PREPARED FOR THE U.S. DEPARTMENT OF ENERGY,
UNDER CONTRACT DE-AC02-76-CHO-3073

PPPL-2605

UC-427

PPPL-2605

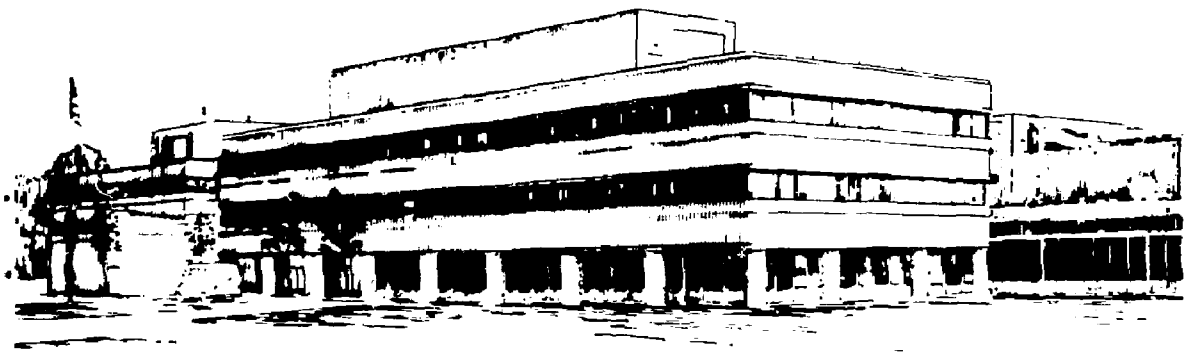
TOWARD THE AUTOMATED ANALYSIS OF PLASMA PHYSICS PROBLEMS

BY

HARRY E. MYNICK

APRIL 1989

PRINCETON
PLASMA PHYSICS
LABORATORY



PRINCETON UNIVERSITY, PRINCETON, NEW JERSEY

Printed in the United States of America

Available from:

National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22161

Price Printed Copy $ * ; Microfiche $4.50

*Pages	NTIS Selling Price
1-25	$7.00
25-50	$8.50
51-75	$10.00
76-100	$11.50
101-125	$13.00
126-150	$14.50
151-175	$16.00
176-200	$17.50
201-225	$19.00
226-250	$20.50
251-275	$22.00
276-300	$23.50
301-325	$25.00
326-350	$26.50
351-375	$28.00
376-400	$29.50
401-425	$31.00
426-450	$32.50
451-475	$34.00
476-500	$35.50
500-525	$37.00
526-550	$38.50
551-575	$40.00
567-600	$41.50

For documents over 600 pages, add $1.50 for each additional 25-page increment.

PPPL--2605

DE89 010271

Toward the Automated Analysis of Plasma Physics Problems

Harry E.Mynick
Princeton Plasma Physics Laboratory
Princeton University
P.O. Box 451
Princeton, New Jersey 08543-0451, U.S.A.

ABSTRACT

A program (CALC) is described, which carries out nontrivial plasma physics calculations, in a manner intended to emulate the approach of a human theorist. This includes the initial process of gathering the relevant equations from a plasma knowledge base, and then determining how to solve them. Solution of the sets of equations governing physics problems, which in general have a nonuniform, irregular structure, not amenable to solution by standardized algorithmic procedures, is facilitated by an analysis of the structure of the equations and the relations among them. This often permits decompositions of the full problem into subproblems, and other simplifications in form, which renders the resultant subsystems soluble by more standardized tools. CALC's operation is illustrated by a detailed description of its treatment of a sample plasma calculation.

1. Introduction

Understanding the processes by which a human analyzes problems in physics is of obvious interest to artificial intelligence (AI), from both a theoretical and a practical standpoint. In earlier work [1] I have described the goals and early structure of a "Plasma Apprentice Program" (PAP), whose general purpose is to automate as much as possible the activities of a human plasma theorist, using techniques from the fields of artificial intelligence and symbolic computation. An apprentice system able to handle even a modest fraction (25%, perhaps) of the tasks which form the human theorist's job would represent a great enhancement of his productivity. Since much of what a theorist does has a structure which is fairly clear, and therefore amenable to machine imitation, the possibilities for a practical apprentice system seem quite real. On the more theoretical side, plasma theory provides a rich and formally developed domain in which to study a range of AI problems, including problem solving, planning, theorem proving, and learning. In its knowledge base (KB), the Apprentice possesses information about the important physical quantities in plasma physics, the equations governing them, something about the relations among them, and a body of mathematical tools for manipulating them. In order to give it the requisite algebraic capability, most of the PAP routines are written in the MACSYMA language, with a few utility routines written in LISP.

At the more ambitious end of the spectrum of potential facilities discussed in Ref. 1 is the automation of setting up and solving nontrivial plasma physics calculations. A good deal of work has been done [2] in which a human user either guides an algebraic manipulator like MACSYMA through some physics calculation, or in which the purpose of the calculation is sufficiently narrow, and the corresponding structure sufficiently clear, that the control structure a human would impose can be written out beforehand. In either case, the user provides the analysis and control structure, regulating the invocation of the MACSYMA tools. In this work, in contrast, the objective is to attempt to automate that analysis and control structure. As a first step in this direction, an early form of a "high-level calculator," CALC, was applied to doing plasma transport calculations. Previously known but nontrivial transport results were recovered, with the user providing only minimal guidance. However, attempting to apply the early version of CALC to other plasma problems as well has brought out a number of deficiencies in the original approach, requiring substantial broadening of its solution strat-

egy. The resultant upgraded version is a good deal more general, less brittle, and more human in its operation than previously. This paper describes the operation of the new CALC, and attempts in so doing to bring out some of the significant issues involved.

I begin Sec. 2 by giving a brief description of the nature of magnetic confinement of plasmas, and then discuss the particular plasma physics problem chosen for illustration, defining the problem, and outlining the course of its solution. It should be recognized that, while the particular problem domain being studied here is plasma physics, the general approach being developed is of much broader applicability, equally relevant to any other domain which is conveniently described mathematically. Section 3 describes how CALC approaches the sample problem. Section 4 provides some summarizing discussion, reflecting on some of the important features of this kind of problem solving which are brought out by CALC's operation.

2. A Model Linear Response Problem

Before focussing on the particular illustrative problem to which CALC is to be applied, a few brief remarks are in order on plasma problems generally. A plasma is an ionized gas. Because it is a gas, it is often convenient to describe it in terms of fluid quantities, such as density, temperature, and pressure. Because it is ionized, the plasma responds to electric and magnetic fields. In particular, while a plasma can flow parallel to a magnetic field **B** almost unimpeded, it has great difficulty moving *across* **B**. This fact provides the basis of devices (such as tokamaks) which " magnetically confine" high-temperature plasmas in the laboratory. The plasma is created within a magnetic field, whose lines run in a circle, (approximately) closing on themselves, and thus defining a toroidal volume. Moving easily along the field, the plasma runs in a circle, and thus does not escape, and it moves across **B** only with difficulty, and so is slow to make its way across the "minor radius" r of the torus, *i.e.*, from the center of the torus to its edge.

