JOURNAL OF COMPUTATIONAL PHYSICS **56**, 175–202 (1984)

Symbolic Manipulation Techniques for Plasma Kinetic Theory Derivations

N. T. GLADD

JAYCOR, *P.O. Box 85154, San Diego, California 92138*

Received July 22, 1983; revised December 28, 1983

Much of the analysis of instabilities and wave propagation in warm plasmas involves perturbative solutions of the Vlasov or collisionless Boltzmann equation. In general, a dielectric tensor which describes the response of the plasma to perturbations must be derived. A typical derivation involves choosing an equilibrium distribution function which models some plasma of interest, calculating a perturbed distribution function by solving a partial differential equation (usually by the method of characteristics), calculating density and current moments of this perturbed distribution function, and using these quantities in Maxwell's equations. For many equilibrium models of physical interest, this process can be completely carried out analytically. However, even in simple cases, the procedure is tedious—involving hundreds or thousands of separate algebraic or calculus operations and is fraught with opportunity for error. It will be illustrated here how the symbolic manipulation language MACSYMA can be used to automate many of the steps involved in such derivations. By way of example, the classic problem of oscillations in a homogeneous, uniformly magnetized plasma will be considered.

1. INTRODUCTION

A substantial amount of modern plasma kinetic theory involves perturbative solutions of the Vlasov or collisionless Boltzmann equation,

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f + \frac{q}{m}\left(\mathbf{E} + \frac{\mathbf{v} \times \mathbf{B}}{c}\right) \cdot \nabla_v f = 0, \tag{1}$$

where $f(\mathbf{x}, \mathbf{v}, t)$ is the distribution function of a given plasma species, and $\mathbf{E}(\mathbf{x}, t)$ and $\mathbf{B}(\mathbf{x}, t)$ are electric and magnetic fields. The quantities f, E, and B are self-consistently linked to Maxwell's equations through the moments for charge and current density,

$$\begin{Bmatrix} \rho(\mathbf{x}, t) \\ \mathbf{j}(\mathbf{x}, t) \end{Bmatrix} = \int d^3v \begin{Bmatrix} q \\ q\mathbf{v} \end{Bmatrix} f(\mathbf{x}, \mathbf{v}, t). \tag{2}$$

In the area of plasma stability theory, the typical procedure is to consider oscillatory perturbations about a stationary equilibrium,

0021-9991/84 $3.00

$$\begin{Bmatrix} f(\mathbf{x}, \mathbf{v}, t) \\ \mathbf{E}(\mathbf{x}, t) \\ \mathbf{B}(\mathbf{x}, t) \end{Bmatrix} = \begin{Bmatrix} f_0(\mathbf{x}, \mathbf{v}) \\ E_0(\mathbf{x}) \\ \mathbf{B}_0(\mathbf{x}) \end{Bmatrix} + \begin{Bmatrix} \delta f(\mathbf{x}, \mathbf{v}) \\ \delta\mathbf{E}(\mathbf{x}) \\ \delta\mathbf{B}(\mathbf{x}) \end{Bmatrix} e^{-i\omega t}, \tag{3}$$

choosing the equilibrium $f_0(\mathbf{x}, \mathbf{v})$, $\mathbf{E}_0(\mathbf{x})$, $\mathbf{B}_0(\mathbf{x})$ so as to model the fields and geometry of a particular plasma configuration of interest. The perturbations satisfy a partial differential equation and are usually obtained by means of the method of characteristics. The final result is typically a complicated transcendental equation for ω and attention is focused on analyzing the parametric conditions under which the perturbations are growing in time,

$$\mathrm{Im}(\omega) > 0. \tag{4}$$

In the related problem of wave propagation in a plasma, an eikonal form [$\sim\exp(i\mathbf{k}\cdot\mathbf{x})$] for the spatial variation is assumed, a similar derivation is performed, and the parametric dependence of $\mathbf{k}$ is analyzed.

Conceptually, either procedure is straightforward; operationally, they are quite complex. Much of the "art" of plasma kinetic analysis lies in the ability to choose equilibria which, on the one hand, are complicated enough to yield a physically interesting model plasma and yet, on the other hand, are sufficiently simple to make the various steps leading to the determination of ω or $\mathbf{k}$ analytically tractable. Even when compromises which lead to numerics are required, it is important to proceed analytically as far as possible. To elaborate on these points, consider that the perturbative solution of (1) may be written as

$$\delta f(\mathbf{x}, \mathbf{v}, t) = -\frac{q}{m}\int_{-\infty}^{t} dt' \left[\delta\mathbf{E}(\mathbf{x}', t') + \frac{\mathbf{v}' \times \delta\mathbf{B}(\mathbf{x}', t)}{c}\right] \cdot \nabla_v f_0(\mathbf{x}', \mathbf{v}'), \tag{5}$$

where the integral is to be taken along the trajectory (characteristic) defined by

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \tag{6}$$

$$\frac{d\mathbf{v}}{dt} = \frac{q}{m}\left[\mathbf{E}_0(\mathbf{x}) + \frac{\mathbf{v}\times\mathbf{B}_0(\mathbf{x})}{c}\right]. \tag{7}$$

The perturbed distribution $\delta f(\mathbf{x}, \mathbf{v}, t)$ must also satisfy the perturbed Maxwell equations,

$$\nabla\cdot\delta\mathbf{E}(\mathbf{x}, t) = 4\pi\sum_j q_j\int d^3v\int_{-\infty}^{t} dt' \frac{q_j}{m_j}\left[\delta E(\mathbf{x}', t) + \frac{\mathbf{v}'\times\delta\mathbf{B}(\mathbf{x}', t)}{c}\right]\cdot\nabla'_v f_0(\mathbf{x}', \mathbf{v}'), \tag{8}$$

$$\nabla \times \delta\mathbf{B}(\mathbf{x}, t) = \frac{4\pi}{c} \sum_j q_j \int d^3v\, \mathbf{v} \int_{-\infty}^{t} dt' \frac{q_j}{m_j} \left[\delta E(\mathbf{x}', t) + \frac{\mathbf{v} \times \delta\mathbf{B}(\mathbf{x}', t)}{c} \right] \cdot \nabla_v f_0(\mathbf{x}', \mathbf{v}') + \frac{1}{c} \frac{\partial}{\partial t} \delta\mathbf{B}(\mathbf{x}, t), \tag{9}$$

through the moment relations for perturbed plasma current and density. These constitute a formidable set of coupled integro-differential equations. Success in dealing with this set of equations stems from a judicious choice of equilibrium, extensive use of symmetry properties, and the frequent invocation of physically reasonable approximations. This success manifests itself through closed form solutions of Eqs. (6) and (7) which, in turn, are sufficiently simple to lead to closed form expressions for the integrals in Eqs. (8) and (9).

From the standpoint of this paper, a key point is that even when a straightforward operational procedure exists for going from choice of equilibrium to an analytically or numerically tractable expression for ω or $\mathbf{k}$, the process typically requires hundreds or thousands of individual algebraic and calculus operations.

In this paper the symbolic manipulation language MACSYMA [1] will be used to automate some of the tedious steps in plasma kinetic theory derivations. Operationally, this will consist of systematically evaluating a complicated quadruple integral and making use of simplifying identities for the various special functions which occur.

To place this particular application of symbolic manipulation in perspective, it is useful to consider some of the general ways in which symbolic manipulation can be used to advantage. The most common use of symbolic manipulation is for the explicit generation and manipulation of complicated mathematical expressions. Typically, a complex expression is built up by the actions of replacing simple objects with more complex objects, by multiple additions and multiplications, by the performing of explicit differentiations or by the application of series expansions for objects within the expression. Such mathematical entities, even if very long and complex, can be given simple names and manipulated with ease. This has a dual benefit. First, it lessens the possibility of error by greatly reducing the amount of human labor required. Second, it allows the consideration of many more special cases and circumstances than would be possible if these entities had to be manually generated and manipulated.

Once complicated mathematical expressions have been generated, symbolic manipulation techniques provide various methods of further processing them. The simplest of these is the explicit numerical evaluation of expressions. The benefits here are the same as those mentioned previously—the ability to go quickly to an explicit answer with a greatly lessened chance of error. Within the MACSYMA system it is also possible to examine expressions graphically. This is extremely useful since it allows a quick examination of the parametric "topology" of an expression. It is also useful for checking the validity of approximations introduced in the generation of the expression. A more complicated way of using symbolic manipulation to further process mathematical expressions is the automatic generation of FORTRAN code for the expressions. This allows access to the power of large computer systems and the full range of numerical tools which are available to these systems.

