Normal Modes of a Loss Cone Plasma Slab with Steep Density Gradient

M. J. Gerver and C. K. Birdsall

Department of Electrical Engineering and Computer Sciences
and Electronics Research Laboratory
University of California, Berkeley, California 94720

A. B. Langdon and D. Fuss
Lawrence Livermore Laboratory
University of California, Livermore, California 94550

The main text of this report (excluding the appendices), with minor changes, has been accepted for publication in Physics of Fluids.

Normal Modes of a Loss Cone Plasma Slab with Steep Density Gradient

M. J. Gerver and C. K. Birdsall
Department of Electrical Engineering and Computer Sciences
and Electronics Research Laboratory
University of California, Berkeley, California 94720

A. B. Langdon and D. Fuss
Lawrence Livermore Laboratory
University of California, Livermore, California 94550

ABSTRACT

The electrostatic flute-like normal modes were found for a collisionless loss cone plasma in a uniform magnetic field with density varying sinusoidally in one direction in space. A local method, appropriate to $kL \gg 1$, where k is the wave number and L the scale length, and a nonlocal method, appropriate to $kL \lesssim 1$, were both used; the ion Larmor radius a_i was not assumed to be small. The usual drift cone mode was found when $a_i \ll L$; other instabilities were found for $a_i \gtrsim L$. In principle, the methods could be applied to other configurations of plasma.

Research sponsored by the Energy Research and Development Administration Contract E(04-3)-34-PA128.

I. Introduction

In analyzing the kinetic instabilities of a non-uniform plasma the assumption is frequently made[1,2] that the Larmor radius is small compared to the scale length of the plasma, although in many experiments[3] the ion Larmor radius a_i is comparable to the scale length L. We have found that it is often possible to find the normal modes, and even to use a local approximation, without assuming $a_i \ll L$. The procedure starts with the integrodifferential Vlasov-Poisson dispersion relation for the electrostatic modes of a non-uniform collisionless plasma. Two different approximate methods can be used to solve this integral equation, reducing it to either a differential equation or a matrix equation. Each method is valid in a different regime, but neither method requires $a_i \ll L$. The two methods have been used to find the properties of the most unstable flute-like mode of a deuterium loss cone plasma with a density varying sinusoidally in space in a direction perpendicular to a uniform magnetic field, for a variety of densities, density gradients, and mirror ratios. (The sinusoidally varying density was used because it simplifies the calculations and because it has been used in computer simulations of the drift cone mode[4,5].) The numerical results of the two methods are in good agreement (typically within 10%) in the regime where both are expected to be valid, thus increasing our confidence in them.

In Sec. II the integrodifferential dispersion relation will be derived for an arbitrary configuration of plasma and magnetic field, as well as for the special case of a uniform magnetic field, and we will discuss the reasons for the usual assumption that $a_i \ll L$, and under what circumstances this assumption can be dispensed with. In Sec. III the two methods of solving the integral equation will be described and

applied to a loss cone plasma with sinusoidally varying density. The range of validity of each method will be found. In Sec. IV the numerical results obtained by the two methods will be given, and the numerical techniques used will be briefly described. In Sec. V the physical significance of these results (particularly for $a_i \gtrsim L$) will be discussed, and a summary and conclusions will be presented in Sec. VI.

II. Dispersion Relation

The linearized Vlasov equation for a plasma with an electrostatic perturbation is

$$\frac{df_{1s}}{dt}(\underset{\sim}{x},\underset{\sim}{v},t) = -\frac{q_s}{m_s}\nabla\phi(\underset{\sim}{x},t)\cdot\frac{\partial f_{0s}}{\partial \underset{\sim}{v}}(\underset{\sim}{x},\underset{\sim}{v}) \tag{1}$$

where f_{1s} is the first-order perturbation in the distribution function, f_{0s} is the zero-order equilibrium distribution function (satisfying $\partial f_{0s}/\partial t = 0$), q_s is the charge and m_s is the mass for species s; and ϕ is the perturbed potential. Eq. (1) can be integrated over the zero-order particle orbits to obtain