The plasma always finds ways to outwit this "magnetic trap" eventually, and the objective is to design the trap so that this takes as long as possible. One quick way for the plasma to reach the wall is if the plasma is "unstable" to an initial small displacement from its ideal, equilibrium position. When the plasma is given such a displacement, it generates fields which, if the trap is not properly designed, will cause the displacement to grow further. The calculation of how the plasma will respond in time to such a small

initial perturbation is a "linear response" calculation, an example of which is described in this paper. Even if there are no such linear instabilities of the plasma, the particles of the plasma are still being bounced about, due to collisions with other plasma particles, and small deviations from the equilibrium fields which are also present. These cause the plasma particles to move *diffusively*, hence on a longer time scale than that induced by an instability, but nevertheless allowing the plasma to escape rapidly enough to be of practical interest. This sort of slower escape of the plasma from the confinement volume is called "plasma transport."

We now turn to consideration of the particular illustrative problem, a linear response calculation, using a simplified set of model fluid equations in its KB, instead of the full fluid equations, which are often used in realistic plasma calculations. Employing the same calculation strategy, CALC is also successful for both the transport calculation on which the earlier version of CALC succeeded, and for a more realistic linear response calculation, using the full fluid equations. The structure of the transport calculation was simpler (in a sense to be made more precise later on) than the linear response calculations, on either of which the earlier CALC fails. I discuss the simplified fluid model (SFM) because it is less complicated, while retaining some of the physics, and bringing out the same structural features of the calculation as using the full fluid model (FFM).

We (and CALC) shall assume a "slab geometry," a simplifying geometry commonly used by plasma theorists (cf.Fig. 1). Parametrizing space by Cartesian coordinates (x, y, z), the slab geometry has straight magnetic field lines $\mathbf{B}$ lying in the z direction, with translational symmetry in the y and z directions, but with inhomogeneities of the magnetic field and equilibrium plasma possible in the x direction, which corresponds to the minor-radial direction r in a realistic confinement device. On this slab equilibrium is superposed a small initial perturbation of the plasma and electric field, which varies sinusoidally in the y-direction. The purpose of the linear response calculation is to determine how this initial perturbation develops in time.

Specifically, the equations of the SFM are

$$\partial_t n - u_{y0}\partial_y n - \partial_x(nu_x) = 0, \tag{1}$$

$$\partial_y E_y = 4\pi e(n - n_0), \tag{2}$$

$$u_x = cE_y/B. \tag{3}$$

Here, for any variable v, $\partial_v \equiv \partial/\partial v$ denotes a partial derivative with respect to v, n is the plasma density, n_0 is a background equilibrium density, as-

sumed to be a known function of x, as is the fluid flow velocity u_{y0} in the y direction. u_x is the fluid flow velocity in the x direction, E_y is the electric field in the y direction, B is the magnitude of the magnetic field, e is the proton charge, and c is the speed of light. Eq. (1) is the "continuity equation," slightly simplified from the exact form valid in the FFM by our assumption that $u_y = u_{y0}$ depends only on x, and by our neglect of flow u_z in the z direction. Eq. (2) is Poisson's equation, simplified from the full equation by neglect of the contributions from the x and z components $E_{x,z}$ of the electric field, and by our replacement of the electron density by a static neutralizing background n_0. Eq. (3) is the x component of the "$E \times B$ drift," describing the response of a magnetized plasma to an electric field applied normal to B. One notes that Eqs. (1) and (2) are differential equations, while Eq. (3) is algebraic, and may be thought of as making u_x a subsidiary variable, a specified function of E_y. Using it in Eq. (1) to eliminate u_x in favor of E_y, one sees in Eqs. (1,2) the self-consistent coupling characterizing plasma problems generally: the motion of the plasma (here characterized by the single variable n) is determined by the fields [Eq. (1)], and the fields (here described by the single variable E_y) are determined by the plasma [Eq. (2)]. The structure is the same for the FFM, but there, several more variables are required to represent the state of the plasma, and the fields. Moreover, while the FFM is a system of pdes in 4 independent variables (t, x, y, z), for the SFM this number has been reduced to only 3, (t, x, y).

The task which CALC will be asked to perform is to find the linear response behavior of the variable E_y, assuming it is governed by the SFM. There are two steps to be taken here. First is the process of collecting the set of equations governing the problem, within the confines of some chosen theoretical model (here, the SFM). This should yield the set (1-3). Second, finding the "linear response" behavior of these equations implies following a certain perturbative approach to solving the system (1-3), involving the following steps: (a) First, compute the "equilibrium" for this system of equations. This entails setting all the time derivatives to zero, and, further simplifying the equations by using the symmetries of the geometry at hand, finding the steady-state values v_0^i for the dynamical variables of the problem [in this case, $\{v^i\} \equiv \{n, E_y\}, (i = 1, 2)$, and, through Eq. (3), u_x]. (b) Linearize the equations about the equilibrium, and solve for the linear portion $v_1^i(t)$ of the v^is. This is facilitated by Fourier transforming the solution in each of the symmetry directions of the unperturbed problem (in the case of the SFM problem, t and y). Equivalently, this means writing the v_1^i as an

amplitude $\hat{v}_1(x)$ times the "eikonal" factor $\exp i(k_y y - \omega t)$. This substitution turns differential operators into algebraic factors, making the resultant linearized equations easier to solve. From these equations one determines the system's "dispersion relation," i.e., the equation $D(\omega) = 0$ which determines the (possibly complex) eigenfrequencies ω of the linearized equations. Instability is implied if the imaginary part of ω is positive. Lastly, (c) solve the dispersion relation for the eigenfrequencies, and, knowing these, solve the linearized equations for the system eigenmodes.

This concludes a brief characterization of the problem to be treated, from a human perspective. In the following section, we will see that the process which the Apprentice goes through in collecting the relevant equations and physical variables, assessing their roles in the problem and, on the basis of this, choosing a solution approach, is quite parallel.

