag{175}$$

where the first 7 quantities are irrelevant for j = 1 and all are irrelevant for j= NDR; similarly, for ISTEP = 2,

$$\bar{R}^n_{k,j} = \left[V^n_{k,j},\ V^{n+1}_{k,j},\ (h_2L)^n_{k,j},\ (h_2L)^{n+1}_{k,j},\ (\Delta\xi_1)_k\ (\Delta\xi_3)_j,\right.$$

$$\left.(h_{31})^{n+1}_k,\ (h_{13})^n_j,\ S^{n+1}_{k,j+1/2},\ D^{n+1}_{k,j+1/2},\ (\Delta\xi_3)_{j+1/2}\right]; \tag{176}$$

the I-component of the Vector $\bar{R}$ is located at VAR(LOCADD + NRGR * (J − 1) + I).

NXTZGR— the pointer for the NZGR = 3 grid quantities associated with explicitly treated interfaces; for ISTEP = 1, the NZGR quantities are

$$\bar{Z}^n_{j,k} = \left[S^n_{j,k+1/2},\ S^n_{j,k+1/2},\ (\Delta\xi_3)_{k+1/2}\right] \tag{177}$$

for 2 ≤ j ≤ NDR − 1 and for 1 ≤ k ≤ NDZ − 1; similarly, for ISTEP = 2,

$$\bar{Z}^n_{j,k} = \left[S^n_{k+1/2,j},\ D^n_{k+1/2,j},\ (\Delta\xi_1)_{k+1/2}\right]. \tag{178}$$

The I-component of the vector $\bar{Z}$ is located at VAR(NXTZGR + NZGR * (J − 1) + I); for LINSEQ = 1, NXTZGR refers to the KK + 1/2 (KKP − 1/2) interface, and for LINSEQ = 2, NXTZGR refers to the KKM + 1/2 (KK − 1/2) interface.

NXTZSYM—the pointer for the symmetric explicit vectors $(\bar{g}^s_3)^n_{j,k+1/2}$ of Eq. (45) and $(\bar{f}^s_1)^n_{k+1/2,j}$ of Eq. (63); the I-component of the vector is located at VAR (NXTZSYM + NV * (J − 1) + I).

NYTZASM—the pointer for the antisymmetric explicit vectors $(\bar{g}^a_3)^n_{j,k+1/2}$ of Eq. (45) and $(\bar{f}^a_1)_{k+1/2,j}$ of Eq. (63); the I-component of the vector is located at VAR (NXTZASM + NV* (J − 1) + I).

KZFLX— the pointers for the sum total of the explicit vectors $(\bar{g}^s_3)^n_{j,k+1/2}$ + $(\bar{g}^a_3)^n_{j,k+1/2}$ + $(\bar{g}^s_3)^n_{j,k+1/2}$ − $(\bar{g}^s_3)^n_{j,k-1/2}$ of Eq. (45) and $(\bar{f}^s_3)^n_{k+1/2,j}$ + $(\bar{f}^a_3)^n_{k+1/2,j}$ + $(\bar{f}^s_3)^n_{k-1/2,j}$ − $(\bar{f}^s_3)^n_{k-1/2,j}$ of Eq. (63); the I-component is located at VAR(KZFLX + NV * (J − 1) + I).

SETPTR— rotates LCM and SCM pointers when advancing from one k- or j-line to another after convergence of the iterations.

BNDRYSW—controls implementation of *BouNDRY SW*itches such as those discussed in Ref. 4; is called after the first iteration along each k- or j-line; sets the values of the KRVAR "array" for J = 1 and J = NDR(NDZ) according to switches that have been turned on or off, if any.

SETBDRY— controls the advancement from t^n to t^{n+1} for boundary k- or j-lines; for ISTEP = 1, the k-lines 1 and K(NDZ) are set according to Eqs. (24) and (25), respectively; for ISTEP = 2, the j-lines 1 and J(NDR) are set according to Eqs. (22) and (23), respectively.

ITCON— checks for convergence of iterations; compares values $\bar{U}^{n+1,\,\ell+1}$ starting at location NAU with value $\bar{U}^{n+1,\ell}$ starting at location KRVAR; compares changes in the differences $\delta Q_{j+1/2,k}$ of Eq. (115) rather than changes in Q itself, as might normally be done; if δQ is less than the appropriate reference value as specified on the TIME STEP CONTROL card, the change in δQ is compared with the reference value; sets NI = 1 if iterations have converged; if the number of iterations KITER exceeds NEXTAVE (initially 6), $\bar{U}^{n+1,\,\ell+1}$ is set to $(\bar{U}^{n+1,\ell+1} + \bar{U}^{n+1,\ell})/2$ prior to the next iteration in an effort to achieve convergence.

MAXVAR— checks for the maximum variation of a dependent variable when advancing from t^n to t^{n+1}; compares values $\bar{U}^{n+1,\,\ell+1}$ starting at location NAU with value $\bar{U}^n$ starting at location KVAR after iterations along a k- or j-line have converged; if a component of $\bar{U}^n$ is less than the appropriate "reference" value as specified on the TIME STEP CONTROL card, the component of $\bar{U}^{n+1,\,\ell+1}$ is compared with the "reference" value.

DTCNTRL— increases, decreases, or maintains the time-step by comparing the maximum variations obtained by MAXVAR with the maximum change CHGMAX specified on the TIMESTEP CONTROL card; if the maximum change exceeds CHGMAX by 50%, or if negative densities or energies occur, will initiate a recycle by interchanging NTIME and NM1TIME and doing miscellaneous other chores; prints relevant ASCII information on timestep control at each timestep.