$$f_{1s}(\underset{\sim}{x},\underset{\sim}{v},t) = \frac{q_s}{m_s}\int_{-\infty}^{t} d\tau\ \nabla\phi[\underset{\sim}{x}_s'(\tau),\tau]\cdot\frac{\partial f_{0s}}{\partial \underset{\sim}{v}_s}[\underset{\sim}{x}_s'(\tau)\underset{\sim}{v}_s'(\tau)] \tag{2}$$

where $\underset{\sim}{x}_s'(\tau)$ is the unperturbed orbit of a particle of species s, and $\underset{\sim}{v}_s'(\tau) \equiv d\underset{\sim}{x}_s'/d\tau$, with $\underset{\sim}{x}_s'(t) = \underset{\sim}{x}$ and $\underset{\sim}{v}_s'(t) = \underset{\sim}{v}$.

Eq. (2) is combined with the Poisson equation

$$\sum_s 4\pi q_s \int f_{1s}(\underset{\sim}{x},\underset{\sim}{v},t)d\underset{\sim}{v} + \nabla^2\phi(\underset{\sim}{x},t) = 0$$

to yield the well-known result[6]

$$\sum_s \frac{4\pi q_s^2}{m_s} \int d\underset{\sim}{v} \int_{-\infty}^{t} d\tau \, \nabla\phi(\underset{\sim}{x}_s', \tau) \cdot \frac{\partial f_{os}}{\partial \underset{\sim}{v}_s'}(\underset{\sim}{x}_s', \underset{\sim}{v}_x') + \nabla^2\phi(\underset{\sim}{x},t)=0 \quad (3)$$

Eq. (3) can be fourier transformed in $\underset{\sim}{x}$: (See Appendix A)

$$\int d\underset{\sim}{k} \, \tilde{\phi}(\underset{\sim}{k}) \left(\int d\underset{\sim}{k}' \exp(i\underset{\sim}{k}' \cdot \underset{\sim}{x}) \left\{ \sum_s \frac{4\pi q_s^2}{m_s} \int d\underset{\sim}{v} \int_{-\infty}^{0} d\tau \right.\right.$$

$$\left.\left. i\underset{\sim}{k} \cdot \frac{\partial \tilde{f}_s}{\partial \underset{\sim}{v}_s}(\underset{\sim}{k}-\underset{\sim}{k}', \underset{\sim}{v}_s') \exp[i\underset{\sim}{k}' \cdot (\underset{\sim}{x}_s' - \underset{\sim}{x}) - i\omega\tau] \right\} - k^2 \right) = 0 \quad (4)$$

where $\tilde{f}_s(\underset{\sim}{k},\underset{\sim}{v}) \equiv (2\pi)^{-3} \int d\underset{\sim}{x} \, f_{0s}(\underset{\sim}{x},\underset{\sim}{v}) \exp(i\underset{\sim}{k}\cdot\underset{\sim}{x})$

$$\tilde{\phi}(k) \equiv (2\pi)^{-3} \int d\underset{\sim}{x} \, \phi(\underset{\sim}{x},0) \exp(-i\underset{\sim}{k}\cdot\underset{\sim}{x})$$

and $\phi(\underset{\sim}{x},t)$ is assumed to vary in time as $\exp(-i\omega t)$.

Eq. (4) can be written in compact form as

$$\int d\underset{\sim}{k} \exp(i\underset{\sim}{k}\cdot\underset{\sim}{x}) \tilde{\phi}(\underset{\sim}{k}) D(\underset{\sim}{x},\underset{\sim}{k},\omega) = 0$$

or

$$D(\underset{\sim}{x}, i\nabla, \omega)\phi(\underset{\sim}{x}) = 0 \quad (5)$$

where D/k^2 is the usual dielectric function, defined by

$$D(\underset{\sim}{x},\underset{\sim}{k},\omega) \equiv -k^2 + \sum_s D_s(\underset{\sim}{x},\underset{\sim}{k},\omega)$$