3. The Automated Linear Response Calculation

We now turn to the approach which PAP takes in solving the SFM problem described in the previous section. In writing input to or output from the Apprentice, we will avoid the use of MACSYMA-specific notation where possible, rewriting the mathematics in more standard mathematical notation. However, it will be helpful to use it occasionally. Thus, square brackets in such expressions, enclosing comma-delimited components, designate a list. The MACSYMA function `get(vr,p)` returns property `p` of variable `vr`, the function `put(vr,vl,p)` gives property `p` of variable `vr` the value `vl`, and the function `part(l,i)` returns the *ith* part of list `l`. An underscore is a legal character within a symbol name. Other symbolic notation is common to MACSYMA and many other programming languages, and so should be clear.

The desired calculation is invoked by issuing the command

`calc(lin_rsp(`E_y`),[geometry=slab,model=sfm])`.

The first argument **exp** [with value `lin_rsp(`E_y`)` here] is the object to be calculated. The second argument, **cmpentxt**, is a list of qualifiers specifying the "computational context" within which the calculation is to occur[1]. At certain points in the calculation, CALC makes use of information about the particular geometry (if any) being considered (*e.g.*, symmetry directions), and it finds such information by checking **cmpentxt** for geometry qualifiers. The qualifier `model=sfm` is used by PAP in selecting from the KB as

governing equations those of the SFM. For example, Eq. (2) is to be used in determining E_y, rather than the full Poisson's equation, which would enter for the FFM.

Earlier, **exp** had to be a purely algebraic object, *i.e.*, one involving only the standard algebraic operations. This made the range of tasks which could be expressed to CALC too restrictive to deal with many tasks of interest. For example, the sequence of operations involved in a linear response calculation, outlined in the previous section, cannot readily be described in a single algebraic formula. Rather, a more natural description is in terms of a *procedure*, each of whose steps may involve algebraic operations, or still other procedures. A purely algebraic expression may also be thought of as describing a particular simple class of procedures (*e.g.*, "first, add x to 3, then, multiply this result by the sum of y minus 7,.."), so this extension of the allowable domain of **exp** represents a natural extension of the language which CALC understands. The operator **lin_rsp** acting on the variable E_y in the invocation of CALC above is the name of a non- purely-algebraic procedure. We refer to such procedures as "scripts," intended to denote, as in the use of the term in natural language comprehension, a sequence of events or operations being implied by some shorthand designation. Currently, encounter of a script name is noted by adding a qualifier to **empcntxt**, *e.g.*, **calctyp=lin_rsp**. At certain points in CALC's operation, indicated below, the presence of script qualifiers in **empcntxt** are checked for, and if present, these affect the choices CALC makes.

We now consider the general method by which the calculation is carried out. CALC runs in a cycle, consisting of first assembling the relevant equations to be solved (*via* the function KITOGEN), then analysing the structure of this system of equations (GENSUPRTREE,CLASSIFY), then solving them (SFRSND), and then absorbing the solution so obtained into the system's KB (DIGEST-SOLN). Often, these solutions introduce new variables into the problem, to which the same cycle must then be applied, until no new variables are introduced.

I now discuss the cycle in a bit more detail. First, the function KITOGEN generates a "solution net" for the problem[1], a graph showing the connections among the variables and equations involved. (As noted in Ref. 1, "KITO" stands for "Known In Terms Of," the question answered by the solution net.) The solution net developed from the initial application of KITOGEN to **lin_rsp**(E_y) is shown in Fig. 2. This given task is assigned to the root node **nd1** of the net. Newly encountered variables, and their

node names, are stored on a list called **stack**. KITOGEN terminates when all variables encountered have been properly added to the solution net, hence when **stack** is empty. Given a node as yet untreated, if the node value is a nonatomic expression, KITOGEN creates a new child node for each variable in that expression which has not yet been encountered, linking the parent expression with the child variable by an appropriate arc. If the variable has already been encountered, only a new parent-child arc is created. If the node value is a variable v, KITOGEN looks under the **givenby** property of v to see what equation it should use to determine it. If the equation $E_v(v,\ldots)$ for v is of the "unravelled" form $v = V(\ldots)$, with the function V independent of v, then a child node to v's node is created, with value $V(\ldots)$. In the SFM example (Fig. 2), such an equation is Eq. (3), with $v \rightarrow u_x$. If E_v is not of this form, so that it must be solved for v, then instead the child node created has value **solvefor**(E_v, v) [abbreviated to **sfr**(E_v, v) in Fig. 2], and this node is put on the list **sfrstack**, in addition to being attached to its parent node in the solution net. Such nodes in Fig. 2 are **nd3** and **nd8**, corresponding to Eqs. (2) and (1), respectively. If no equation is found from the request **get(v,givenby)**, the node is a terminal (*i.e.*, "leaf") node, and v is treated as a fundamental quantity, such as e or y in Fig. 2.

At the end of KITOGEN's operation, a solution net has been generated, **stack** is empty, **sfrstack** contains the nodes representing each of the equations (or sets of equations) which must be solved, and from the net structure, a preliminary classification of the variables encountered has been developed for subsequent use (see below).

The objective now is to determine how to simultaneously solve all of the equations whose nodes are in **sfrstack**, for all the variables listed there. Often, this task can be decomposed into a number of separate subsets of equations and variables, which, when possible, considerably lessens the difficulty of solution. The decomposition is accomplished by the routine GENSUPRTREE, which partitions the solution net into a covering network of "strongly connected components," or "supernodes," *i.e.*, collections of nodes having the property that each node in a given supernode is both a descendant and an ancestor of every other member of the supernode, and therefore of itself. While this "super-net" is not necessarily a tree, it does share with trees the property that it contains no loops (circuits) as the solution net may, and therefore establishes a clear order in which the supernodes are to be solved, *viz.* from the terminal supernodes upward (*i.e.*, toward the ancestor supernodes).

The initial stage of the problem, shown in Fig. 2, possesses only a single nontrivial supernode (i.e., one which contains more than a single node) **snd1**, and this contains all of Eqs. (1–3). Thus, the topology of the solution net, as brought out by its supernode structure, reflects the coupling which exists among the equations in the problem. The full solution net, shown in Fig. 3, possesses two additional supernodes **snd2** and its parent **snd3**. These correspond, respectively, to the zero and first-order portions of the problem, and, as the direction of descent correctly indicates, it is first appropriate to solve the zero-order supernode **snd2**, and then **snd3**.

The transport calculation, in contrast, had no nontrivial supernodes, hence no coupling of the equations to solve. It is in this sense that the structure of the transport calculation is simpler than the linear response calculation for either the SFM or FFM, and for this reason that the earlier CALC control structure could cope with it. Moreover, the supernode structure of the SFM and FFM problems are identical, though the nontrivial supernodes for the FFM each contain more nodes. This corresponds to the human sense that these two problems have essentially the same structure, but with the SFM being simpler in detail.