More in the spirit of analysis, symbolic manipulation systems allow the nonnumerical simplification of complicated expressions. This is accomplished by reorganizing the expressions. Simple examples are the gathering of coefficients and the extraction of parts of expressions. In a more fundamental sense, complicated expressions are reorganized by the use of pattern matching. That is, a given expression is scanned for the occurrence of a subexpression for which a known simpler form exists. This simpler form is then substituted into the original expression in place of the pattern which was found. The repetitive application of such a procedure reorganizes and simplifies the original expression. Examples of this procedure are factoring and the use of identities for special functions. In contrast to the generation of expressions, the reorganization of expressions through pattern matching is not an explicit procedure. Many difficult questions present themselves. How should an expression be examined in order to guarantee that imbedded patterns will indeed be found? Once a pattern has been found, which of several possibilities for a simplifying identity should be invoked? What principles should be used to guide the reorganization procedure? In attacking problems of this class the boundary between the techniques of mathematical symbolic manipulation and the techniques of artificial intelligence becomes blurred.

Although the literature on the application of symbolic manipulation techniques to the mathematical problems of interest to plasma physics is sparse, the interested reader can find [2–9] discussion of the use of MACSYMA to perform the various types of symbolic manipulation. In this particular paper, no attempt will be made to apply the methodology of symbolic manipulation to a problem of current research interest. Instead, the methodology required to solve a classic problem of plasma kinetic theory will be illustrated. Specifically, symbolic procedures based on MACSYMA will be used to derive elements of the conductivity tensor describing the response of a homogeneous, uniformly magnetized plasma to small perturbations.

In writing this paper, the author was struck by the difficulty of presenting some of the ideas in a form which is characteristic of most scientific literature. To some extent this is because of the relative newness of the application of symbolic methods to problems in physics. More significant, perhaps, is that while the idea of symbolic manipulation is easily grasped intuitively, the actual ability to perform such manipulations is severely hampered by the arcane structure of symbolic languages. In symbolic manipulation, intuition almost always loses to syntax. Therefore, the author feels that in order to present symbolic methods which can actually be used, it is necessary to go into more detail than would be required to describe a methodology to be implemented in FORTRAN or some other familiar language.

The symbolic methods discussed in this paper were developed to facilitate kinetic theory analyses of the complicated magnetic confinement geometry represented by the Elmo Bumpy Torus (EBT) [10]. The symbolic derivation of dielectric tensors specific to the wave propagation and stability properties of the EBT plasma geometry will be presented in the future.

In Section 2, the kinetic theory derivation of the dielectric response of a homogeneous plasma in a uniform magnetic field will be reviewed, with emphasis placed on detailing the various methematical operations which must be performed. In Section 3, some of the general ideas behind the application of symbolic manipulation to this type of problem will be presented. In Section 4, the symbolic procedures which parallel the analytical derivations detailed in Section 2 will be carried out. In Section 5, a summary and discussion of extensions of the methodology will be presented. Finally, an Appendix follows which gives detailed listings and discussion of the various MACSYMA procedures utilized in the derivation.

2. Analytical Derivation of Model Dielectric Tensor

The derivation of the dielectric tensor for a hot, homogeneous plasma in a uniform magnetic field was first presented by Bernstein in 1958 [11]. Literally hundreds of plasma analyses have invoked the methodology introduced in this seminal paper. In addition to providing a theoretical base for calculations dependent on kinetic effects, this derivation helped to clarify the procedure of including finite temperature effects in fluid theories by providing an independent check of the tedious perturbative procedures required in those calculations. This derivation is now included in standard graduate level plasma texts [12, 13] and is a convenient vehicle for illustrating the use of symbolic manipulation techniques in plasma kinetic theory.

Assuming a stationary, homogeneous plasma immersed in a uniform magnetic field $\mathbf{B} = B_0 \hat{1}_z$, Eq. (1) becomes

$$\frac{q}{m}\left(\frac{\mathbf{v} \times B_0 \hat{1}_z}{c}\right) \cdot \nabla_v f_0(\mathbf{v}) = 0. \tag{10}$$

It is easy to show that any function of the form

$$f_0(v_\perp^2, v_z), \tag{11}$$

where $v_\perp = (v_x^2 + v_y^2)^{1/2}$, satisfies this equation. Following tradition, specifically consider the Maxwellian distribution,

$$f_0(\mathbf{v}) = n_0 \left(\frac{m}{2\pi T}\right)^{3/2} \exp\left[-\frac{m}{2T}(v_\perp^2 + v_z^2)\right]. \tag{12}$$

In this case the perturbed distribution function is given by

$$\delta f(\mathbf{x}, \mathbf{v}, t) = \frac{q f_0(\mathbf{v})}{T} \int_{-\infty}^{t} dt' \left(\delta \mathbf{E}(\mathbf{x}', t') + \frac{\mathbf{v}' \times B_0 \hat{1}_z}{c}\right) \cdot \mathbf{v}'. \tag{13}$$

Note that the last term vanishes for this special case of constant magnetic field and isotropic equilibrium model. The orbits in a uniform magnetic field may be written as

$$v_x = v_\perp \cos(\omega_c \tau + \psi) \tag{14}$$

$$v_y = v_\perp \sin(\omega_c \tau + \psi) \tag{15}$$

$$v_z = \text{constant} \tag{16}$$

$$x' - x(t) = \frac{v_\perp}{\omega_c} [\sin(\omega_c \tau + \psi) - \sin \psi] \tag{17}$$

$$y' - y(t) = -\frac{v_\perp}{\omega_c} [\cos(\omega_c \tau + \psi) - \cos \psi] \tag{18}$$

$$z' - z(t) = v_z \tau, \tag{19}$$

where $\omega_c = qB_0/mc$, $\tau = t' - t$.

At this point an eikonal form for the perturbation is assumed,

$$\delta E, \delta f(\mathbf{x}, \mathbf{v}, t) \sim \delta \hat{E}, \delta \hat{f} \exp[-i\omega t + ik_y y(t) + ik_z z(t)]. \tag{20}$$

The isotropy of the model allows the choice $\mathbf{k} = k_y \hat{1}_y + k_z \hat{1}_z$ without loss of generality. The explicit form of δf becomes

$$\delta \hat{f} = \frac{q f_0}{T} \int_{-\infty}^{0} d\tau [v_\perp \cos(\omega_c \tau + \psi)\, \delta \hat{E}_x + v_\perp \sin(\omega_c \tau + \psi)\, \delta \hat{E}_y + v_z\, \delta \hat{E}_z]$$
$$\times \exp \left\{ -i\omega\tau - ik_y \frac{v_\perp}{\omega_c} [\cos(\omega_c \tau + \psi) - \cos \psi] + ik_z v_z \tau \right\}. \tag{21}$$

Now the τ integral may be explicitly evaluated by making use of the identity,

$$e^{\pm i\gamma \left\{ \begin{smallmatrix} \sin\theta \\ \cos\theta \end{smallmatrix} \right\}} = \sum_l J_l(\gamma)\, e^{\pm il \left\{ \begin{smallmatrix} \theta \\ (\pi/2 - \theta) \end{smallmatrix} \right\}}. \tag{22}$$

After some algebra, terms in Eq. (21) may be arranged so that the τ integrals are of the form

$$\int_{-\infty}^{0} d\tau\, e^{-i\tilde{\omega}\tau} = \frac{1}{-i\tilde{\omega}}, \tag{23}$$

where, for reasons of causality, $\tilde{\omega}$ is assumed to have a small positive imaginary part to eliminate the contribution at $\tau = -\infty$. The perturbed distribution function evaluates to

$$\delta\hat{f} = i\frac{qf_0}{T}e^{+i(k_y v_\perp/\omega_c)\cos\phi}\sum_l J_l\left(\frac{k_y v_\perp}{\omega_c}\right)e^{-il(\pi/2)}$$
$$\times\left\{\frac{(v_\perp/2)[\delta\hat{E}_x - i\,\delta\hat{E}_y]\,e^{i(l+1)\phi}}{\omega-(l+1)\,\omega_c - k_z v_z} + \frac{(v_\perp/2)[\delta\hat{E}_x + i\,\delta\hat{E}_y]\,e^{i(l-1)\phi}}{\omega-(l-1)\,\omega_c - k_z v_z}\right.$$
$$\left.+\frac{v_z\,\delta\hat{E}_z\,e^{il\phi}}{\omega - l\omega_c - k_z v_z}\right\}. \tag{24}$$

Next, a change of dummy summation index is made in order to write δf in the more convenient form

$$\delta\hat{f} = i\frac{qf_0}{T}e^{+i(k_y v_\perp/\omega_c)\cos\phi}\sum_l e^{il\phi}$$
$$\times\left\{\frac{v_\perp}{2}[\delta\hat{E}_x + i\,\delta\hat{E}_y]\,J_{l-1}\left(\frac{k_y v_\perp}{\omega_c}\right)e^{-i(l-1)(\pi/2)}\right.$$
$$+\frac{v_\perp}{2}[\delta\hat{E}_x + i\,\delta\hat{E}_y]\,J_{l+1}\left(\frac{k_y v_\perp}{\omega_c}\right)e^{-i(l+1)(\pi/2)}$$
$$\left.+\,v_z\,\delta\hat{E}_z\,J_l e^{-il(\pi/2)}\right\}\Big/(\omega - l\omega_c - k_z v_z). \tag{25}$$

Invoking the Bessel summation identities,

$$J_{l+1}(\gamma) + J_{l-1}(\gamma) = \frac{2l}{\gamma}J_l(\gamma) \tag{26}$$

$$J_{l-1}(\gamma) - J_{l+1}(\gamma) = 2J_l'(\gamma), \tag{27}$$

leads to

$$\delta\hat{f} = i\frac{qf_0}{T}e^{+i(k_y v_\perp/\omega_c)\cos\phi}\sum_l e^{-il((\pi/2)-\phi)}$$
$$\times\left[\frac{iv_\perp J_l'(k_y v_\perp/\omega_c)\,\delta\hat{E}_x + (l\omega_c/k_y)\,J_l(k_y v_\perp/\omega_c)\,\delta\hat{E}_y + v_z J_l(k_y v_\perp/\omega_c)\,\delta\hat{E}_z}{\omega - l\omega_c - k_z v_z}\right], \tag{28}$$

a closed form for the perturbed distribution function.