OUTK— transfers a completed k- or j-line from SCM locations starting at NAU to the appropriate location in the LCM block starting at NP1TIME.

ENDMAL— a subroutine to terminate ANIMAL; does a variety of ASCII output prior to termination.

VOLINT1— used when the moving-grid option is used; calculates the volume of the mesh; the value is then used in MOVGRID to adjust the CUTOFF density (see Ref. 4) as the volume changes.

15. PREPARATORY SUBROUTINES

NEXTHHP—calculates scale factors h_1, h_2, h_3 and grid velocities v_1^G and v_3^G at times t^n and t^{n+1} for all points along line NXTK; note that the actual functional form depends on ICOORD.

NEXTBBP— calculates external magnetic field components at time t^n and t^{n+1} for all points along line NXTK; note that the actual functional form depends on ICOORD.

NEXTK— brings dependent variable values $\bar{U}^n$ for all points along line NEXTK into SCM from the appropriate LCM location in the block beginning at NTIME.

NEXTTC— controls the calculation of the components of the transport-coefficient vector $\bar{K}$ of Eq. (162) for all points along line NXTK; sets up appropriate arguments and then calls subroutines EQNST and TRANCO.

EQNST— calculates equation-of-state quantities f_i, f_d, T, and θ and their derivatives with respect to ϵ and ρ; for input values of RO, searches EOSRO array to find the two density values between which RO lies, then brings into SCM locations starting at pointer NSEOST the corresponding EOS tables from LCM beginning at location NAEOST + NATAB(NRO); the tables corresponding to the two densities are spline interpolated in ϵ, then linear in ρ, to give values for f_i and f_d.

TRANCO— calculates transport coefficients; computes components of $\bar{K}$ of Eq. (162) not calculated by EQNST; all quantities are computed as algebraic functions.

NXTZGR— calculates the grid quantities of Eqs. (177) and (178); note that the actual functional form depends on ICOORD as well as ISTEP.

NXTZFLX— controls the calculation of the explicit fluxes associated with the KK+1/2(KKP−1/2) interfaces for LINSEQ = 1 and the KKM+1/2 (KK−1/2) interfaces for LINSEQ = 2; sets up COMMON blocks /9/, /10/, and /11/ for explicit coefficient subroutines DBCZFLX, UBCZFLX, and ZFLX.

SETRGR— calculates the grid quantities of Eqs. (175) and (176); note that the actual functional form depends on ICOORD as well as ISTEP.

SETSGN— determines the value for the algebraic sign of various quantities used in the spatial-difference equations for the implicitly treated interfaces; checks to see if the signs vary with each iteration, and, if so, sets the signs to zero to aid convergence; relevant variables are

SGNW1— s (v_1^R) in Eqs. (118), (119), (122), and (123).
SGNDRO—s $(\delta\rho)$ in Eqs. (118) and (148).
SGNPDV— has values 0 or 1; is used to multiply C_d of Eq. (130).
SGNDV1— s (δv_1) if $\delta v_1 < 0$, 0 otherwise; multiplies C_q^n of Eq. (127).
SGNDEI— s $(\delta\epsilon)$ in Eq. (122).

SETBC— sets up COMMON/23/ for coefficient subroutine BCRZ.

SETRP— sets up COMMON blocks /9/, /10/, /11/, and others for implicit coefficient subroutines LBCRFLX, RBCRFLX, and RFLX.

16. COEFFICIENT SUBROUTINES

BCRZ— establishes the coefficients for the boundary condition Eqs. (22) to (25); performs tests for boundary "switches" in accordance with Ref. 4; variables RO, V1 . . . refer to the first zone-center inside a boundary; variables RORM, V1RM . . . refer to the point to the "left" of the zone-center; variables RORP, V1RP . . . refer to the point to the "right" of the zone-center; if input JJJ is 1, RORM, V1RM . . . are "boundary values"; the coefficient $E_{\ell m}$ of the m-component of $\bar{U}$ in the ℓ-component equation is stored in location VAR(NBCE + L + NV * (M − 1)); the ℓ-component of $\bar{F}$ is located at VAR(NBCF + L); the component $H_{\ell m}$ of the matrices $\bar{H}$ in Eqs. (22) and (24) are stored in location HH(L + NV * (M − 1)); the component $H_{\ell m}$ of the matrices $\bar{H}$ in Eqs. (23) and (25) are stored in location HH(NV * NV + L + NV * (M − 1)); the appropriate boundary conditions are determined by arrays IBCRR, IBCR, ISBCRR, ISBCR of COMMON/28/, which are the corresponding arrays of COMMON/4/ for ISTEP = 1 and which are the arrays IBCZZ, IBCZ, ISBCZZ, ISBCZ, respectively, of COMMON/4/ for ISTEP = 2.

ZFLX— calculates explicit fluxes $\bar{g}_3^n$ of Eq. (45) and $\bar{f}_1^n$ of Eq. (63) and explicit forces $\bar{g}_3^n$ of Eq. (45) and $\bar{f}_1^n$ of Eq. (63); always assumes it is working with a j,k+1/2 interface; important variables are

RO, V1 ... — $\rho^n_{j,k}$, $(v_1^R)^n_{j,k}$ ⋯ .
ROZP, V1ZP ... — $\rho^n_{j,k+1}$, $(v_1^R)^n_{j,k+1}$ ⋯ .
ROPZ, V1PZ ... — $\overline{\rho}^n_{j,k+1/2}$, $(\overline{v}_1^R)^n_{j,k+1/2}$ ⋯ .
DROPZ, DV1PZ ... — $\delta\rho^n_{j,k+1/2}$, $\delta(v_1^R)^n_{j,k+1/2}$ ⋯ .