The supernode structure being determined, the supernode-solving routine SFRSND is called successively on the supernode of the first element of **sfrstack**, until **sfrstack** is empty. Once a supernode **snd** has been solved, SFRSND removes all nodes which **snd** has in **sfrstack**. Before solving a given supernode, SFRSND first checks that all its children have been solved and, if not, recursively applies SFRSND to them. In the SFM illustration, for example, this ensures that the zero-order problem will be solved before the first-order problem.

Having found a supernode to be solved, SFRSND assembles, from all its nodes in **sfrstack**, the full set of equations (assembled into a list **eqlst**) and variables v^i (assembled into a list **vlst**) for which it is to solve. It then calls the routine CLASSIFY, which uses the structure of the equations in **eqlst** to partition the variables occurring in this **snd**'s subproblem into "variable classes" VCi (i=0–4). VC0 is the set of constant parameters in the problem, VC1 is the set of "independent variables" (t, x and y, for the full SFM problem of **snd1**), VC2 is that subset of dynamical variables v^i in **vlst** judged to be determined by a differential equation in **eqlst**, while VC3 is that subset of **vlst** judged to be determined by equations in **eqlst** which are algebraic in those v^i. Finally, variable class VC4, containing those variables (like u_x in the SFM problem) which may be regarded as

auxiliary functions of the other variables, has already been determined by the preliminary classification occurring in KITOGEN. Parallel to each of VC2–4 are equation lists **eqlst2-4**, containing the determining equations of the corresponding VCi. On this first pass, CLASSIFY returns

VC0 = $[e, B, c, u_{y0}, n_{00}]$
VC1 = $[t, x, y]$
VC2 = $[E_y, n]$
VC3 = [ ]
VC4 = $[u_x]$
eqlst2 = $[\partial_t E_y = 4\pi e(n - n_{00}), \partial_t n + u_{y0}\partial_y n + \partial_x(cE_y n/B) = 0]$
eqlst3 = [ ]
eqlst4 = [uxeq],

where **uxeq** means Eq. (3). One notes that the variable classification is in accord with the roles which a human would assign them.

Solving a supernode means obtaining expressions for each of the v^i comprising VC2 and VC3 of the "unravelled" form mentioned above in connection with KITOGEN. On obtaining such a form for a v^i, it is transferred from VC2 or VC3 to VC4, and the corresponding unravelled expression is placed on **eqlst4**. It should be noted that this variable partition is relative to the given supernode under consideration. A given variable plays different mathematical roles for different supernodes.

Once the classification from CLASSIFY has been obtained, SFRSND first calls subroutine SFA, which solves the algebraic equations in **eqlst3** for the variables in VC3, then transferring these to VC4. Using the equations in **eqlst4**, the differential equations in **eqlst2** are then written just in terms of the variables in VC0–2, by the subroutine VC4XPND. The differential equations in this "expanded" form are then submitted to subroutine SFD, which attempts their solution. SFD refers to **cmpentxt** for indications of what kinds of solution procedures might be appropriate. There, the qualifier **calctyp=lin_rsp** tells SFD that the perturbative procedure, outlined at the end of Sec. 2, should be used. In the transport calculation, in which **cmpentxt** provided no such information, CALC first embarked on a qualitative analysis of the kinetic equation whose solution was required, and on this basis decided to attempt a perturbative solution. Here, CALC is informed from **cmpentxt** that a perturbative solution is appropriate, and so is able to skip the initial analysis of the kinetic equation. This decided, however, the routines employed to generate the expansion hierarchy, which essentially

encapsulate a certain mathematical technique, are the same.

For some systems of equations, SFA and SFD are able to solve the equations outright, and when this occurs, they return a list of the desired unravelled equations giving the sought–after v^i. For other problems, the "solution" returned is not a real solution, but is instead a formal solution in terms of new variables, which must themselves be solved for in order to genuinely solve the problem, but for which the governing equations are (hopefully) simpler than those for the original set. This parallels, of course, a familiar human technique for reducing the complexity of a problem. For the linear response solution procedure described at the end of Sec. 2 (and in many other cases as well), two such formal solutions are used. The first was the introduction of an expansion, $v^i = v_0^i + \epsilon v_1^i + \ldots$, in which the new dynamical variables introduced are the v_0^i and v_1^i, and ϵ is an expansion parameter. The second was the introduction of the eikonal form for the first–order variables, $v_1^i(t, x, y) = \hat{v}_1^i(x) \exp i(k_y y - \omega t)$, with new variables the $\hat{v}_1^i$. From the initial call to SFRSND on the SFM problem, for example, where VC2=$[E_y, n]$ and VC3=$[\,]$, all the equations to be solved are differential, so that only SFD is called, returning a list of the form `[solutions,INWHICH,new_info]`. Here, `solutions` is a list of solutions for variables already encountered. In the present case, `solutions` is

$[n_0 = n_{00}(x), E_y = E_{y0} + \epsilon \hat{E}_{y1} \exp i(k_y y - \omega t),$
$n = n_{00}(x) + \epsilon \hat{n}_1 \exp i(k_y y - \omega t)]$.

This introduces some new variables (*e.g.*, $\hat{E}_{y1}$), which are determined by new governing equations. The latter are contained in the list `new_info`, along with some additional qualifiers (`xpnparm=`ϵ`,eikn1=`$\ldots$), which provide both CALC and the user with useful information. For the present calculation, `new_info` is

$[$givenby$([E_{y0}],$ $[(cE_{y0}/B)\partial_x n_{00}(x) + (cn_{00}(x)/B)\partial_x E_{y0} = 0]]$,
givenby$([\hat{E}_{y1}, \hat{n}_1],$
$[ik_y \hat{E}_{y1} = 4\pi e \hat{n}_1, (c\hat{E}_{y1}/B)\partial_x n_{00}(x) - i\omega\hat{n}_1 + ik_y \hat{n}_1 u_{y0} + (c\hat{n}_1/B)\partial_x E_{y0} = 0]]$,
xpnparm = ϵ, eikn1 = $\exp i(k_y y - \omega t)]$.

This response from SFD and SFRSND is structured so that its English meaning when read from the terminal is basically self–explanatory.

The output from SFRSND is then given as input to the routine DIGEST-SOLN, whose function is to absorb this information into the solution net, and into the system's KB. The new information to the solution net comes

from **solutions**: variables previously given only by nodes **solvefor(.,.)** in the net now are given by unravelled expressions. These new expressions are duly attached to the nodes for the variables they describe, these nodes are put into **stack** for treatment by the next call of KITOGEN, and those newly unravelled variables in variable class VC2 are transferred to VC4, completing the bookkeeping mentioned earlier. The additions to the KB come from **new_info**. Thus, the two **givenby(.,.)** forms in **new_info** cause the governing equations for the zero and first-order parts of the problem to be put under the "givenby" property of these newly introduced variables, so that KITOGEN may treat them on the second pass of the problem in just the same fashion as it dealt with the original members of its KB on the first pass. In this sense, PAP is able to "learn," *i.e.*, to expand its KB with new findings in a uniform fashion.