The derivation proceeds with the calculation of the perturbed plasma current,

$$\delta\mathbf{j} = \int d^3v\,q\mathbf{v}\,\delta\hat{f}(v). \tag{29}$$

The desired dielectric tensor for the plasma response is

$$\vec{\epsilon} = 1 + \frac{4\pi i}{\omega}\vec{\sigma}, \tag{30}$$

where $\vec{\vec{\sigma}}$ is the conductivity tensor which may be extracted from the perturbed current through the constitutive relationship,

$$\delta \mathbf{j} = \vec{\vec{\sigma}} \cdot \delta \mathbf{E}. \tag{31}$$

For the Maxwellian equilibrium distribution function, Eq. (12), the explicit expression for the perturbed current is

$$\begin{aligned} \delta \hat{\mathbf{j}} = & \left(i\frac{q^2 n_0}{T}\right)\left(\frac{m}{2\pi T}\right)^{3/2} \int_{-\infty}^{\infty} dv_z \exp\left(-\frac{mv_z^2}{2T}\right) \int_0^{\infty} v_\perp dv_\perp \exp\left(-\frac{mv_\perp^2}{2T}\right) \\ & \times \int_0^{2\pi} d\psi (v_\perp \cos\psi \hat{1}_x + v_\perp \sin\psi \hat{1}_y + v_z \hat{1}_z) \\ & \times \sum_l \left[\frac{iv_\perp J_l' \,\delta\hat{E}_x + (l\omega_c/k_y) J_l \,\delta\hat{E}_y + v_z J_l \,\delta\hat{E}_z}{\omega - l\omega_c - k_z v_z}\right] \\ & \times e^{-il((\pi/2)-\psi)} e^{i(k_y v_\perp/\omega_c)\cos\psi}. \end{aligned} \tag{32}$$

To illustrate the mathematical steps involved in evaluating this triple integral, it is sufficient to consider only one component—say δj_x. Specifically, the ψ integral

$$I_\psi = \int_0^{2\pi} d\psi \cos\psi e^{-il((\pi/2)-\psi)} e^{i(k_y v_\perp/\omega_c)\cos\psi} J_l\left(\frac{k_y v_\perp}{\omega_c}\right) \tag{33}$$

is evaluated by using the identity Eq. (22) and

$$\int_0^{2\pi} d\psi\, e^{i(l-p)\psi} = 2\pi \delta_{l,p} \tag{34}$$

to obtain

$$I_\psi = \sum_p \pi J_p \left(\frac{k_y v_\perp}{\omega_c}\right) e^{-i(l-p)(\pi/2)} (\delta_{p,l+1} + \delta_{p,l-1}). \tag{35}$$

Then, upon summation and the use of the Bessel identity Eq. (27)

$$I_\psi = -2i\pi J_l' \left(\frac{k_y v_\perp}{\omega_c}\right). \tag{36}$$

The x-component of the perturbed current is then given by

$$\begin{aligned} \delta \hat{j}_x = \hat{1}_x & \left(i\frac{q^2 n_0}{T}\right)\left(\frac{m}{2\pi T}\right)^{3/2} \int_{-\infty}^{\infty} dv_z\, e^{-mv_z^2/2T} \int_0^{\infty} v_\perp dv_\perp\, e^{-mv_\perp^2/2T} \\ & \times \sum_l (-2i\pi v_\perp J_l') \left[\frac{iv_\perp J_l' \,\delta\hat{E}_x + (l\omega_c/k_y) J_l \,\delta\hat{E}_y + v_z J_l \,\delta\hat{E}_z}{\omega - l\omega_c - k_z v_z}\right]. \end{aligned} \tag{37}$$

The $v_\perp$-integrations are variations of Weber's second exponential integral

$$\int_0^\infty dt\, t \exp(-p^2t^2)\, J_\nu(at)\, J_\nu(bt) = \frac{1}{2p^2} \exp\left(-\frac{a^2+b^2}{4p^2}\right) I_\nu\left(\frac{ab}{2p^2}\right). \tag{38}$$

Introducing the explicit notation,

$$2\int_0^\infty x\, dx\, e^{-x^2} \begin{Bmatrix} J_l^2(ax) \\ xJ_l'(ax)\, J_l(ax) \\ [xJ_l'(ax)]^2 \end{Bmatrix} = \begin{Bmatrix} \Gamma_{l,0}(b) \\ \Gamma_{l,1}(b) \\ \Gamma_{l,2}(b) \end{Bmatrix}, \tag{39}$$

where $\Gamma_{l,0} = I_l(b)\, e^{-b}$, $b = a^2/2$, and $\Gamma_{l,1}(b)$, $\Gamma_{l,2}(b)$ may be expressed in terms of $\Gamma_{l,0}(b)$, leads to

$$\begin{aligned} \delta \hat{f}_x = \hat{1}_x \left(\frac{2}{\sqrt{\pi}} \frac{q^2 n_0}{m}\right) \int_{-\infty}^\infty dV_z\, e^{-V_z^2} \\ \times \sum_l \left[\frac{\Gamma_{l,2}(b)(i\, \delta\hat{E}_x) + (l\omega_c/k_y\bar{v})\, \Gamma_{l,1}(b)\, \delta\hat{E}_y + V_z \Gamma_{l,1}(b)\, \delta\hat{E}_z}{\omega - l\omega_c - k_z\bar{v}V_z}\right], \end{aligned} \tag{40}$$

where $V_z = v_z/\bar{v}$, $\bar{v} = (2T/m)^{1/2}$. The remaining integral is related to the error function of imaginary argument—the well-known plasma dispersion function. Using the notation

$$\int_{-\infty}^\infty dx\, e^{-x^2} \frac{x^{(\alpha-1)}}{x-\zeta} = \sqrt{\pi}\, Z_\alpha(\zeta), \tag{41}$$

the final result for $\delta\hat{f}$ may be written

$$\begin{aligned} \delta\hat{f}_x = \hat{1}_x \left(\frac{-2q^2 n_0}{mk_z\bar{v}}\right) \sum_l \Big[\Gamma_{l,2}(b)\, Z_1(\zeta_l)(i\, \delta\hat{E}_x) \\ + \left(\frac{l\omega_c}{k_y\bar{v}}\right) \Gamma_{l,1}(b)\, Z_1(\zeta_l)\, \delta\hat{E}_y + \Gamma_{l,1}(b)\, Z_2(\zeta_l)\, \delta\hat{E}_z\Big], \end{aligned} \tag{42}$$

where $\zeta_l = (\omega - l\omega_c)/k_z\bar{v}$. The elements σ_{xx}, σ_{xy}, and σ_{xz} may be extracted from Eq. (42) by inspection. The calculation of the other components of σ can be carried out in an analogous fashion, and Eq. (30) may be used to explicitly write the elements of the dielectric tensor.

Note that even for the very simple model of a homogeneous plasma in a uniform magnetic field, the procedure for deriving the dielectric response of the plasma to perturbations is a tedious multistep process. In carrying out this procedure it was necessary to make multiple use of the identities,

$$e^{\pm ia\cos\theta} = \sum_l J_l(a)\, e^{\pm il((\pi/2)-\theta)} \tag{43}$$

$$\int_{-\infty}^{0} d\tau \, e^{-i\omega\tau} = \frac{1}{-i\omega} \tag{44}$$

$$\sum_{l} \frac{J(l)}{\omega - (l \pm 1)\,\omega_c} = \sum_{p} \frac{J(p \mp 1)}{\omega - p\omega_c} \tag{45}$$

$$J_{l+1}(a) \pm J_{l-1}(a) = \begin{Bmatrix} \dfrac{2l}{a} J_l(a) \\ -2J_l'(a) \end{Bmatrix} \tag{46}$$

$$\int_{0}^{2\pi} d\theta \, e^{i(l-m)\theta} = 2\pi\delta_{l,m} \tag{47}$$

$$\sum_{m} f(m)\,\delta_{l,m} = f(l) \tag{48}$$

$$2\int_{0}^{\infty} x\,dx\,e^{-x^2} \begin{Bmatrix} J_l^2(ax) \\ xJ_l'(ax)\,J_l(ax) \\ [xJ_l'(ax)]^2 \end{Bmatrix} = \begin{Bmatrix} \Gamma_{l,0}(b) \\ \Gamma_{l,1}(b) \\ \Gamma_{l,2}(b) \end{Bmatrix} \tag{49}$$

$$\int_{-\infty}^{\infty} dx \frac{e^{-x^2} x^{(\alpha-1)}}{x - \zeta} = \sqrt{\pi}\, Z_\alpha(\zeta). \tag{50}$$

More realistic equilibrium models of plasmas add greatly to the complexity and tedium of this procedure.