H(1,1), H(1,2), H(2,1) ... — $(h_1)^n_{j,k}$, $(h_1)^n_{j,k+1}$, $(h_2)^n_{j,k}$ ··· .
H12PZ—$S^n_{j,k+1/2}$.

H12IPZ—$D^n_{j,k+1/2}$.

G1B—flux from (corresponding to) Eq. (118) or (119).
G2B—flux from Eq. (121).
G4B—flux from Eq. (120).
G6B—flux from Eq. (122) or (123).
G4DS—force from Eq. (125).
G4HS—force from Eq. (126).
G4F—flux from Eq. (141).
G2F—flux from Eq. (142).
G4J—flux from Eq. (151).
G4DA—flux from Eq. (150).
G8B—flux from Eq. (124).
G8H—flux from Eq. (154).
G8D—flux from Eq. (132).
G6JS—force from Eq. (128).
G6JA—flux from Eq. (129).
G6FS—forces from Eqs. (143) and (144).
G6FA—flux from Eq. (145).
G6H—flux from Eq. (131).
G6LS—force from Eq. (133).
G6LA—flux from Eq. (134).
G8F—flux from Eq. (135).
G6PS—force from Eq. (136).
G6PA—flux from Eq. (137).
VAP—vector array corresponding to $\bar{\bar{g}}^a_3$ or $\bar{\bar{f}}^a_1$.
VSP—vector array corresponding to $\bar{\bar{g}}^s_3$ or $\bar{\bar{f}}^s_1$.

UBCZFLX— calculates boundary fluxes and forces corresponding to those of ZFLX for the k = NDZ interface on ISTEP = 1 and the j = NDR interface on ISTEP = 2; variable names are essentially same as for ZFLX.

DBCZFLX— calculates boundary fluxes and forces corresponding to those of ZFLX for the k = 1 interface on ISTEP = 1 and the j = 1 interface on ISTEP = 2; variable names are slightly different, e.g., RO, V1 . . . changed to ROZM, V1ZM . . . , ROZP, V1ZP . . . changed to RO, V1 . . . , PZ changed to MZ, B in fluxes changed to A, and D in fluxes changed to C . . . .

RFLX— calculates the coefficients $\bar{\bar{a}}$, $\bar{\bar{b}}$, $\bar{\bar{c}}$, $\bar{\bar{d}}$, $\bar{\bar{v}}$, $\bar{\bar{w}}$ of Eqs. (42) to (45) and Eqs. (60) to (63); always assumes it is working with a j+1/2,k interface; in the code listing, the "explicit" fluxes (evaluated at $\bar{U}^{n+1,\ell}$) and forces are defined in the vicinity of the corresponding coefficients; the fluxes and forces are

F1B— flux from (corresponding to) Eq. (118) or (119).
F2B— flux from Eq. (120).
F4B— flux from Eq. (121).
F6B— flux from Eq. (122) or (123).
F2DS— force from Eq. (125).
F2HS— force from Eq. (126).
F2F— flux from Eq. (141).
F4F— flux from Eq. (142).
F2J— flux from Eq. (151).
F2DA—flux from Eq. (150).

F8B— flux from Eq. (124).
F8H— flux from Eq. (154).
F8D— flux from Eq. (132).
F6JS— force from Eq. (128).
F6JA— flux from Eq. (129).
F6FS— forces from Eqs. (143) and (144).
F6FA— flux from Eq. (145).
F6H— flux from Eq. (131).
F6LS— force from Eq. (133).
F6LA— flux from Eq. (134).
F8F— flux from Eq. (135).
F6PS— force from Eq. (136).
F6PA— flux from Eq. (137).

Important variables are

RO, V1 . . . — $\rho^{n+1,\ell}_{j,k}$, $(v^R_1)^{n+1,\ell}_{j,k}$ . . . .

RORP, V1RP . . . — $\rho^{n+1,\ell}_{j+1,k}$, $(v^R_1)^{n+1,\ell}_{j+1,k}$.

ROP, V1P . . . — $\rho^{n+1,\ell}_{j+1/2,k}$, $(v^R_1)^{n+1,\ell}_{j+1/2,k}$.

DROP, DV1P . . . — $\delta\rho^{n+1,\ell}_{j+1/2,k}$, $\delta(v^R_1)^{n+1,\ell}_{j+1/2,k}$.

H(1,1), H(1,2), H(2,1) . . . — $(h_1)^{n+1,\ell}_{j,k}$, $(h_1)^{n+1,\ell}_{j+1,k}$, $(h_2)^{n+1}_{j,k}$, ...

HP(1,1), HP(1,2), HP(2,1) . . . — $(h_1)^{n+1}_{j,k}$, $(h_1)^{n+1}_{j+1,k}$, $(h_2)^{n+1}_{j,k}$ . . . .

H23PR— $S^{n+1}_{j+1/2,k}$.

H32IPR— $D^{n+1}_{j+1/2,k}$.