The first item in **solutions** has a somewhat different origin from the rest. In the process of generating the equation hierarchy given in **new_info**, for the first pass of the linear response problem, SFRSND calls subroutine IMPOSE-EQUIL-COND$, which makes use of geometry-specific information, in this case stored under the variable **slab**, to incorporate specializing features of the equilibrium geometry under consideration into the governing equations. This includes use of the symmetry directions, and also may include particular forms normally assumed about certain physical quantities. In this case, $n_0 = n_{00}(x)$ specifies that n_0 is to be given by the background density $n_0 = n_{00}(z)$, hence treated as a known but unspecified function of x, overriding equations from the equation hierarchy which would otherwise be used to solve for it. Thus, the total geometric information used in this calculation is stored under two properties of **slab**. Under the property **sym_dirns** is stored the list $[x, y, z]$, and under the property **subst_list** is stored the substitution $[n_0 = n_{00}(x)]$.

From the **givenby** forms in **new_info** and from Fig. 3, one notes a feature with which CALC has been provided, needed in order to permit it to deal with solving coupled *sets* of equations for *sets* of variables. If the **givenby** forms only specified a single variable v and a single specifying equation **eq**, each **givenby** could simply be implemented in DIGEST-SOLN by the MACSYMA command **put(v,eq,'givenby)**. Since instead sets of coupled variables and equations are [illegible]ed, it is no longer always possible to assign a given new equation to a sp[illegible] new variable. An example is the first order set of equations shown above, in which n_1 and E_{y1} enter on equal footing. Accordingly, such sets of equations are assigned to the **givenby** property

of a "variable-set" variable (in this case, called **vrblset2**), and the set of variables $[E_{y1}, \dot{n}_1]$ is assigned to **vrblset2**'s **vrbls** property. For consistent extension of the KB, therefore, DIGEST-SOLN puts **part(vrblset2,1)** under E_{y1}'s **givenby**, and similarly for $\dot{n}_1$. With this structure, the next pass of KITOGEN, operating just as already described, will correctly link the variable sets and equation sets as illustrated in Fig. 3, without having to make any artificial assignments of individual variables to equations.

This completes the first cycle, in which the equations of the full problem have been treated, resulting in the attachment to the solution net of expressions involving the variables of the zero and first order portions of the problem. The second pass is then entered, and proceeds in the same way. Now, GENSUPRTREE finds two new supernodes **snd2** and **snd3**, as already mentioned. Following the structure of the 'supertree,' SFRSND is first applied to **snd2**, which in this case involves solving only a single differential equation for E_{y0}, carried out by SFD. Then, SFRSND solves **snd3**, which is a simple algebraic system of two linear equations. For linear response calculations such as this one, SFA returns the form

$[E_{y1}, \dot{n}_1]$, INWHICH,

$$\text{givenby}(\omega, \begin{bmatrix} ik_y & -4\pi e \\ c/B\partial_x n_{00}(x) & ik_y u_{y0} - i\omega \end{bmatrix} \cdot \begin{bmatrix} \dot{E}_{y1} \\ \dot{n}_1 \end{bmatrix} = 0)].$$

The equation to be placed under the **givenby** property of ω here is the eigenmode equation, and setting the determinant of the matrix in that equation equal to zero yields the dispersion relation. This information is absorbed by DIGEST-SOLN, and both **stack** and **sfrstack** being empty, CALC returns the final response,

$[\epsilon E_{y1} \exp i(k_y y - \omega t)$, INWHICH,

$$\text{givenby}(\omega, \begin{bmatrix} ik_y & -4\pi e \\ c/B\partial_x n_{00}(x) & ik_y u_{y0} - i\omega \end{bmatrix} \cdot \begin{bmatrix} \dot{E}_{y1} \\ \dot{n}_1 \end{bmatrix} = 0)].$$

Only the final step of the solution procedure outlined at the end of Sec. 2, the actual solution of these equations, has not been carried out, this being within the reach of standard symbolic algebra facilities, or of newer extensions of these (*e.g.*, Sacks' NEWTON program [3], which is able to approximately solve a wide range of algebraic equations, extending the range of problems exactly solvable by the MACSYMA routine SOLVE). CALC returns the

form shown above, which bridges the gap between the problem statement and existing algebraic capabilities, and shows in a perspicuous fashion the information of interest in a linear response problem.

4. Discussion

Regarding CALC as a particular type of problem solver, one notes that it implements the common problem-solving technique of attempting to convert a problem into a set of simpler subproblems, each of which is nearer to "primitive" problems, in this context, those which are immediately solvable by existing automated algebra facilities. We have seen two general ways in which this process of problem reduction is carried out: (a) "*problem transformation*," *i.e.*, introducing a "formal solution," whose form captures something about the nature of the problem, resulting in more tractable equations for the new variables introduced by the formal solution, and (b) "*problem partition*," achieved by studying the topology of the equations in the problem, as represented by the solution net and its covering supernet. These two methods are adequate to allow PAP to deal with both the transport and linear response problems considered so far.

Figuring importantly in these two techniques is the process of *classification* of the equations and variables in the problem. For example, the VCi variable classes are used in applying both the perturbation expansion and eikonal types of problem transformation, as well as the non-formal solution methods employed by SFRSND. The attributes in terms of which this classification is described give the beginnings of a semantic significance, *i.e.*, a *meaning*, to the variables classified. While MACSYMA is indifferent to a variable's significance, the meanings which a scientist attaches to the different variables in a set of relevant equations play a great role in how he thinks about the equations, and what manipulations on these equations he is likely to attempt. The variable classification which CALC uses, and its effect on CALC's solution strategy, models in a simple way this more human way of thinking about doing mathematics.

The variable classification evolved in order to permit CALC to solve systems of equations, and is therefore essentially mathematical in character; it does not yet possess something which might be identified with the *physical* associations which a human attaches to the quantities appearing in physical equations, except insofar as the mathematical role the variables play determines the way people think about them physically. Thus, while a

human has a quite different mental image of mass and electrical inductance, to CALC applied to considering the mathematically identical problems of a damped simple harmonic oscillator and an LRC circuit, they are completely isomorphic, and therefore indistinguishable. This isomorphism is something which would be broken by a more fully developed KB, one from which it would be possible to consider problems in which *both* mechanical and electrical elements are present. Additionally, however, there is in CALC's present operation nothing of the human difference in visualization of these two systems, which probably also significantly affects how a human goes about getting a physical feel for a problem. However, it is unclear in what way two systems, governed by precisely the same mathematics, may be distinguished, the difference in visualization mentioned above perhaps amounting to two representations of the same system, induced by the human sensory apparatus, with different strengths and weaknesses for purposes of analysis.