3. Some General Aspects of the Symbolic Derivation

In this section, some general aspects of the procedure by which the MACSYMA symbolic manipulation language may be used to duplicate the multistep derivation detailed in the previous section are discussed. No attempt is made to explain all of the MACSYMA commands invoked since that language has literally hundreds of commands. Detailed explanations should be sought in the MACSYMA user's manual [1].

To some extent, MACSYMA has an English-like structure and may be understood as read. In the author's opinion, this is very deceptive for attempts to "logically" deviate from well-trodden paths almost always lead to difficulties which are hard to resolve. The frequent frustration experienced when this happens is a prime reason why the powerful tool of symbolic manipulation has been so slowly accepted by the scientific and engineering community.

When beginning to learn MACSYMA, progress is often painfully slow (at least, in the author's experience). Then, once some feeling for syntax has been obtained and a sufficient number of tricks have been learned, progress becomes much more rapid. The word "trick" is used deliberately since the act by which a procedure, which has proved almost impossible to the novice, is accomplished by an expert in a few keystrokes has, quite seriously, been named the "Svengali" effect [14].

The MACSYMA procedures utilized here are not novel. The intent is to splice elementary features of MACSYMA together in order to systematically attack a complex class of physics problems. In addition, there has been no attempt at optimization and every choice between elegance and clarity has been decided in favor of the latter.

The task is to develop a set of MACSYMA procedures which will sequentially perform the mathematical operations required in the derivation detailed in the previous section. The essence of this task is the development of subprocedures which scan mathematical expressions and recognize certain patterns for which replacement rules exist. MACSYMA has a built-in procedure for pattern recognition—DEFMATCH—but this procedure often fails if the pattern is complex or if the pattern which is sought is buried in a large expression. Therefore, steps must be taken to facilitate the pattern searching as much as possible.

This is accomplished by adopting a standard notation for the various variables and parameters, by suppressing the use of explicit notation indicating summation and integration, and, most importantly, by breaking complicated expressions into a sum of terms and examining each term individually. These simplifying methods will be discussed first before passing to the central core of the problem—recognizing specific mathematical patterns and replacing them with simplifying identities.

3.1. *Standard Notation*

The perturbed distribution function, $\delta\hat{f}$, Eq. (21) has the functional dependence

$$\delta\hat{f} = \delta\hat{f}\{q, T, f_0, v_\perp, v_z, \psi, \omega_c, \tau, \omega, k_y, k_z, \delta E_x, \delta E_y, \delta E_z\}, \tag{51a}$$

$$f_0 = f_0(n_0, m, T, v_\perp, v_z). \tag{51b}$$

The corresponding MACSYMA notation is

$$\text{DELTAF(Q, T, FO, VPR, VPL, PSI, OMC, T, OM, } K_Y, K_Z, E_X, E_Y, E_Z) \tag{52}$$

$$\text{FO(NO, M, T, VPR, VPL)} \quad . \tag{53}$$

The velocity variables, VPR and VPL, are actually dimensionless: $\text{VPR} \equiv v_\perp/\bar{v}$, where $\bar{v} = (2T/m)^{1/2}$, with $\text{VTH} \equiv \bar{v}$. In the course of the derivation, additional dimensionless variables and functions are introduced—

$$a = \frac{k_y \bar{v}}{\omega_c} \equiv A, \qquad \zeta_l = \frac{\omega - l\omega_c}{k_z \bar{v}} \equiv \text{ZETA}[L],$$
$$J_l(x) \equiv J[L](x), \qquad \Gamma_{l,j}(x) \equiv \text{GAM}[L, J](X), \tag{54}$$
$$Z_j(\zeta_l) \equiv \text{ZFCN}[J](\text{ZETA}[L])$$

where quantities enclosed by [ ] appear as subscripts.

To simplify pattern searches and the appearance of expressions, the use of summation and integration notation is suppressed. For example,

$$e^{ia\cos\theta} = \sum_l J_l(a)\, e^{il((\pi/2)-\theta)} \tag{55}$$

will appear as

```
   %I A COS(THETA)                 %I L (%PI/2 - THETA)
%E                  = J (A) %E
                       L
```
(56)

Summation indices are L and P, and the integration variables are T, PSI, VPR, and VPL.

3.2. *Breaking Complicated Expressions into Parts*

To deal with the simplifying procedure of breaking a complicated expression into terms which may be operated on individually, recursion is chosen rather than iteration. In MACSYMA, the generic recursive procedure—OPERATION—which simplifies—EXPRESSION—is written as

```
     OPERATION(EXPRESSION):=BLOCK(
[other_needed_local_variables     ],
EXPRESSION:EXPAND(EXPRESSION),
IF EXPRESSION_IS_SUM(EXPRESSION)
  THEN
    RETURN(OPERATION(FIRST(EXPRESSION))
          +OPERATION(REST(EXPRESSION))),


procedures_for_simplifying_a_single_term,


RETURN(   simplified_form_of_single_term   ));
```
(57)

In Eq. (57) the word BLOCK identifies OPERATION as a separate procedure. EXPAND guarantees that EXPRESSION appears as a sum of terms. EXPRESSION_IS_SUM is a Boolean function which tests EXPRESSION to see if it is a single term or a sum of terms (see Appendix A). If it is a sum, then OPERATION is called recursively on the parts of EXPRESSION. If EXPRESSION is a single term, then the required simplifications are performed. Most of the procedures developed for the kinetic theory derivation follow the general pattern of Eq. (57).

3.3. *Pattern Recognition*

The key aspect of simplifying a given expression involves recognizing a mathematical pattern which may be replaced with a known simpler form. A generic MACSYMA procedure for accomplishing pattern matching could probably be written, but the idea is best illustrated by a simple example. Suppose the values A, B, C of the expression

$$AX^B \exp(CX), \tag{58}$$

are to be determined. The MACSYMA procedure would be

```
MATCH_PATTERN(EXPRESSION):=BLOCK(
[A,B,C,TEMPLATE],
MATCHDECLARE([A,B,C],FREEOF(X)),
DEFMATCH(TEMPLATE,A*X^B*EXP(C*X)),
TEMPLATE(EXPRESSION),
RETURN([A,B,C]));                                    (59)
```

Here MATCHDECLARE ascribes the property that A, B, C be independent of X, DEFMATCH sets up the explicit pattern matcher—TEMPLATE—which is then applied to EXPRESSION. As an explicit example—

```
match_pattern(x^(7/2)*exp(-%i*x));
```

returns

```
        7
[1, -, - %I]                                         (60)
        2
```

In the case of more complicated patterns it is usually necessary to further decompose EXPRESSION in order to achieve a successful matching.

3.4. *Checking for Errors*

As will become clear in what follows, the expressions which are being manipulated become very long and complicated. The question naturally arises as to how to check their correctness. In a basic sense, the question of the correctness of intermediate expressions is no different than the question of the correctness of the intermediate numbers in a complicated numerical calculation. It is only our experience in dealing with humanly manageable symbolic expressions that makes us think that we should be able to check their correctness at every stage in the calculation. In practice, our typical check on correctness is to see if a given expression is consistent with its immediate predecessor. Emphasis should be placed on checking the correctness of the final expressions and, in general, the procedures used are analogous to those used in numerical computation.

Some specific techniques in which the correctness of this plasma kinetic theory derivation was checked are as follows. First, the program was made as modular as possible and each module was debugged and checked independently. Second, the starting expressions were carefully typed, checked, and stored, once and for all, on disk. This eliminated the errors which could occur due to repetitive typing. Third, the overall procedure was checked by constructing artificially simpler starting equations which nevertheless contained all of the generic features required in the pattern matching sequence. Fourth, the entire problem treated here is, in essence, an error check since it may be performed analytically. If the techniques developed in solving this known problem are found to yield the correct results, then great confidence is gained in applying this code to starting equations which are more complicated but involve no new generic procedures.

In the case of problems which do not lead to known answers, then standard techniques of analysis could be used to check the answers. For example, limiting cases could be extracted and checked against known results. Another error checking procedure would be to solve the problem by an alternate procedure and compare the two results. This latter method is much more feasible in the realm of computational symbolic manipulation than it is in a manual calculation. The amount of time required to set up an alternative computational procedure for carrying out a calculation is usually only a small fraction of the time required to manually carry out a calculation.

4. Structure of the Specific MACSYMA Kinetic Theory Derivation

MACSYMA is usually thought of as an interactive language whereby expressions are typed in and processed immediately with the operator having various options after each procedure. However, in this case, a complex but well-structured calculation is being performed and it is desirable to avoid errors associated with typing large expressions. Therefore, the operation is performed in a batch mode where basic expressions are stored on disk files, then read in as required and sequentially processed. This also allows the storage of intermediate results so that if (when) trouble occurs it is not necessary to repeat the entire calculation.