$$D_s(\underset{\sim}{x},\underset{\sim}{k},\omega) \equiv \frac{4\pi q_s^2}{m_s} \exp(-i\underset{\sim}{k} \cdot \underset{\sim}{x}) \int d\underset{\sim}{k}' \exp(i\underset{\sim}{k}' \cdot \underset{\sim}{x})$$

$$\int d\underset{\sim}{v} \int_{-\infty}^{0} d\tau \, i\underset{\sim}{k} \cdot \frac{\partial \tilde{f}_s}{\partial \underset{\sim}{v}_s'}(\underset{\sim}{k}-\underset{\sim}{k}', \underset{\sim}{v}_s') \exp[i\underset{\sim}{k}' \cdot (\underset{\sim}{x}_s' - \underset{\sim}{x}) - i\omega\tau] \quad (6)$$

If the magnetic field varies over a distance comparable to a Larmor radius, then it is difficult to calculate the zero-order orbits $x_s'(\tau)$, and to find an equilibrium distribution function $f_{0s}(\underset{\sim}{x},\underset{\sim}{v})$. In a low-$\beta$ plasma, however, it is possible for the scale length of the magnetic field (i.e. $B_0/|\nabla B_0|$) to be much greater than the scale length of the plasma, so we can assume the magnetic field $\underset{\sim}{B}_0(x)$ to be uniform in space, or varying gradually, in calculating the particle orbits and the constants of motion, without assuming that $f_{0s}(\underset{\sim}{x},\underset{\sim}{v})$ varies gradually as a function of $\underset{\sim}{x}$. For uniform $\underset{\sim}{B}_0$, the constants of motion are $v_\perp$, $v_\parallel$, and $\underset{\sim}{x}_{gc\perp} \equiv x_\perp + \underset{\sim}{v} \times \hat{B}_0 \omega_{cs}^{-1}$ (where $\hat{B}_0$ is a unit vector parallel to $\underset{\sim}{B}_0$, and $v_\perp$ and $v_\parallel$ are components of $\underset{\sim}{v}$ perpendicular and parallel to $\underset{\sim}{B}_0$), so

$$f_{0s}(\underset{\sim}{x},\underset{\sim}{v}) = g_s(v_\perp, v_\parallel, \underset{\sim}{x}_{gc\perp})$$

and (see Appendix A)

$$\tilde{f}_s(\underset{\sim}{k},\underset{\sim}{v}) = \exp[-i\underset{\sim}{k}\cdot(\underset{\sim}{v} \times \hat{B}_0)\, \omega_{cs}^{-1}]\tilde{g}_s\,(v_\perp, v_\parallel, \underset{\sim}{k}) \tag{7}$$

where

$$\tilde{g}_s(v_\perp, v_\parallel, \underset{\sim}{k}) \equiv (2\pi)^{-3} \int d\underset{\sim}{x}\; g_s(v_\perp, v_\parallel, \underset{\sim}{x})\, \exp(i\underset{\sim}{k} \cdot \underset{\sim}{x})$$

The particle orbits are

$$\underset{\sim}{x}_s'(\tau) = \underset{\sim}{x} + \frac{\underset{\sim}{v}_\perp}{\omega_{cs}} \sin \omega_{cs}\tau + \frac{\underset{\sim}{v} \times \hat{B}_0}{\omega_{cs}} (1-\cos \omega_{cs}\tau)$$

$$+ v_\parallel \hat{B}_0\, \tau \tag{8}$$

[For non-uniform $\underset{\sim}{B}_0(\underset{\sim}{x})$, the particle drifts would have to be included in Eq.(8) as well, and the constants of motion replaced by adiabatic constants of motion.]