The current variable classification resides in two places, in the VC1 variable classification, and also in the KB, both in the "permanent" portion KB0, present at the start of PAP's operation, and in the extension dKB, developed in the course of the calculation. An example taken from dKB, illustrating some of the classifying attributes needed for CALC's operation, is the list of properties developed for the first order density n_1 in the course of the calculation:

property	*value*
givenby	$n_1 = \hat{n}_1 \exp i(k_y y - \omega t)$
depends_on	$\{t, x, y\}$
xpansn_coef_of	n
origin	xpansn_coef_of

This may be paraphrased as "n_1 is givenby the (eikonal form ...). It depends upon the VC1 variables t, x, and y. It was created as an expansion coefficient of the variable n." The last attribute here ties n_1 to n, which, in addition to providing needed information to SFRSND for the present problem, also allows n_1 to inherit useful information from n. This helps to fulfill a design criterion for PAP, that, like a human, it shouldn't have to be told the same information more than once.

Related to the issues of variable semantics and classification, it is worth reiterating the central role in CALC's operation played by the solution net and its associated super-net. As already indicated, these objects capture something of what humans think of as the "structure" of a calculation, so

that similar calculations have similar solution net structures. Solution nets thus fill a similar role to the semantic nets used in theories of reasoning and learning by analogy [4], or to automated analysis of case law [5].

Finally, a significant aspect of PAP which has evolved with the present work with CALC, and for which further evolution seems important, is the *language* in terms of which PAP and the user communicate. This should be rich enough that many of the plasma physics questions one might wish to ask can be phrased within it, in a fairly natural format. The mathematical "scripts" illustrated here in a simple way seem to offer significant potential for extension of the PAP language, as do the English-like compound forms (*e.g.*, `[solutions, INWHICH,new_info]`) in terms of which PAP routines can communicate their findings to other routines, and to the user.

Acknowledgments

I am grateful to Charles Karney and Elisha Sacks for informative discussions related to this work, and for their comments on the manuscript. This work supported by U.S.Department of Energy Contract No. DE-AC02-76-CHO3073.

References

[1] H.E. Mynick,"Alpha Particle Effects as a Test Domain for PAP. A Plasma Apprentice Program," *Physica Scripta* **T16**, 133-142 (1987).

[2] See, for example, *Proceedings of the 1979 MACSYMA Users Conference* (Washington, DC, June, 1979).

[3] E.P. Sacks,"An Approximate Solver for Symbolic Equations," (submitted for publication).

[4] P.H. Winston,"Learning New Principles From Precedents and Exercises," *Artificial Intelligence* **19**, 321 (1982).

[5] E.L. Rissland, E.M. Valcarce, K.D. Ashley, "Explaining and Arguing With Examples," *Proceedings of National Conference on Artificial Intelligence* (AAAI-84), Austin, TX (1984).

Figures

Fig. 1. Depiction of the physical situation considered in the sample problem to which CALC is applied. To a plasma density $n_0(x)$ varying in the x direction, and uniform in the y and z directions, is applied a small perturbation n_1, sinusoidal in the y direction.

Fig. 2. Solution net generated by the first call to KITOGEN on the SFM problem. The main goal (at "root" node **nd1**) appears at the top, and subgoals arising from this main task appear below it. The upward directed dashed lines lead to nodes already established at the time of the creation of the "parent" node from which they emanate. The upward directed line labelled **nd2** emanating from **nd13** indicates an arc which goes to **nd2**, not drawn for simplicity of display.

Fig. 3. Complete solution net for the SFM problem, using the same conventions as for Fig. 2, and in addition indicating the three nontrivial supernode groupings.