Arbitrarily, the calculation is divided into two stages. Stage 1 specifies the perturbed distribution function and evaluates the time integral. Stage 2 specifies which moment will be calculated and performs the three required velocity integrals. These stages are schematically represented in Figs. 1 and 2. To illustrate the symbolic derivation, the steps involved are shown for a simplified expression. Specifically, only the δE_x component of δf

$$\delta f_x = \frac{q f_0}{T} \int_{-\infty}^{0} d\tau \, v_x' e^{ik_y(y'-y) + ik_z(z'-z) - i\omega\tau} \tag{61}$$

is considered. Having performed this integral in Stage 1, Stage 2 is illustrated by considering only the j_x moment of δf_x

$$\delta j_{xx} \equiv \int d^3v \, v_x \, \delta f_x. \tag{62}$$

This is equivalent to calculating only the element σ_{xx} of the conductivity tensor. The explicit statements of Stage 1 (see Fig. 1) are

```
BATCHLOAD([PAPER,FCNS]);
BATCH(PAPER,DELF);
DELTAFX:EV(EV(SUBST([E[X]=1,E[Y]=0,E[Z]=0],DELTAF)));
DELTAFX2:EXP_TRIG_TO_BES(DELTAFX,OMC*T+PSI,COS(OMC*T+PSI),L)$
DELTAFX2:SUBST([K[Y]=A*OMC/VTH],DELTAFX2);
DELTAFX2:SUBST([J[L](A*VPR)=J[L](AVPR)],DELTAFX2);
DELTAFX3:TAU_INTEGRATION(DELTAFX2,T);
DELTAFX4:MAKE_COM_DENOM(DELTAFX3,P)$
DELTAFX4:SUBST([P=L],DELTAFX4);
DELTAFX4:FACTOR(DELTAFX4);
DELTAFX5:SIMPLIFY_JBES_COMBOS(DELTAFX4,3);
DELTAFX5:FACTOR(DELTAFX5);
STRINGOUT([PAPER,RSTAG1],%);&
```

Each of the statements is now discussed in turn.

```
BATCHLOAD([PAPER,FCNS]);
```
(63)

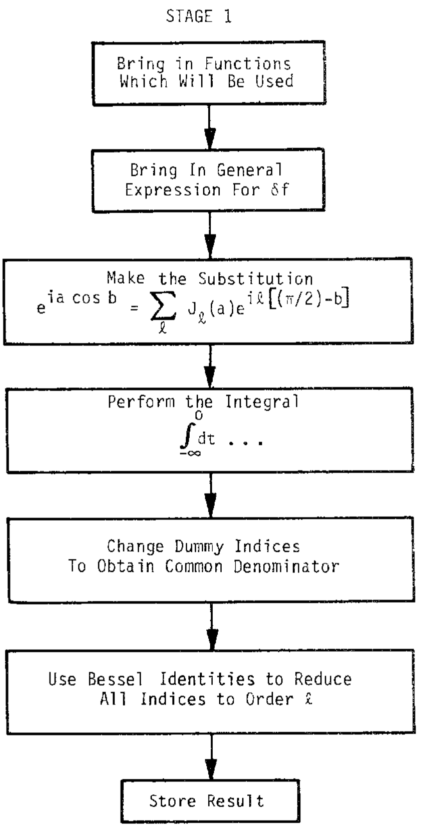

FIG. 1. Stage 1—Performing the orbit integral.

This "quietly" loads the various functions which will be needed, that is, without explicitly displaying those functions. Each function is shown and discussed in Appendix A.

```
BATCH(PAPER,DELF);
```
(64a)

This loads the expressions for δf.

```
deltaf:(q*fo/tmp)*(v[x]*e[x]+v[y]*e[y]+v[z]*e[z])*eikonal$

     eikonal:exp(-%i*om*t+%i*k[y]*dely+%i*k[z]*delz)$

     dely:-(vpr*vth/omc)*(cos(omc*t+psi)-cos(psi))$

     delz:vpl*vth*t$

     V[X]:VPR*VTH/2*(EXP(%I*(OMC*T+PSI))+EXP(-%I*(OMC*T+PSI)))

     V[Y]:VPR*VTH/(2*%I)*(EXP(%I*(OMC*T+PSI))-EXP(-%I*(OMC*T+PSI)))

     v[z]:vth*vpl$
```
(64b)

```
DELTAFX:EV(EV(SUBST([E[X]=1,E[Y]=0,E[Z]=0],DELTAF)));
```
(65a)

This forms the expression δf_x above. The expression must be evaluated—EV—twice in order to force substitutions for various subquantities in DELTAF to be made.

```
                 %I (OMC T + PSI)      - %I (OMC T + PSI)
     FO Q (%E                   + %E                  ) VPR VTH
                       %I (COS(OMC T + PSI) - COS(PSI)) VPR VTH K
                                                                  Y
   %I T VPL VTH K  - ------------------------------------------ - %I OM T
                 Z                        OMC
  %E
 /(2 TMP)
```
(65b)

As it stands, this MACSYMA output is rather difficult to read. The confusion arises from the fact there is no explicit indication that the first line (ending with VPR VTH) is to multiply the second line (starting with %E). This is because multiplication is implicitly indicated in MACSYMA output while other operations are explicitly indicated. The reader should be wary of this inconsistency with "normal" mathematical expressions in reading the formulae to follow.

```
DELTAFX2:EXP_TRIG_TO_BES(DELTAFX,OMC*T+PSI,COS(OMC*T+PSI),L)$
DELTAFX2:SUBST([K[Y]=A*OMC/VTH],DELTAFX2);
DELTAFX2:SUBST([J[L](A*VPR)=J[L](AVPR)],DELTAFX2);
```
(66a)

This function implements the identity Eq. (22). Simplifying substitutions are then made before displaying the result.

```
    J (AVPR) FO Q VPR VTH EXPT(%E, (%I OMC T VPL VTH K
     L                                               Z

                                               2
+ %I A OMC COS(PSI) VPR + (%I OMC  - %I OM OMC) T + %I OMC PSI)/OMC

                                 %PI
- %I L (- OMC T - PSI + ---))/(2 TMP) + J (AVPR) FO Q VPR VTH
                                 2              L

EXPT(%E, (%I OMC T VPL VTH K  + %I A OMC COS(PSI) VPR
                            Z

          2                                                                  %PI
+ (- %I OMC  - %I OM OMC) T - %I OMC PSI)/OMC - %I L (- OMC T - PSI + ---))
                                                                              2

/(2 TMP)
```

(66b)

This expression illustrates why only a single term in $\delta\hat{f}$ is displayed. The phenomenon is called "intermediate expression swell" and is a serious problem in symbolic manipulations.

```
DELTAFX3:TAU_INTEGRATION(DELTAFX2,T);
```

(67a)

This function carries out the time integration.

```
                          2 %I A COS(PSI) VPR + (2 %I L + 2 %I) PSI - %I %PI L
                          ----------------------------------------------------
                                                  2
J (AVPR) FO Q VPR %E                                                              VTH
 L
-------------------------------------------------------------------------------------
            2 TMP (%I VPL VTH K  + (%I L + %I) OMC - %I OM)
                               Z
                             2 %I A COS(PSI) VPR + (2 %I L - 2 %I) PSI - %I %PI L
                             ----------------------------------------------------
                                                     2
 + J (AVPR) FO Q VPR %E
 VTH/(2 TMP (%I VPL VTH K  + (%I L - %I) OMC - %I OM))
                         Z
```

(67b)

```
DELTAFX4:MAKE_COM_DENOM(DELTAFX3,P)$
DELTAFX4:SUBST([P=L],DELTAFX4);
DELTAFX4:FACTOR(DELTAFX4);
```

(68a)

This function changes the dummy summation index in order to form a common denominator for the two terms. The result is then simplified further before presentation.

```
      - (J      (AVPR) - J      (AVPR)) FO Q VPR
          L + 1           L - 1

                                        %I %PI L
   %I A COS(PSI) VPR + %I L PSI - --------
                                           2
 %E                                               VTH

/(2 TMP (VPL VTH K  + L OMC - OM))
                  Z
```

(68b)

```
DELTAFX5:SIMPLIFY_JBES_COMBOS(DELTAFX4,3);
DELTAFX5:FACTOR(DELTAFX5);
```
(69a)

This function evokes the Bessel identities in Eqs. (26) and (27) to reduce the order of the Bessel functions,

```
                                                                     %I %PI L
                                     %I A COS(PSI) VPR + %I L PSI - --------
   d                                                                    2
(----- (J (AVPR))) FO Q VPR %E                                                VTH
 dAVPR    L
-------------------------------------------------------------------------------
                         TMP (VPL VTH K  + L OMC - OM)
                                       Z
```
(69b)

and forms the final Stage 1 expression. This expression should be compared with the coefficient of δE_x in Eq. (28).

```
STRINGOUT([PAPER,RSTAG1],%);
```
(70)

This function saves the last result to disk.