Putting Eqs. (7) and (8) into Eq. (6), and using the Bessel function identity[7]

$$\exp(i\, a \sin\theta) = \sum_\ell J_\ell(a) \exp(i\ell\theta)$$

we can do the integration over τ and over the azimuthal direction of $\underset{\sim}{v}$ to obtain (see Appendix A)

$$D_s(\underset{\sim}{x},\underset{\sim}{k},\omega) = \frac{4\pi q_s^2}{m_s} \exp(-i\underset{\sim}{k}\cdot\underset{\sim}{x}) \int d\underset{\sim}{k}' \exp(i\underset{\sim}{k}'\cdot\underset{\sim}{x}) \int_{-\infty}^{\infty} dv_\parallel$$

$$\int_0^\infty 2\pi v_\perp \, dv_\perp \sum_\ell [\hat{k}_\perp\cdot(\hat{k}_\perp' + i\hat{B}_0 \times \hat{k}_\perp')]^\ell \, J_\ell\left(\frac{k_\perp v_\perp}{\omega_{cs}}\right) J_\ell\left(\frac{k_\perp' v_\perp}{\omega_{cs}}\right)$$

$$(\omega-\ell\omega_{cs}-k_\parallel v_\parallel)^{-1} \left[\frac{-i\underset{\sim}{k}\cdot(\hat{B}_0\times\underset{\sim}{k}')}{\omega_{cs}} \tilde{g}_s(v_\perp,v_\parallel,\underset{\sim}{k}-\underset{\sim}{k}')\right.$$

$$\left. -k_\parallel \frac{\partial}{\partial v_\parallel}\tilde{g}_s(v_\perp,v_\parallel,\underset{\sim}{k}-\underset{\sim}{k}') - \frac{\ell\omega_{cs}}{v_\perp}\frac{\partial}{\partial v_\perp}\tilde{g}_s(v_\perp,v_\parallel,\underset{\sim}{k}-\underset{\sim}{k}')\right] \qquad (9)$$

where $k_\parallel \equiv \underset{\sim}{k}\cdot\hat{B}_0$, $\underset{\sim}{k}_\perp \equiv \underset{\sim}{k} - k_\parallel\hat{B}_0$, $\hat{k}_\perp \equiv \underset{\sim}{k}_\perp/k_\perp$.

If the scale length of the plasma is much greater than a Larmor radius, then only $\underset{\sim}{k}' \approx \underset{\sim}{k}$ will contribute significantly to the integral over $\underset{\sim}{k}'$ in Eq.(9), and $D_s(\underset{\sim}{x},k,\omega)$ can be expressed in terms of $g_s(v_\perp,v_\parallel,\underset{\sim}{x})$, and its gradient at the same point x in space[8]: (see Appendix B)

$$D_s(\underset{\sim}{x},\underset{\sim}{k},\omega) = -\frac{4\pi q_s^2}{m_s}\int dv_\parallel \int 2\pi v_\perp dv_\perp \sum_\ell J_\ell^2\,(k_\perp v_\perp/\omega_{cs})$$

$$(\omega-\ell\omega_{cs}-k_\parallel v_\parallel)^{-1}\,[\underset{\sim}{k}\cdot(\hat{B}_0\times\nabla g_s)/\omega_{cs} + k_\parallel\partial g_s/\partial v_\parallel$$

$$+ (\ell\omega_{cs}/v_\perp)\,\partial g_s/\partial v_\perp] \qquad (10)$$

If a suitable $g_s(v_\perp,v_\parallel,\underset{\sim}{x})$ is chosen, however, it may be possible to integrate Eq. (9) over $\underset{\sim}{k}'$ analytically; in this case there is no need to assume that the Larmor radius is small compared to the scale length.

III. Methods of Solution

We will consider only modes with $k_\parallel = 0$, so the integration over $v_\parallel$ can be done by replacing $g_s(v_\perp, v_\parallel, \underset{\sim}{x})$ by

$$g_{s\perp}(v_\perp, \underset{\sim}{x}) \equiv \int dv_\parallel \; g_s(v_\perp, v_\parallel, \underset{\sim}{x})$$

[with $\tilde{g}_{s\perp}(v_\perp, \underset{\sim}{k})$ similarly defined] in Eq. (9). We will further assume that $g_{s\perp}$ is Maxwellian and that there are no temperature gradients

$$g_{s\perp}(v_\perp, \underset{\sim}{x}) = (2\pi v_s^2)^{-1} \exp(-v_\perp^2/2v_s^2) n_s(\underset{\sim}{x}).$$

A loss-cone distribution can be constructed by subtracting two Maxwellians[9]

$$g_{s\perp}(v_\perp, \underset{\sim}{x}) = [2\pi v_s^2(1-R^{-1})]^{-1} \; n_s(\underset{\sim}{x})$$
$$[\exp(-v_\perp^2/2v_s^2) - \exp(-Rv_\perp^2/2v_s^2)] \qquad (11)$$

where R is the mirror ratio. The two components can be formally treated as two different species, with thermal velocities v_s and $v_s R^{-1/2}$. For the rest of this paper, unless otherwise noted, all species will be assumed to have Maxwellian distributions, with the understanding that loss cone species will be formally treated as two different Maxwellian species.