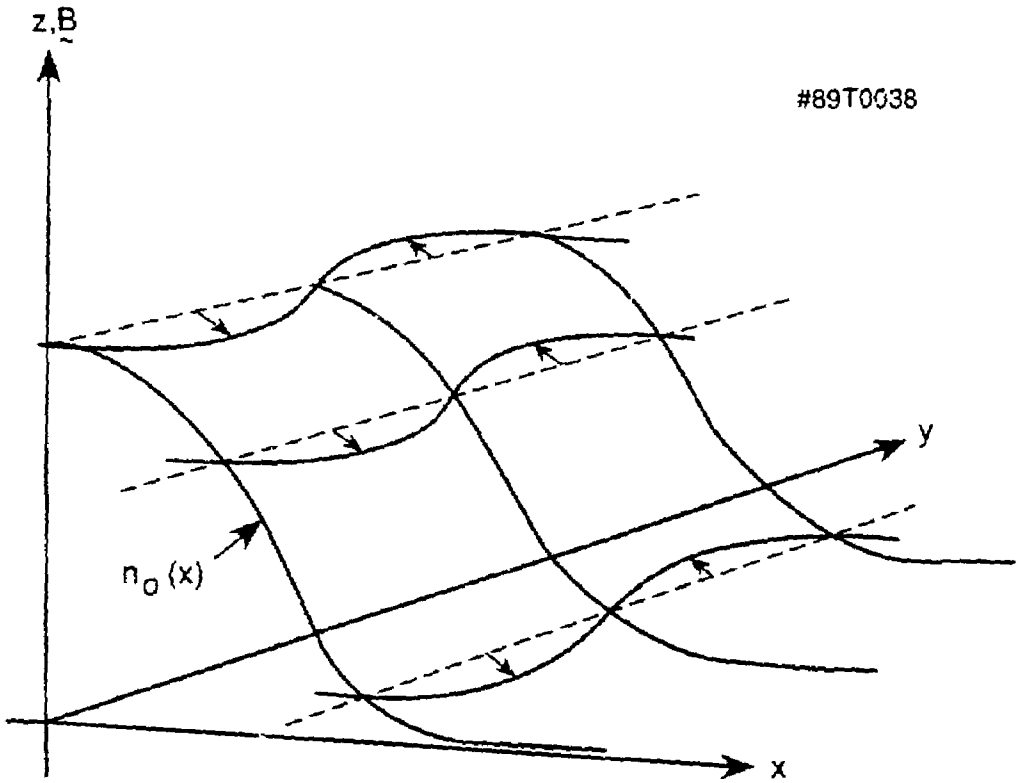


Fig. 1

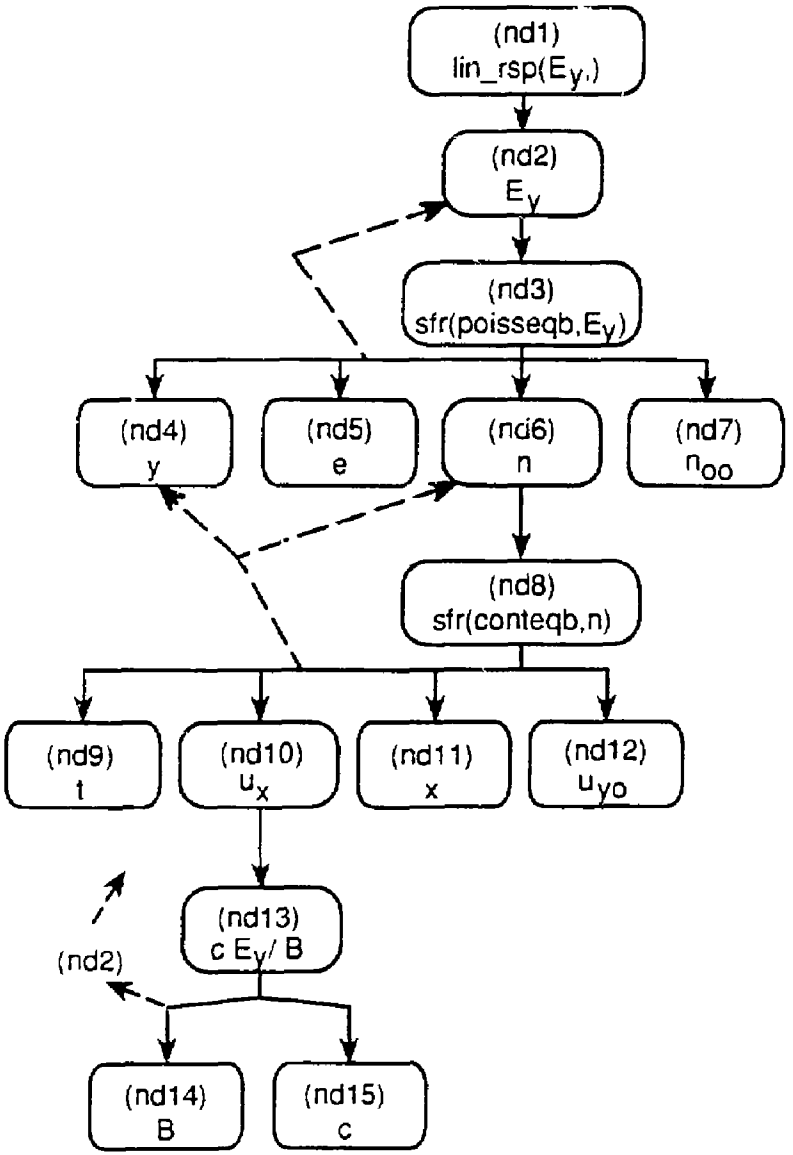

(nd1)
lin_rsp(Ey,)
(nd2)
Ey
(nd3)
sfr(poisseqb,Ey)
(nd4)
y
(nd5)
e
(nd6)
n
(nd7)
noo
(nd8)
sfr(conteqb,n)
(nd9)
t
(nd10)
ux
(nd11)
x
(nd12)
uyo
(nd13)
c Ey/ B
(nd2)
(nd14)
B
(nd15)
c

Fig. 2

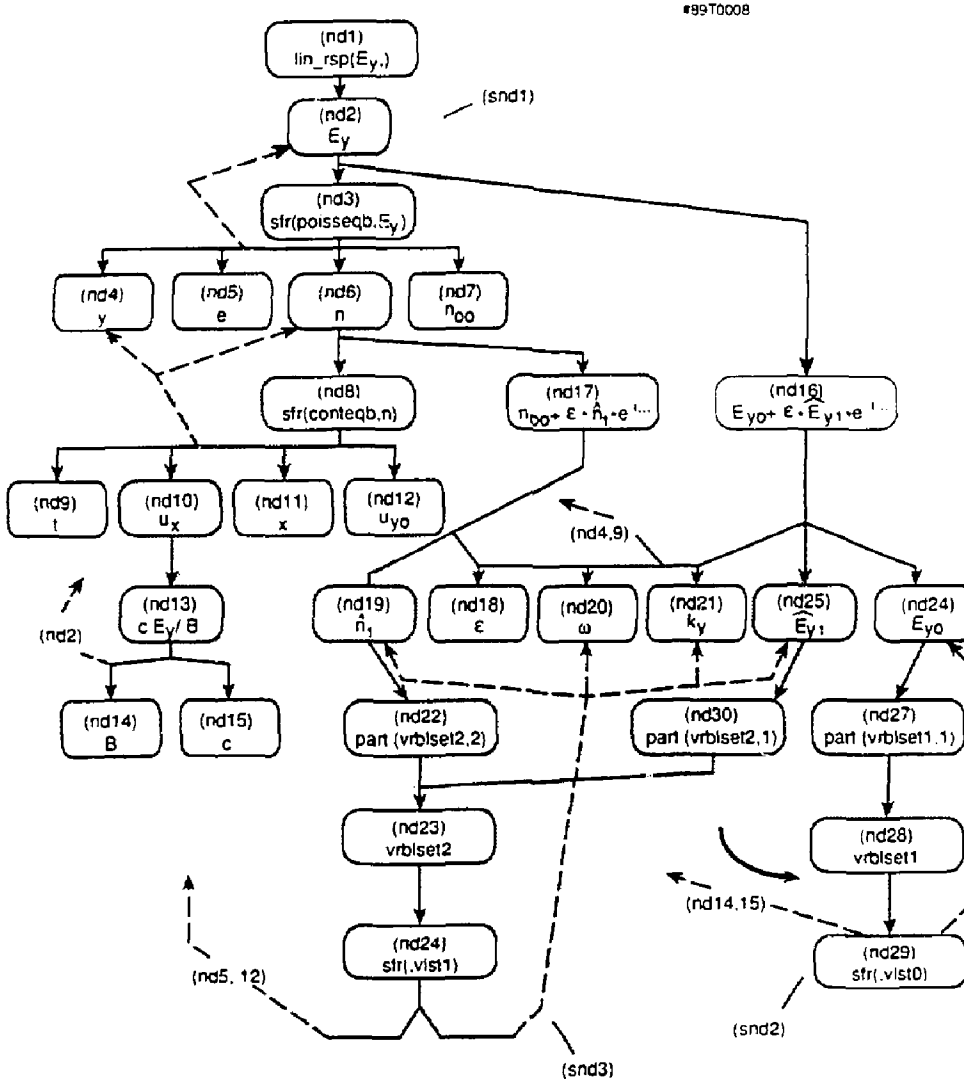

(nd1)
lin_rsp(Ey,)
(snd1)
(nd2)
Ey
(nd3)
sfr(poisseqb,Ey)
(nd4)
y
(nd5)
e
(nd6)
n
(nd7)
noo
(nd8)
sfr(conteqb,n)
(nd17)
noo+ ε · n̂1·e ...
(nd16)
Eyo+ ε · Ey1·e ...
(nd9)
t
(nd10)
ux
(nd11)
x
(nd12)
uyo
(nd4,9)
(nd2)
(nd13)
c Ey/ B
(nd19)
n̂1
(nd18)
ε
(nd20)
ω
(nd21)
ky
(nd25)
Ey1
(nd24)
Eyo
(nd14)
B
(nd15)
c
(nd22)
part (vrblset2,2)
(nd30)
part (vrblset2,1)
(nd27)
part (vrblset1,1)
(nd23)
vrblset2
(nd28)
vrblset1
(nd14,15)
(nd24)
str(,vlst1)
(nd29)
sfr(,vlst0)
(nd5, 12)
(snd2)
(snd3)