AAP— coefficient array corresponding to matrix $\bar{\bar{a}}$.
BAP— coefficient array corresponding to matrix $\bar{\bar{b}}$.
ASP— coefficient array corresponding to matrix $\bar{\bar{c}}$.
BSP— coefficient array corresponding to matrix $\bar{\bar{d}}$.
VAP— vector array corresponding to vector $\bar{v}$.
VSP— vector array corresponding to vector $\bar{w}$.

RBCRFLX— calculates coefficients for boundary fluxes and forces corresponding to those of RFLX for the j = NDR interface on ISTEP = 1 and the k = NDZ interface on ISTEP = 2; variable names are essentially same as for RFLX.

LBCRFLX— calculates coefficients for boundary fluxes and forces corresponding to those of RFLX for the j = 1 interface on ISTEP = 1 and the k = 1 interface on ISTEP = 2; variable names are slightly different, e.g., RO, V1 . . . changed to RORM, V1RM . . ., RORP, V1RP . . . changed to RO, V1 . . ., PR changed to MR, B in fluxes changed to A, D in fluxes changed to C.

MAT2— computes the "local" coefficients $\bar{\bar{r}}$, $\bar{\bar{s}}$, $\bar{\bar{g}}^c_3$, $\bar{\bar{t}}^c_1$ of Eqs. (42), (45), (60), (63) and computes the time-derivative coefficients $\bar{\bar{q}}$ and $\bar{m}$; assembles all coefficients including those calculated by ZFLX, RFLX, UBCZFLX, RBCRFLX, DBCZFLX, LBCRFLX into the coefficients $\bar{\bar{A}}$, $\bar{\bar{B}}$, $\bar{\bar{C}}$, and $\bar{V}$ of the tridiagonal linearized difference Eqs. (21) and (32); when completed, calculation of $\bar{\bar{E}}$ and F begins.

17. TRIDIAGONAL SOLVER SUBROUTINES

MAIN program— the DO 710 loop calculates the $\bar{E}$'s and $\bar{F}$'s according to Eqs. (27) to (30); the quantity $\bar{U}_{j,k}^{n+1,\,q+1}$ of Eq. (31) is calculated between statements 715 and 775; the DO 780 loop does the backward substitution indication in Eqs. (26) and (33); within the DO 710 loop, the DO 620 loops sets the vector array V to the quantity $\bar{V} - \bar{C} \cdot \bar{F}$ of Eqs. (28) and (30); the DO 630 loop, replaced by ASCENTF subroutine SUMBCE, sets the matrix array $\bar{A}$ to the quantity $\bar{A} + \bar{C} \cdot \bar{E}$ of Eqs. (27) to (30); the DO 635 loop adds to matrix array $\bar{B}$ the quantity $\bar{C} \cdot \bar{H}$ of Eq. (27).

SUMBCE— an ASCENTF subroutine to replace the triple-loop FORTRAN DO 630 loop of the MAIN program; sets the matrix array A to the quantity $\bar{A} + \bar{C} \cdot \bar{E}$ of Eqs. (27) to (30).

TRIANG— used to solve linear algebraic systems of equations of the form $\bar{A} \cdot \bar{X} = \bar{B}$ where $\bar{A}$ is an NV × NV matrix, $\bar{X}$ is an NV × NNV unknown matrix, and B is an NV × NNV matrix; solution procedure is Gaussian elimination, which triangularizes the matrix A; TRIANG is used to calculate the $\bar{E}$'s, $\bar{F}$'s, and $\bar{U}$ of Eqs. (27) to (31), since the inverses do not need to be computed; it does not take into account the fact that matrix A may be reducible, a fact that could lead to a reduction in execution time.[30]

18. OUTPUT SUBROUTINES

RUNDATA—prints a variety of problem information in the ASCII output file "ANIMREC" (see attached fiche for ANIMAL user's manual).

OUTPUT1— controls ASCII output in file "MHDOUT," which occurs at time and timestep intervals that appear as input data on the OUTPUT card; sets up input arrays for PLOTR.

PLOTR— does an ASCII "plot" of the NP points of array A as a function of array R.

NUMZBZ— prints ASCII zonal numbers zone-by-zone into output file "MHDOUT."

BBEXT— calculates "external" magnetic field component at a specified zone; used in conjunction with OUTPUT1, NUMZBZ.

SCALFAC— calculates scale factors at a specified zone; used in conjunction with BBEXT, OUTPUT1, NUMZBZ.

WOFORD— sets up and controls sending of ASCII messages to the RECORD OF ORDER produced by the ORDER controller that runs ANIMAL during operator-run production.

ORDMES— transmits the messages set up by WOFORD.

IFAR— ASCENTF subroutine to convert floating point numbers to Hollerith character equivalent.

BODARZ— *B*uffers *O*ut *DA* and *RZ*; controls the writing of the DA and RZ arrays into the BINARY output disc or tape file RECORD (see attached fiche for ANIMAL user's manual).

BOVAR— *B*uffers *O*ut *VAR*iables; controls the writing of time-dependent quantities, including all dependent variables, into the BINARY output file RECORD at the time and timestep intervals specified on the OUTPUT card; ANIMAL makes a BINARY record that has the following structure:

Record 1—DA array, NDA = 200 words.

Record 2—RZ array, NDR + NDZ words.

Record 3—quantities at timestep zero; DADT array, first NDADT = 100 words; RZ array only if moving-grid option is used, next NDR + NDZ words; marker-particle positions and miscellaneous related words, next 2 * NMP + 8 words; all dependent variables, next NV * NDR * NDZ words; if the total number of words exceeds NRECWD = 5000, the dump is broken into as many NRECWD records as required with the last record ≤ NRECWD.