The explicit statements of Stage 2 (see Fig. 2), which perform the velocity integrals are

```
BATCHLOAD([PAPER,FCNS]);
BATCH(PAPER,RSTAG1);
DELTAFX:%TH(2);
DELTAJX:Q*VTH*VPR*(1/2)*(EXP(%I*PSI)+EXP(-%I*PSI))*DELTAFX;
DELTAJX:VTH^3*VPR*DELTAJX;
DELTAJX2:EXP_TRIG_TO_BES(DELTAJX,PSI,COS(PSI),P);
DELTAJX2:SUBST([J[P](A*VPR)=J[P](AVPR)],DELTAJX2);
DELTAJX3:PSI_INTEGRATION(DELTAJX2,PSI,P);
DELTAJX3:SIMPLIFY_JBES_COMBOS(DELTAJX3,3);
FO:NO*(1/(%PI))^(3/2)*(1/VTH^3)*EXP(-VPR^2-VPL^2);
DELTAJX3:SUBST([J[X](A*VPR)=J[L](AVPR)],DELTAJX3);
deltajx3:subst([diff(j[l](avpr),avpr)=djl],deltajx3);
deltajx3:subst([j[l](avpr)=jl],deltajx3);
deltajx3:subst([avpr=a*vpr],deltajx3);
deltajx3:subst([jl=j[l](avpr),djl=diff(j[l](avpr),avpr)],deltajx3);
DELTAJX3:EV(DELTAJX3);
DELTAJX4:VPR_INTEGRATION(DELTAJX3);
DELTAJX5:VPL_INTEGRATION(DELTAJX4);
STRINGOUT([PAPER,RSTAG2],%);
```

These statements are now discussed in detail.

```
BATCHLOAD([PAPER,FCNS]);
BATCH(PAPER,RSTAG1);
DELTAFX:%TH(2);
```
(71a)

These commands load the library of procedures detailed in Appendix A, load the results of the Stage 1 calculation, and assign the name DELTAFX to that result. The cryptic command %TH(2) means: take the expression two lines back. BATCH command leaves a message as its last line and this must be avoided.

```
                                                                     %I %PI L
                                     %I A COS(PSI) VPR + %I L PSI - --------
   d                                                                    2
(----- (J (AVPR))) FO Q VPR %E                                                VTH
 dAVPR    L
-------------------------------------------------------------------------------
                         TMP (VPL VTH K  + L OMC - OM)
                                       Z
```
(71b)

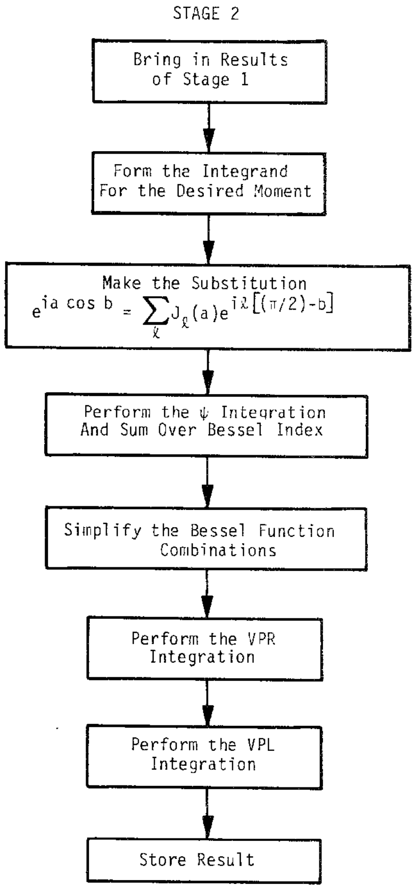

FIG. 2. Stage 2—Performing the moment integrals.

```
DELTAJX:Q*VTH*VPR*(1/2)*(EXP(%I*PSI)+EXP(-%I*PSI))*DELTAFX;
DELTAJX:VTH^3*VPR*DELTAJX;
```
(72a)

The first command sets up the j_x moment of δf_x, i.e., $q\bar{v}v_x\,\delta f_x = q\bar{v}v_\perp \cos\psi\,\delta f_x$; $\cos\psi$ is written in exponential form, anticipating that integrals of the form $\int d\psi\, e^{ic\psi}$ are to be performed. The second command completes the specification of the integrand. Recall the integration is being performed in cylindrical coordinates, with dimensionless variables like $v_\perp = \text{VPR}*\text{VTH}$, and usual integration notation $\int_a^b dx$ is being suppressed.

```
    d                         %I PSI       - %I PSI   2     3
  (----- (J (AVPR))) FO (%E          + %E          ) Q   VPR
   dAVPR   L

                                    %I %PI L
 %I A COS(PSI) VPR + %I L PSI - --------
                                       2
%E

/(2 TMP (VPL VTH K  + L OMC - OM))
                  Z
```
(72b)

```
DELTAJX2:EXP_TRIG_TO_BES(DELTAJX,PSI,COS(PSI),P);
DELTAJX2:SUBST([J[P](A*VPR)=J[P](AVPR)],DELTAJX2);
```
(73a)

This command invokes the identity, Eq. (22),

```
                        d
     J (AVPR) (----- (J (AVPR))) FO
      P        dAVPR   L

    (2 %I L + 2 %I) PSI - %I %PI L
    ------------------------------ + %I P (%PI/2 - PSI)
                  2                                           2    3    5
 %E                                                          Q  VPR  VTH

/(2 TMP VPL VTH K  + 2 L OMC TMP - 2 OM TMP)
                 Z

                      d
 + J (AVPR) (----- (J (AVPR))) FO EXPT(%E,
    P        dAVPR   L

(2 %I L - 2 %I) PSI - %I %PI L           %PI          2    3    5
------------------------------ + %I P (--- - PSI)) Q  VPR  VTH
              2                           2

/(2 TMP VPL VTH K  + 2 L OMC TMP - 2 OM TMP)
                 Z
```
(73b)

```
DELTAJX3:PSI_INTEGRATION(DELTAJX2,PSI,P);
```
(74a)

This command performs two functions: (a) it performs the ψ integration obtaining, in this case, the delta functions $\delta_{l,p\pm 1}$ and (b) it performs the P summation over these delta functions.

```
                           d                   2    3    5
2 %I %PI J      (AVPR) (----- (J (AVPR))) FO Q  VPR  VTH
          L + 1         dAVPR   L
---------------------------------------------------------
       2 TMP VPL VTH K  + 2 L OMC TMP - 2 OM TMP
                      Z

                                    d                   2    3    5
             2 %I %PI J      (AVPR) (----- (J (AVPR))) FO Q  VPR  VTH
                       L - 1         dAVPR   L
         - ---------------------------------------------------------
                    2 TMP VPL VTH K  + 2 L OMC TMP - 2 OM TMP
                                   Z
```
(74b)

```
DELTAJX3:SIMPLIFY_JBES_COMBOS(DELTAJX3,3);
```
(75a)

The order of the Bessel functions is again reduced.

```
                     d                  2     2    3    5
         2 %I %PI (----- (J (AVPR)))  FO Q  VPR  VTH
                  dAVPR   L
       - ----------------------------------------------
              TMP VPL VTH K  + L OMC TMP - OM TMP
                           Z
```
(75b)

```
FO:NO*(1/(%PI))^(3/2)*(1/VTH^3)*EXP(-VPR^2-VPL^2);
```
(76a)

This introduces an explicit form for the equilibrium distribution functions. This substitution is made as late in the calculation as possible to reduce the size of the expressions which must be used in previous manipulations.

```
               2       2
         - VPR   - VPL
    NO %E
    -----------------
          3/2     3
       %PI     VTH
```
(76b)

```
[illegible]
deltajx3:subst([avpr=a*vpr],deltajx3);
deltajx3:subst([jl=j[l](avpr),djl=diff(j[l](avpr),avpr)],deltajx3);
DELTAJX3:EV(DELTAJX3);
```
(77a)

A series of substitutions are required here because of the inability of MACSYMA to deal with derivatives with respect to functions.

```
                                                         2         2
                  d                  2       2      3    - VPR   - VPL       2
    2 %I %PI (------ (J (AVPR)))  NO Q   VPR   %E                    VTH
               dAVPR    L
  - ----------------------------------------------------------------------
                3/2
           %PI     (TMP VPL VTH K  + L OMC TMP - OM TMP)
                                 Z
```
(77b)

```
DELTAJX4:VPR_INTEGRATION(DELTAJX3);
```
(78a)

This performs the $v_\perp$ integration. The method used for this integration is different than that used for previous integrations—see Appendix A for details.

```
                                  2
                                 A        2    2
                    %I GAM     (--) NO Q   VTH
                          L, 2 2
  - ------------------------------------------------------------
                             2                                2
                          VPL                              VPL
    SQRT(%PI) (TMP VPL %E      VTH K  + (L OMC - OM) TMP %E      )
                                    Z
```
(78b)

```
DELTAJX5:VPL_INTEGRATION(DELTAJX4);
```
(79a)

This performs the v_z integration and finishes the calculation. Compare with Eq. (42) [TMP=MVTH^2/2].

```
               2
              A         2                L OMC - OM
    %I GAM   (--) NO Q   VTH ZFCN (- ----------)
          L, 2 2                  1      VTH K
                                              Z
  - ---------------------------------------------
                          TMP K
                               Z
```
(79b)

```
STRINGOUT([PAPER,RSTAG2],%);
```

This stores the result to disk.