Fig. 3

EXTERNAL DISTRIBUTION IN ADDITION TO UC-420

Dr. Frank J. Paoloni, Univ of Wollongong, AUSTRALIA
Prof. M.H. Brennan, Univ Sydney, AUSTRALIA
Plasma Research Lab., Australian Nat. Univ., AUSTRALIA
Prof. I.R. Jones, Flinders Univ., AUSTRALIA
Prof. F. Cap, Inst Theo Phys, AUSTRIA
Prof. M. Heindler, Instut fur Theoretische Physik, AUSTRIA
M. Goossens, Astronomisch Instituut, BELGIUM
Ecole Royale Militaire, Lab de Phys Plasmas, BELGIUM
Commission-European, Dg-XII Fusion Prog, BELGIUM
Prof. R. Boucique, Rijksuniversiteit Gent, BELGIUM
Dr. P.H. Sakanaka, Instituto Fisica, BRAZIL
Instituto De Pesquisas Espaciais-INPE, BRAZIL
Documents Office, Atomic Energy of Canada Limited, CANADA
Dr. W.P. Bachynski, MPB Technologies, Inc., CANADA
Dr. H.M. Skarsgard, University of Saskatchewan, CANADA
Dr. H. Barnard, University of British Columbia, CANADA
Prof. J. Teichmann, Univ. of Montreal, CANADA
Prof. S.R. Sreenivasan, University of Calgary, CANADA
Prof. Tudor W. Johnston, INRS-Energie, CANADA
Dr. Bolton, Centre canadien de fusion magnetique, CANADA
Dr. C.R. James, Univ. of Alberta, CANADA
Dr. Peter Lukac, Komenskeho Univ, CZECHOSLOVAKIA
The Librarian, Culham Laboratory, ENGLAND
The Librarian, Rutherford Appleton Laboratory, ENGLAND
Mrs. S.A. Hutchinson, JET Library, ENGLAND
C. Mouttet, Lab. de Physique des Milieux Ionises, FRANCE
J. Radet, CEN/CADARACHE - Bat 506, FRANCE
Ms. C. Rinni, Librarian, Univ. of Ioannina, GREECE
Dr. Tom Mual, Academy Bibliographic Ser., HONG KONG
Preprint Library, Hungarian Academy of Sciences, HUNGARY
Dr. B. Das Gupta, Saha Inst of Nucl. Phys., INDIA
Dr. P. Kaw, Institute for Plasma Research, INDIA
Dr. Phillip Rosenau, Israel Inst. of Tech, ISRAEL
Librarian, Int'l Ctr Theo Phys, ITALY
Prof. G. Rostagni, Istituto Gas Ionizzati Del CNR, ITALY
Miss Clelia De Palo, Assoc EURATOM-ENEA, ITALY
Dr. G. Grosso, Istituto di Fisica del Plasma, ITALY
Dr. H. Yamato, Toshiba Res & Dev, JAPAN
Prof. I. Kawakami, Atomic Energy Res. Institute, JAPAN
Prof. Kyoji Nishikawa, Univ of Hiroshima, JAPAN
Director, Dept. Large Tokamak Res. JAERI, JAPAN
Prof. Satoshi Itoh, Kyushu University, JAPAN
Research Info Center, Nagoya University, JAPAN
Prof. S. Tanaka, Kyoto University, JAPAN
Library, Kyoto University, JAPAN
Prof. Nobuyuki Inoue, University of Tokyo, JAPAN
S. Mori, JAERI, JAPAN
H. Jeong, Librarian, Korea Advanced Energy Res Inst, KOREA
Prof. D.I. Choi, The Korea Adv. Inst of Sci & Tech, KOREA
Prof. B.S. Liley, University of Waikato, NEW ZEALAND
Institute of Plasma Physics, PEOPLE'S REPUBLIC OF CHINA
Librarian, Institute of Phys., PEOPLE'S REPUBLIC OF CHINA
Library, Tsing Hua University, PEOPLE'S REPUBLIC OF CHINA
Z. Li, Southwest Inst. Physics, PEOPLE'S REPUBLIC OF CHINA
Prof. J.A.C. Cabral, Inst Superior Tecnico, PORTUGAL
Dr. Octavian Petrus, AL I CUZA University, ROMANIA
Dr. Jan de Villiers, Fusion Studies, AEC, SO AFRICA
Prof. M.A. Hellberg, University of Natal, SO AFRICA
C.I.E.M.A.T., Fusion Div. Library, SPAIN
Dr. Lennart Stenflo, University of UMEA, SWEDEN
Library, Royal Institute of Tech, SWEDEN
Prof. Hans Wilhelmson, Chalmers Univ of Tech, SWEDEN
Centre Phys des Plasmas, Ecole Polytech Fed, SWITZERLAND
Bibliotheek, Fom-Inst Voor Plasma-Fysica, THE NETHERLANDS
Metin Durgut, Middle East Technical University, TURKEY
Dr. D.D. Ryutov, Siberian Acad Sci, USSR
Dr. G.A. Eliseev, Kurchatov Institute, USSR
Dr. V.A. Glukhikh, Inst Electrophysical Apparatus, USSR
Prof. O.S. Padchenko, Inst. of Phys. & Tech. USSR
Dr. L.M. Kovrizhnykh, Institute of Gen. Physics, USSR
Nuclear Res. Establishment, Julich Ltd., W. GERMANY
Bibliothek, Inst. Fur Plasmaforschung, W. GERMANY
Dr. K. Schindler, Ruhr-Universitat Bochum, W. GERMANY
ASDEX Reading Rm, c/o Wagner, IPP/Max-Planck, W. GERMANY
Librarian, Max-Planck Institut, W. GERMANY
Prof. R.K. Janev, Inst of Phys, YUGOSLAVIA