.
.
.
.
.

Record n—same quantities as Record 3 except at a later timestep.

In Record 3 . . . . Record n, the dependent variables are stored in the order J + NDR * (K − 1 + NDZ * (I − 1)), rather than the order given by Eq. (163), to facilitate post-processing; since input/output can be done only from SCM locations, the output quantities must be brought into SCM from LCM; originally, once the words to be dumped were collected, ANIMAL used a simple BUFFER OUT statement; however, to survive under the Livermore time-sharing operating system and apparently poor tape-unit maintenance, an output procedure using all of the subroutines below was developed as a replacement for the simple BUFFER OUT statement; an END OF FILE is written after each complete dump; the END OF FILE is overwritten when a new dump is made; the dumps contain all necessary information for restart or post-processing.

BUFFO— writes a specified number of words on discs or tape; called by BODARZ, BOVAR; on tape, checks for errors by backspacing and rereading as well as checking unit status; if errors occur, will retry to write up to four times, then write blank tape in an effort to bypass bad spot on tape.

BSPVAR— backspaces a tape unit over a dump for rereading.

STATUS— checks the status of a tape unit.

WRBLNK— writes 6 in. of blank tape on a unit.

SETDEN— sets the density of a tape unit.

ENDFIL— ASCENTF subroutine to cause an END OF FILE to be written on a tape or disc; this routine is included because a more recent version in CLOB would not correctly write an EOF on disc.

GETIOC— gets the IOC or minus word associated with a logical unit number; used in conjunction with STATUS, WRBLNK, and SETDEN.

NEWTPE— starts a new disc file or tape when the current one is filled; reads TAPE card; initiates new disc or tape so it is completely independent of the previous one.

BIDARZ— controls the reading of the DA and RZ array from the BINARY output upon restart.

BIVAR— controls the reading of time-dependent quantities, including all dependent variables, from the BINARY output upon restart.

BUFFIN— reads a specified number of words from disc or tape; called by BIDARZ, BIVAR.

19. MARKER-PARTICLE SUBROUTINES

General Information. Marker, or tracer, particles are used to trace in a Lagrangian manner the motion of fluid elements. At initialization, the initial positions of an array of particles are specified. At the end of each successive timestep, a velocity for each particle is determined by interpolating on the velocity field calculated by the Eulerian code ANIMAL. The velocity is multiplied by the timestep to give a position displacement. The marker particles can be considered as vertices of a Lagrangian calculation. Because a Eulerian formulation is significantly different from a Lagrangian formulation, the marker particles do not always accurately represent the actual mass distribution in the calculation. It appears to be quite difficult to formulate a particle "pusher" that is always consistent with the way a Eulerian code moves mass.

MPINIT— initializes the marker-particles arrays; reads the MARKER PARTICLE card.

MARKER— controlling subroutine for particle pushing.

SEARCH— finds the position of a particle on the ANIMAL mesh.

GETDXDT—finds velocities of corners of a box in which the particle is located; actually works with $\frac{d\xi}{dt}$, which is not a true velocity if ξ is an angle.

INTERP1— linearly interpolates on the velocities determined by GETDXDT to give an actual velocity.

INTERP2— a dummy subroutine.

MPOUT— prints ASCII output giving positions of the particles as well as giving a plot.

20. DIAGNOSTIC SUBROUTINES

General Information. These subroutines are used when ANIMAL enters the TEST MODE (see attached fiche for ANIMAL user's manual). ANIMAL requests NCHK, the number of zones to be checked, and J2BCHK and K2BCHK [the corresponding (j, k) pairs of the zones to be checked]. A working disc file SCRATCH is created when required.

CHECK— writes coefficients $\bar{\bar{A}}$, $\bar{\bar{B}}$, $\bar{\bar{C}}$, and $\bar{V}$ for specified zones; writes $\bar{\bar{E}}$'s and $\bar{\bar{F}}$'s for a specified k- or j-line; writes $\bar{U}$'s for a specified k- or j-line; after an iteration has been completed, calls SETRP and sets up COMMON/25/ with variables $\bar{U}^{n+1,q+1}$, then calls CHKLBCR, CHKRBCR, or CHKRFLX, as appropriate.

SPECHK— lists dependent variables, transport coefficients, mesh quantities, etc. associated with the interfaces of a specified zone.

CHKRFLX—essentially a duplication of the coding of RFLX, with coding added to actually evaluate the values for fluxes and forces after $\bar{U}^{n+1,q+1}$ have been computed, such as indicated in Eq. (65); the resultant values are stored in COMMON/23/, which are then printed by CHECK; the values $\bar{U}^{n+1,q+1}$ are stored in COMMON/25/.

CHKRBCR—essentially a duplication of the coding of RBCRFLX, with coding added to actually evaluate the values for fluxes and forces after $\bar{U}^{n+1,q+1}$ have been computed.

CHKLBCR—essentially a duplication of the coding of LBCRFLX, with coding added to actually evaluate the values for fluxes and forces after $\bar{U}^{n+1,q+1}$ have been evaluated.