Finally, we assume a slab geometry, with density $n_s(\underset{\sim}{x})$ depending only on x and $\underset{\sim}{B}_0$ in the z direction. Then Eq. (9) becomes (see Appendix C)

$$D_s(\underset{\sim}{x}, \underset{\sim}{k}, \omega) = \frac{4\pi q_s^2}{m_s} \int_{-\infty}^{\infty} dk'' \exp(ik''x) \sum_\ell \exp(i\ell\alpha)$$
$$\exp[-(k^2+k'^2)a_s^2/2] I_\ell (k\,k'a_s^2) \; (\omega - \ell\omega_{cs})^{-1}$$
$$(-ik_y k''/\omega_{cs} + \ell\,\omega_{cs}/v_s^2) \; \tilde{n}_s(k'') \qquad (12)$$

where $k'' \equiv k_x' - k_x$, $k_y' \equiv k_y$, $\alpha \equiv i\log_e\left(\frac{k_x' + ik_y}{k'}\right) - i\log_e\left(\frac{k_x + ik_y}{k}\right)$ is the angle between the vectors $\underset{\sim}{k}$ and $\underset{\sim}{k}'$,

$$a_s \equiv v_s/\omega_{cs},$$

and $$\tilde{n}_s(k'') \equiv (2\pi)^{-1} \int_{-\infty}^{\infty} dx \exp(-ik''x) n_s(x).$$

For a sinusoidal density profile,

$$n_s(x) = (1 + \Delta_s \cos k_0 x)\, n_{s0} \tag{13}$$

we have $\tilde{n}_s(k'') = (2\pi)^{-1} n_{s0}[\delta(k'') + (\Delta_s/2)\delta(k'' + k_0) + (\Delta_s/2)\,\delta(k'' - k_0)]$.

(There is no requirement that Δ_s be small, in fact usually we will use $\Delta_s \simeq 1$; see Fig. 1.)

Then the integral over k'' in Eq. (12) becomes a sum over three terms, $k'' = 0, -k_0, +k_0$. Since we are assuming no equilibrium electric fields, we must have

$$\sum_s q_s \int \underset{\sim}{dv}\, f_{0s}(\underset{\sim}{x},\underset{\sim}{v}) = 0 \text{ for all } \underset{\sim}{x}$$

or (see Appendix D)

$$\sum_s q_s\, n_{s0}\Delta_s \exp(-k_0^2 a_s^2/2) = 0 \tag{14a}$$

and

$$\sum_s q_s n_{s0} = 0 \tag{14b}$$

The eigenmodes $\phi(\underset{\sim}{x})$ satisfying Eq. (5) must be of the form

$$\phi(\underset{\sim}{x}) = \phi(x) \exp(ik_y y)$$

since $D(\underset{\sim}{x},\underset{\sim}{k},\omega)$ has no dependence on y, and we have been assuming $k_\parallel = 0$. We can then write Eq. (5) as