5. Summary and Discussion

In the preceding, techniques of symbolic manipulation using MACSYMA have been applied to the complicated multistep derivations which characterize the kinetic theory of plasma stability and wave propagation analyses. The analytical derivation of a classic problem in plasma kinetic theory—the dielectric response of a warm, homogeneous strongly magnetized plasma to small perturbations—was reviewed and mathematical procedures involved in this derivation were summarized. Next, a general framework was discussed through which the complicated expressions which characterized the derivation could be simplified to the point where the pattern matching facilities currently available in MACSYMA could be used. Next, the sequence of MACSYMA procedures which duplicated the analytical derivation were detailed with a restriction, for reasons of compactness to a single representative term of the general 3×3 dielectric. Finally, in an Appendix, the actual procedures which were used to match mathematical patterns and replace them with simplifying identities were listed and discussed.

This work was undertaken as a learning experience for the author and as a prelude to attacking the derivations expected in developing a kinetic theory of stability and wave propagation in the confinement geometry Elmo Bumpy Torus [10].

There are numerous ways in which the methodology presented here can be extended. In point of fact, the problem explicitly treated here is only a prototype of a kinetic theory derivation. There are two general ways in which such derivations become more complex. First is the inclusion of more plasma geometry and physical effects within the context of the linear perturbation theory. Second is the extension of the perturbation theory to higher order, thus entering the realm of nonlinear physics. Both of these avenues have been extensively explored in the past twenty years and there remains much to be done. A major obstacle in these analyses is the same one which is amply illustrated in the prototype derivation presented here—an enormous amount of tedious calculation is required to reach the expressions which serve as the effective starting point of an analysis of interest.

In the case of linear calculations, the complicating factors come in various forms. The inclusion of more geometry increases the complexity of the equilibrium distribution function about which the perturbation theory is built. This means many more terms in the starting expressions and also introduces terms requiring much more complex pattern matching than was required by the prototypical calculation presented here. The fields associated with complex geometry also greatly complicate the orbits of the plasma particles. In general, these complicated orbits may not be written in closed form and perturbation theories must be developed to approximate them by forms which make further calculation tractable. If multiple plasma species are present, the number of terms in a calculation can balloon rapidly. Furthermore, the approximate techniques used to treat the different species are often quite different. If the species interact through collisions, then the basic equation for the evolution of the distribution function becomes inhomogeneous. In this case, it is usually necessary to solve the equation for the starting equilibrium distribution function by some perturbative technique in order to proceed analytically. A basic feature of "realistic" plasma stability calculations is the presence of several nested perturbation calculations within the basic derivation. Indeed, the final dispersion equation. once obtained, is itself typically solved by multiple perturbation schemes. There are many opportunities to automate the straightforward but tedious aspects of these calculations.

The extension of the perturbation theory to higher order—the so-called "weak turbulence" approach—can lead to an enormous explosion of the number of terms. Physical and geometrical approximations are usually introduced at a very early stage in the derivation in order to cull these terms. Frequently, strong restrictions on the frequency and wave number spectrum to be examined have to be introduced in order to proceed. Furthermore, there exist various ways of truncating the hierarchial expansion and these can lead to quite different results. In some cases, the leading terms in the theory are chosen in an almost hueristic fashion. There is much to be done in the way of systematic examinations of these nonlinear theories but researchers are deterred by the monumental algebraic requirements of such analyses. Computational symbolic manipulation makes the systematic examination of some nonlinear systems quite feasible.

In a more far-reaching sense, it would be of considerable interest to attempt to couple the techniques of computational symbolic manipulation with the heuristic techniques of artificial intelligence in order to mimic the operational procedures of "expert" plasma theorists in performing these perturbation calculations. The methodologies used by these experts can be quite subtle but they are not numerous. In point of fact, the requisite knowledge domain is probably relative narrow, consisting of a procedural knowledge of a few perturbation techniques of applied mathematics, knowledge of a modest number of identities relating special functions and the specification of some heuristics indicating the techniques to apply in different situations. The goal is always the same—a "simple" dispersion equation—and the starting point is always the same set of equations. Expert systems have been developed in the much more knowledge-rich domains of medicine and geology.

APPENDIX A: Discussion of MACSYMA Procedures

In this Appendix, the various MACSYMA procedures, which have been developed to perform explicit tasks in kinetic theory derivations are discribed.

```
expression_is_sum(expression):=block(
if not atom(expression) and part(expression,0)="+"
  then
    return(true)
  else
    return(false));
```
(A1)

This procedure tests EXPRESSION to see if it is a sum or a single term.

```
exp_tris_to_bes(expression,ars,tris_fon,bes_index):=block(
[a,b,c,sign,phase,continue,exptris_template],
expression:expand(expression),
if expression_is_sum(expression)
  then
    return(exp_tris_to_bes(first(expression),ars,tris_fon,bes_index)
          +exp_tris_to_bes(rest(expression),ars,tris_fon,bes_index)),
matchdeclare([a,b,c],freeof(tris)),
matchdeclare(ars,true),
defmatch(exptris_template,c*exp(a*tris_fon+b),ars),
exptris_template(expression,ars),
a:-%i*a,
if not atom(a) and part(a,0)="-"
  then
    sign:-1
  else
    sign:1,
if tris_fon=sin(ars)
  then
    phase:ars
  else
    phase:%pi/2-ars,
return(c*exp(b)*j[bes_index](sign*a)
      *exp(%i*sign*bes_index*phase)));                                  (A2)
```

This is a pattern matching routine of the general form discussed in Section 3. It invokes the identity Eq. (22).

```
tau_integration(expression,var):=block(
[a,b,c,expvar_template],
expression:expand(expression),
if expression_is_sum(expression)
  then
    return(tau_integration(first(expression),var)
          +tau_integration(rest(expression),var)),


matchdeclare(var,true),
matchdeclare([a,b,c],freeof(var)),
defmatch(expvar_template,c*exp(a*var+b),var),
expvar_template(expression,var),
return(c*exp(b)/a));                                                    (A3)
```

This routine performs integrations of the form Eq. (23).

```
make_com_denom(expression,nex_index):=block(
[work,coef_omc,coef_l,soln_for_l,idum],
expression:expand(expression),
if expression_is_sum(expression)
  then
    return(make_com_denom(first(expression),new_index)
          +make_com_denom(rest(expression),new_index)),


work:denom(expression),
for idum
  while
    hipow(work,omc)>1
      do
        work:ratsimp(work/omc),
coef_omc:coeff(expand(work),omc,1),
coef_l:coeff(expand(coef_omc),l,1),
if coef_l # 1
  then
    coef_omc:coef_omc/coef_l,
soln_for_l:solve(coef_omc_new_index,l),
soln_for_l:first(soln_for_l),
return(subst(soln_for_l,expression)));                                  (A4)
```

This routine changes the dummy summation index l so as to achieve a common denominator of the form $(\omega - l\omega_c - k_z v_z)$. It duplicates the procedure which transforms Eq. (24) into Eq. (25). As written, it is somewhat more general than needed for the explicit example treated in this paper. Specifically, the DO loop acts to remove common multipliers OMC which may be in the denominator of EXPRESSION and will confuse the search for the coefficient of OMC in the pattern OM–(...)OMC–KZXVTH*VPL.

```
SIMPLIFY_JBES_COMBOS(EXPR,ITER):=BLOCK(
[LIST_OF_INDICES,MAX_INDEX,MIN_INDEX,COEF,SSUM,DDIF,
RULELIST,SUMRULE,DIFRULE],
ITER:ITER+1,
IF ITER > 5
   THEN
     ERROR(TOO_MANY_ITERATIONS),
EXPR:EXPAND(EXPR),
LIST_OF_INDICES:MAKE_LIST_OF_INDICES(EXPR),
MAX_INDEX:APPLY(MAX,LIST_OF_INDICES),
MIN_INDEX:APPLY(MIN,LIST_OF_INDICES),
IF MAX_INDEX = L
   THEN
     RETURN(EXPR),


COEF[MAX_INDEX]:COEFF(EXPR,J[MAX_INDEX](ARG)),
COEF[MIN_INDEX]:COEFF(EXPR,J[MIN_INDEX](ARG)),
SSUM:RATSIMP(1/2*(COEF[MAX_INDEX]+COEF[MIN_INDEX])),
DDIF:RATSIMP(1/2*(COEF[MAX_INDEX]-COEF[MIN_INDEX])),
RULELIST:BES_SUB_RULES(MAX_INDEX),
SUMRULE:FIRST(RULELIST),
DIFRULE:FIRST(REST(RULELIST)),
EXPR:EXPR+SSUM*SUMRULE+DDIF*DIFRULE,
EXPR:SUBST([J[MAX_INDEX](ARG) = 0,
J[MIN_INDEX](ARG) = 0],EXPR),
SIMPLIFY_JBES_COMBOS(EXPR,ITER))$
```
(A5)