21. POST-PROCESSING CODES

The ANIMAL "system" consists of a post-processor MALPP and auxiliary codes TTOD, TCETC, and PLOT in addition to the ANIMAL code. The functions of the post-processor and auxiliary codes are described in the post-processor user's manual (on microfiche, inside back cover). In these codes there are no complicated mathematical algorithms with the exception of the computer graphics algorithms used in standard CLOB graphics subroutines. The post-processing codes use many subroutines with names and functions identical to or similar to subroutines in ANIMAL; in some cases the coding is also identical, but the user should not automatically assume that the coding is identical. The coding for TTOD, TCETC, and PLOT is relatively straightforward, although the coding for TTOD gets rather cumbersome because of the many functions TTOD can perform. When attempting to understand the coding of all post-processing codes, this author recommends that the user consider a particular function and follow the coding through from start to end for a typical set of input data.

The function of MALPP is to read an ANIMAL BINARY dump, manipulate the information from the dump, and do the appropriate computer graphics or ASCII output or both. For "vs time" plots, the relevant information must be stored until all dumps have been processed. Then the graphics can be performed. MALPP's main working storage is COMMON/19/, which is broken up into arrays A, B, C, and D. The graphics routines are set up to process the information in these four arrays, so the function of the "physics" routines is to set up the arrays properly.

Because MALPP is nearly 150 subroutines, a description of the subroutines will not be given here. Many of the subroutines have recognizable mnemonic names.

22. TEST PROBLEMS

When an individual gets involved with a code for the first time, it is good practice to dream up test problems for which solutions are known or for which comparisons with other codes can be made. For the code developer, the test problems can unveil bugs. For the user, the test problems can delineate the capabilities and limitations of a code.

The test problems listed below are some this author has used. Several have been suggested by others. Unfortunately, all but one are one-dimensional. If any of the readers have other useful test problems, this author would appreciate hearing of them. In this report no detailed calculational results from running the test problems on ANIMAL are given; however, a brief description of ANIMAL's performance is included.

TEST 1—1D SQUARE WAVE: cartesian coordinates $0 \leq x \leq 100$ cm; 50 uniform zones; $\rho = 10^{-5}$ kg/m^3 for 10 cm $\leq x \leq 30$ cm, $\rho = 10^{-9}$ kg/m^3 elsewhere; T = 0.01 eV for 10 cm $\leq x \leq 30$ cm, T = 1×10^{-5} eV elsewhere; $v_i = 10^7$ m/s everywhere; B = 1.0×10^{-4} for 10 cm $\leq x \leq 30$ cm, B = 1.0×10^{-7} elsewhere; $\frac{d}{dt} = 0$ at boundaries for all quantities; all transport coefficients set to zero; ideal gas, $\gamma = 5/3$; follow calculations for 0.6 μs. ANIMAL performance—because the fluid velocity is much greater than the magnetosonic velocity, all square pulses should translate 60 cm without distortion. However, since ANIMAL is a Eulerian code, dispersive and diffusive effects are present and the density wave evolves into a Gaussian-like shape. Negative densities do not occur because of the second-order (in Δx) mass diffusion given in Eq. (118). The temperature wave remains very flat but widens considerably. No negative temperatures occur because of the second-order diffusion introduced in Eq. (122). The magnetic field wave widens into a Gaussian-like main pulse with oscillations trailing the main pulse. The oscillations are a result of the dispersive errors introduced by the convective flux of Eq. (124). Because there is no dominant diffusive truncation error, some magnetic field values in the trailing oscillations are negative.

TEST 2—1D IDEAL MHD PINCH: cartesian coordinates; $0 \leq x \leq 100$ m; 40 uniform zones; $\rho = 10^{-3}$ kg/m^3; T = 1.044×10^{-2} eV; B = 0; all transport coefficients set to zero; ideal gas, $\gamma = 7/5$; at t = 0 apply a step-function vacuum magnetic field, B = 9.301×10^{-2} Wb/m^2 to x = 0 boundary; follow calculations for 0.05 s. ANIMAL performance—this is an MHD generalization of the SCTP-I-A hydrodynamics test problem suggested by Hicks.[36] A shock propagates toward increasing x. The shock is driven by a magnetic piston, which also moves toward increasing x. The magnetic piston/gas interface is a "contact discontinuity." Representation of the piston (or vacuum)/gas interface is a nontrivial problem that has plagued computational magnetohydrodynamicists since the early calculations of Hain et al.[13] ANIMAL uses a new "background plasma" method.[4] For this problem, the "background plasma" parameters are $\rho = 10^{-6}$ kg/m^3 and T = 10^{-4} eV. The inflow velocity is not specified, and ANIMAL computes the correct value to within 1%. Both the "contact discontinuity" and the shock appear as transition regions over several zones. The average compression behind the shock is correct, but the density, temperature, and velocity profiles show "ringing" and "overshoot," which is characteristic of second-order methods. The shock arrives at the x = 100-m boundary at roughly the correct time. ANIMAL's predecessor[17] did not perform nearly as well on the hydrodynamic analog of this problem; the poor shock propagation can be traced to the conservation[3] properties of the predecessor.

TEST 3—1D RIEMANN SHOCK TUBE PROBLEM: cartesian coordinates; -1 cm $\leq x \leq 2$ cm; 30 uniform zones; $\rho = 10^3$ kg/m^3 and T = 695.6 eV for $x < 0$; $\rho = 1$ kg/m^3 and T = 6.956×10^{-4} for x = 0; all transport coefficients set to zero; ideal gas, $\gamma = 5/3$; follow calculations for 5×10^{-8} s. ANIMAL performance—this is similar to the SCTP-V test problem discussed by Hicks.[36] These particular parameters were suggested by Trigger.[33] The correct solution is a strong shock propagating into the low-density material and movement of the "contact discontinuity" of the two fluids. Because the initial pressure jump is a factor of 10^9, this is a severe problem. ANIMAL's shock position is correct within 10%, and ANIMAL's accuracy is comparable to LLL's explicit hydrocodes (for this type of problem, ANIMAL's timestep will be comparable to the explicit limit). The temperature in the shocked material is also acceptable. The contact and shock in the density profile are difficult to distinguish at these short times with so few zones. If the "velocity fractional step" method of Section 6 is used, the shock propagates significantly too slow, with corresponding inaccuracies in all other relevant quantities.