This procedure simplifies combinations of Bessel function sums and differences so as to reduce the order of all Bessel functions to L. It is also more general than is needed for our example and is designed to handle the complex combinations which occur in more realistic problems. Basically, it replaces the combination $J_{l+n} \pm J_{l-n}$ with lower order quantities. It invokes the procedures, MAKE_LIST_OF_INDICES, GRAB_INDEX, and BES_SUB_RULES.

```
MAKE_LIST_OF_INDICES(EXPR):=BLOCK(
[LIST_OF_INDICES,I,IMAX,WORK,INDEX,]
LIST_OF_INDICES:[],
EXPR:EXPAND(EXPR),
IMAX:LENGTH(EXPR),
FOR I THRU IMAX DO
   (IF PART(EXPR,0) = "+"
     THEN
     (WORK:FIRST(EXPR),
      EXPR:REST(EXPR))
     ELSE WORK:EXPR,
  INDEX:GRAB_INDEX(WORK),
  IF NOT MEMBER(INDEX,LIST_OF_INDICES)
     THEN
       LIST_OF_INDICES:CONS(INDEX,LIST_OF_INDICES),


RETURN(LIST_OF_INDICES));
```
(A6)

This procedures passes EXPRESSION and makes a list of all the Bessel indices appearing in it.

```
GRAB_INDEX(EXPR):=BLOCK(
[A,INDEX],
MATCHDECLARE([A,INDEX],FREEOF(ARG)),
DEFMATCH(JMATCH,A*J[INDEX](ARG)),
JMATCH(EXPR),
IF FREEOF(L,INDEX)
   THEN
     RETURN(L)
   ELSE
      RETURN(INDEX));
```
(A7)

This procedure is invoked by MAKE_LIST_OF_INDICES. It determines the index of the Bessel function appearing in an expression.

```
BES_SUB_RULES(INDEX):=BLOCK(
[N],
IF INDEX = L+1
   THEN RETURN([2*L/ARG*J[L](ARG),-2*DIFF(J[L](ARG),ARG)])
   ELSE
     (N:INDEX-L,
     RETURN([-(J[L+N-2](ARG)+J[L-N+2](ARG))+2*(L+N-1)/ARG*J[L+N-1](ARG)+2*(L-N+
1)/ARG*J[L-N+1](ARG),-(J[L+N-2](ARG)-J[L-N+2](ARG))+2*(L+N-1)/ARG*J[L+N-1](ARG)
-2*(L-N+1)/ARG*J[L-N+1](ARG)])))$
```
(A8)

This procedure provides the Bessel substitution rules

$$J_{l+1}+J_{l-1}=\frac{2l}{x}J_l(x); \qquad l\pm 1,$$
$$J_{l+1}-J_{l-1}=-2_lJ'(x); \qquad l\pm 1, \tag{A9}$$

$$J_{l+n}\pm J_{l-n}=-(J_{l+n-2}\pm J_{l-n+2})+\frac{2(l+n)}{x}J_{l+n+1}$$
$$\pm\frac{2(l-n)}{x}J_{l-n+1}; \qquad l\pm n, \tag{A10}$$

when evoked by SIMPLIFY_JBES_COMBOS.

```
psi_integration(expression,var,bes_index):=block(
[aa,b,c,expvar_template,soln_for_bes_index],
expression:expand(expression),
if expression_is_sum(expression)
  then
    return(psi_integration(first(expression),var,bes_index)
          +psi_integration(rest(expression),var,bes_index)),


matchdeclare(var,true),
matchdeclare([aa,b,c],freeof(var)),
defmatch(expvar_template,c*exp(aa*var+b),var),
expvar_template(expression,var),
soln_for_bes_index:solve(aa,bes_index),
soln_for_bes_index:first(soln_for_bes_index),
return(subst(soln_for_bes_index,2*%pi*c*exp(b))));
```
(A11)

This procedure performs the integration

$$\int_0^{2\pi} d\psi\, e^{i[p-f(l)]} = 2\pi\delta_{p,f(l)} \tag{A12}$$

and then performs the sum over the implied summation index P, invoking the Kronecker-δ as required.

```
vpr_integration(expression):=block(
[term1,term2,test,sub,num_tests,com_term,work,work2],
expression:expand(expression),
if expression_is_sum(expression)
  then
    return(vpr_integration(first(expression))
          +vpr_integration(rest(expression))),


term1:j[1](avpr),  term2:vpr*diff(j[1](avpr),avpr),
test[1]:term1^2,  sub[1]:gam[1,0](a^2/2)/2,
test[2]:term1*term2,  sub[2]:gam[1,1](a^2/2)/2,
test[3]:term2^2,  sub[3]:gam[1,2](a^2/2)/2,
num_tests:3,
com_term:vpr*exp(-vpr^2),
work:ratsimp(expression/com_term),
for i:1 thru num_tests do
  (work2:ratsimp(work/test[i]),
   if freeof(vpr,work2)
     then
       (work2:work2*sub[i],  i:num_tests)),


return(work2));
```
(A13)

This procedure performs the VPR integration. It illustrates a different method of performing the integration. Rather than attempting a pattern match, it simply trys a sequence of substitution rules. This is quite practical in the case where it is known that the integrand must be one or another of a few related forms.

```
vpl_integration(expression):=block(
[aa,bb,cc,nn,work,num_work,den_work,com_term,
  linear_template,poly_template],
expression:expand(expression),
if expression_is_sum(expression)
  then
    return(vpl_integration(first(expression))
          +vpl_integration(rest(expression))),


com_term:exp(-vpl^2),
work:ratsimp(expression/com_term),
num_work:num(work),
den_work:denom(work),
matchdeclare([aa,bb,cc,nn],freeof(vpl)),
defmatch(linear_template,aa*vpl+bb),
defmatch(poly_template,cc*vpl^nn),
linear_template(den_work),
poly_template(num_work),
return(ratsimp(cc/aa)*sqrt(%pi)*zfcn[nn+1](ratsimp(-bb/aa))));
```
(A14)

This procedure performs the VPL integration. It illustrates the method of matching a complex pattern by removing a common factor, and then matching separately on the numerator and the denominator.

Acknowledgments

The author would like to thank Ellen Golden, Jeff Golden, Leo Harten, and Bob Berman of MIT for their help with MACSYMA. He would also like to acknowledge Bruce Curtis of the National Magnetic Fusion Energy Computing Center for his general assistance with symbolic computing. This research was done with the aid of MACSYMA, a large symbolic manipulation program developed at the MIT Laboratory for Computer Science and supported from 1975 to 1983 by the National Aeronautics and Space Administration under Grant NSG-1323, by the Office of Naval Research under Grant N00014-77-C-0641, by the U.S. Department of Energy under Grant ET-78-C-02-4687, and by the U.S. Air Force under Grant F49620-79-C-020, and since 1982 by Symbolics, Inc. of Cambridge, Massachusetts. MACSYMA is a trademark of Symbolics, Inc. This work was supported by the U.S. Department of Energy.

Note added in proof. Interested readers are referred to the recently published, excellent exposition on MACSYMA: "Computer Algebra in Applied Mathematics: An Introduction to MACSYMA," R. H. Rand (Pitman, 1984).

References

1. "MACSYMA Reference Manual," Mathlib Group, Laboratory for Computer Science, Massachusetts Institute of Technology, Version Ten, 1983.
2. A. Bers, J. L. Kulp, and C. F. F. Karney, *Comput. Phys. Comm.* **12** (1976), 81.
3. J. L. Kulp, Ph.D. thesis, Massachusetts Institute of Technology.
4. C. F. F. Karney, "Proceedings of the MACSYMA Users' Conference," NASA, Berkeley, California, 1977.
5. G. O. Cook, Jr., Ph.D. thesis, Brigham Young University; Lawrence Livermore Laboratory Report UCRL-53324, 1982.
6. M. C. Wirth, Lawrence Livermore Laboratory Report UCRL-52996; Ph.D. thesis, 1980.
7. B. Char, "Proceedings, Second MACSYMA Users' Conference," 1979.
8. B. Char and B. McNamara, "Proceedings, Second MACSYMA Users' Conference," 1979.
9. E. L. Lafferty, "Proceedings, Second MACSYMA Users' Conference," 1979.
10. R. A. Dandl and the EBT Group, Oak Ridge National Laboratory Report No. ORNL/TM-6457, 1978.
11. I. B. Bernstein, *Phys. Rev.* **109** (1958), 10.
12. N. A. Krall and A. W. Trivelpiece, "Principles of Plasma Physcis," McGraw–Hill, New York, 1973.
13. S. Ichimaru, "Basic Principles of Plasma Physics," Benjamin, New York, 1973.
14. R. Berman, "Workshop on the Use of Symbolic Manipulation in Plasma Physics," Los Alamos, N.M., 1982.