TEST 4—REFLECTED SHOCK: cartesian coordinates; $0 \leq x \leq 50$ cm; 25 zones, $0 \leq x \leq 25$ cm, each adjacent zone 1.16 times larger than the adjacent zone at greater x; 25 zones, 25 cm $\leq x \leq 50$ cm, each adjacent zone 1.16 times larger than the adjacent zone at lesser x; $\rho = 10^3$ kg/m^3; T = 10^{-2} eV; $v_i = 3 \times 10^5$ m/s; all transport coefficients set to zero; ideal gas, $\gamma = 5/3$; $\frac{d}{dt} = 0$ for all quantities at x = 0; rigid wall at x = 50 cm; follow for 5.0 μs. ANIMAL performance—this problem models a translating fluid coming into contact with a rigid wall. A strong shock wave with compression ratio 4 should propagate away from the wall. This problem was originally suggested by Le Blanc[32] to test the effect of nonuniform zoning. For uniform zoning, ANIMAL will calculate the correct average compression, and the density profile will show small-amplitude oscillations behind the shock. However, with nonuniform zoning there is a tendency to overcompress by a factor of nearly 10% for $x > 25$ cm and a tendency to undercompress by a factor of nearly 10% for $x < 25$ cm. This same tendency is observed in a variety of LLL Eulerian and Lagrangian codes, and the actual overcompression or undercompression depends on the coefficient used in the "artificial viscosity."[32] This problem suggests that nonuniform zoning should be used cautiously.

TEST 5—1D COAXIAL PINCH: cylindrical coordinates; 1 cm $\leq r \leq 10$ cm; 45 uniform zones; $\rho = 3 \times 10^{-4}$ kg/m^3; T = 3 eV; boundaries are electrically and thermally insulating; radiation losses turned off, but thermal conductivity and resistivity on; ideal gas equation-of-state, $\gamma = 5/3$; apply a sinusoidal current along the cylindrical axis; the current amplitude is 450 kA and the quarter cycle is 5 μs; follow the calculations for 15 μs. ANIMAL performance—this is a "real" pinch problem. The plasma separates from both inner and outer boundaries,

leaving behind a "vacuum." At a later time, the plasma returns to contact both boundaries. ANIMAL's performance on this problem is reported in Ref. 4.

TEST 6—1D DIFFUSION PROBLEMS: for constant thermal conductivity (resistivity), exact solutions to the thermal (resistive) diffusion equation are available in a variety of references that will not be cited here.

TEST 7—2D ISOLATED SYSTEM: initial conditions are an arbitrary distribution of plasma and magnetic field; the system is surrounded by electrically conducting, thermally insulating walls. ANIMAL performance—mass and magnetic flux are constant to within machine roundoff errors. The total system energy—kinetic + thermal + magnetic—is constant to within 0.01% in spite of strong turbulence. This problem is intended to verify the conservation properties of the code.

23. ACKNOWLEDGMENTS

The ANIMAL code as it exists today is a result of contributions from many people. Joe Pettibone, as a former group leader, provided the guidance and much of the physical insight necessary to build ANIMAL. John Stevens, Larry Suter, and Dave Kraybill participated in the development of the physical model; all spurred the development of the code by conceiving interesting and challenging physical problems. Rollin Harding participated in many useful discussions on numerical methods. John Stevens added several features to MALPP, including the original "offline" marker particles that greatly added to our physics understanding. Larry Suter made modified versions of ANIMAL and MALPP to handle problems of interest to him. Ray Cochran added many user-convenience and system-interaction capabilities to both codes, considerably increased ANIMAL's operating speed, and performed quite a variety of miscellaneous, very necessary chores. Alan Mankofsky, a 1975 summer employee, generated ANIMAL's "online" marker-particle capability. John Brasunas, a 1976 summer employee, converted handwritten user's manuals to the current online versions. Finally, Grant Cook, a graduate student at the National Magnetic Fusion Energy Computer Center, did extensive proofreading of the manuscript.

REFERENCES

1. K. V. Roberts and D. E. Potter, "Magnetohydrodynamic Calculations," in *Methods in Computational Physics*, B. Alder, S. Fernbach, and M. Rotenberg, Eds. (Academic Press, New York, 1970), vol. 9, p. 339.
2. R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial Value Problems* (Interscience, New York, 1976), 2nd ed.
3. I. R. Lindemuth, *J. Computational Phys.* **18**, 119 (1975). Erratum, *J. Comput. Phys.* **19**, 338 (1976).
4. I. R. Lindemuth, *J. Comput. Phys.* **25**, 104 (1977).
5. I. R. Lindemuth, J. S. Pettibone, J. C. Stevens, R. C. Harding, D. M. Kraybill, and L. J. Suter, *Phys. Fluids* **27**, 1723 (1978).
6. I. R. Lindemuth and T. R. Jarboe, *Nucl. Fusion* **18**, 929 (1978).
7. I. R. Lindemuth, and M. M. Widner, *Magnetohydrodynamic Behavior of Thermonuclear Fuel in a Preconditioned Electron-Beam Target*, Lawrence Livermore Laboratory, Livermore, CA 94550, submitted for publication to *Nuclear Fusion*.
8. J. C. Stevens, The KRAKATOA Program, Lawrence Livermore Laboratory, Livermore, CA 94550, in preparation.
9. R. C. Harding, D. M. Kraybill, I. R. Lindemuth, J. S. Pettibone, J. C. Stevens, and L. J. Suter, *Numerical Computation of the Preionization Phase of a Pinch Discharge*, Lawrence Livermore Laboratory, Livermore, CA 94550, submitted for publication to *Plasma Physics*.
10. J. Allbritton, J. Cohen, R. S. Devoto, and R. Rowlands, "Flute Instability of a Laser-Pellet Plasma in a Magnetic Mirror," in *Proc. Ann. Mtg. Theoretical Aspects Controlled Thermonuclear Fusion, Madison, Wisconsin, April, 1976* (University of Wisconsin, Madison, Wisconsin, 1976).
11. L. J. Suter and I. R. Lindemuth, "2D Simulations of CO_2 Laser Breakdown and Propagation in Hydrogen Gas," in *Proc. IEEE Int'l. Conf. Plasma Science, Rensselaer Polytechnic Institute, Troy, New York, May, 1977* (IEEE, New York City, 1977): D. M. Kraybill, I. R. Lindemuth, and L. J. Suter, *Bull. Am. Phys. Soc.* **22** (9), 1204 (1977); L. J. Suter and I. R. Lindemuth, *Bull. Am. Phys. Soc.* **22** (9), 1119 (1977).
12. L. J. Suter, D. M. Kraybill, I. R. Lindemuth and J. C. Stevens, "Line and Continuum Radiation as a Liner Implosion Diagnostic," in *Proc. 2nd Topical Conf. High Temp. Plasma Diagnostics, Santa Fe, New Mexico, March, 1978* (Los Alamos Scientific Laboratory, Los Alamos, NM 87544, 1978).
13. K. Hain, G. Hain, K. V. Roberts, S. J. Roberts and W. Koppendorfer, *Z. Naturforsch. A* **15**, 1039 (1960).
14. D. Duchs, *Phys. Fluids* **11**, 2010 (1968).
15. J. R. Freeman and F. O. Lane, "Initial Results from a Two-Dimensional Lax-Wendroff Hydromagnetic Code," in *Proc. 2nd Conf. Numerical Simulation of Plasma, Paper C7*, LA-3990 (Los Alamos Scientific Laboratory, Los Alamos, NM 87544, 1968).
16. W. Schneider, *Z. Physik* **252**, 147 (1972).
17. I. Lindemuth and J. Killeen, *J. Comput. Phys.* **13**, 181 (1973); I. R. Lindemuth, Lawrence Livermore Laboratory, Livermore, CA, 94550, UCRL-51103 (1971).
18. D. E. Potter, *Phys. Fluids* **14**, 1911 (1971).
19. J. R. Freeman, *Nucl. Fusion* **11**, 425 (1971).
20. K. V. Brushlinsky, *Comp. Meth. in App. Mech. and Eng.* **6**, 293 (1975).
21. F. Hofmann, *Nucl. Fusion* **14**, 438 (1974).
22. H. C. Lui and C. K. Chu, *Phys. Fluids* **18**, 1277 (1975).
23. J. U. Brackbill, "Numerical Magnetohydrodynamics for High Beta Plasmas," in *Methods in Computational Physics*, B. Alder, S. Fernbach, and M. Rotenberg, Eds. (Academic Press, New York, 1976), vol. 16.
24. S. I. Braginskii, "Transport Processes in a Plasma," in *Reviews of Plasma Physics*, M. A. Leontovich, Ed. (Consultants Bureau, New York, 1965), vol. I, p. 205.
25. J. C. Stevens, Lawrence Livermore Laboratory, Livermore, CA, 94550, private communication (1976).
26. H. R. Griem, *Plasma Spectroscopy* (McGraw-Hill, New York, 1964).
27. L. Spitzer, *Physics of Fully Ionized Gases* (Interscience, New York, 1962), 2nd ed.
28. R. H. Pennington, *Introductory Computer Methods and Numerical Analysis* (Macmillan, New York, 1965).
29. W. R. Briley and H. McDonald, *J. Comput. Phys.* **24**, 372 (1977).
30. W. R. Briley and H. McDonald, *On the Structure and Use of Linearized Block ADI and Related Schemes*, Scientific Research Associates, Inc., Glastonburg, Connecticut, R78-3 (1978).
31. J. Douglas and J. Gunn, *Num. Math.* **6**, 428 (1964).
32. J. M. LeBlanc, Lawrence Livermore Laboratory, Livermore, CA 94550, private communication (1975).

33. K. R. Trigger, Lawrence Livermore Laboratory, Livermore, CA 94550, private communication (1973).
34. T. E. Rudy, Lawrence Livermore Laboratory, Livermore, CA 94550, private communication (1975).
35. R. C. Cochran, Lawrence Livermore Laboratory, Livermore, CA 94550, private communication (1977).
36. D. Hicks, *Hydrocode Test Problems*, Air Force Weapons Laboratory, Kirkland AFB, New Mexico, AFWL-TR-67-127 (1968).

LG

*LLL: 1979/6