$$\int dk_x \exp(ik_x x)\, D(x,k_x,\omega)\, \tilde{\phi}(k_x) = 0$$

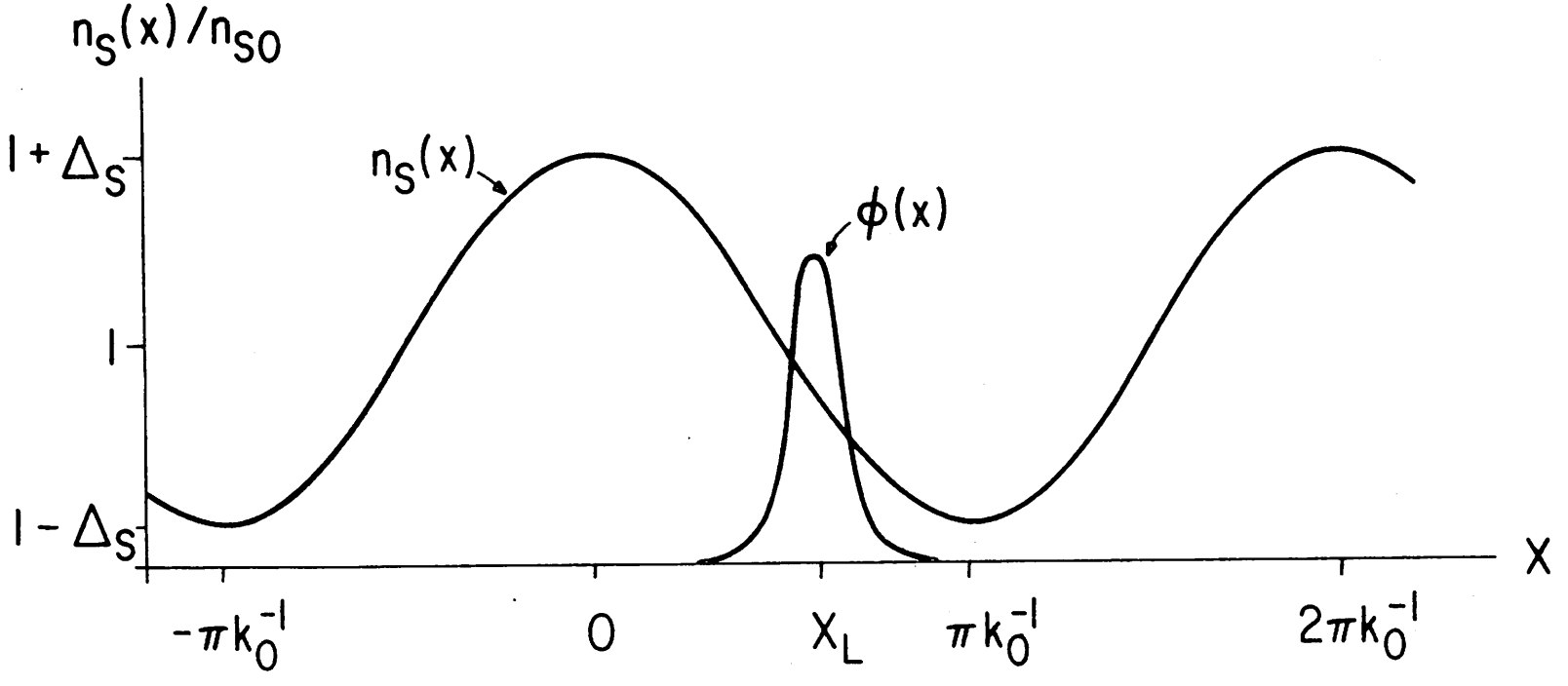


Figure 1. Typical guiding center profile $n_s(x)$ given by Eq.(13), with typical normal mode potential $\phi(x)$

or

$$D(x,-i\frac{\partial}{\partial x},\omega)\phi(x) = 0 \tag{15}$$

where k_y is understood to be a parameter, and

$$\tilde{\phi}(k_x) \equiv (2\pi)^{-1} \int_{-\infty}^{\infty} dx\ \phi(x)\ \exp(-ik_x x)$$

We want to find the normal modes $\phi_n(x)$ and the corresponding frequencies ω_n which satisfy Eq. (15) with appropriate boundary conditions.

A. Local Method

One approach is to expand $D(x,k_x)$ in a power series around some $x = x_L$, $k_x = k_L$, and $\omega = \omega_L$ (all complex, in general) such that

$$D(x_L,k_L,\omega_L) = \frac{\partial D}{\partial x}(x_L,k_L,\omega_L) = \frac{\partial D}{\partial k_x}(x_L,k_L,\omega_L) = 0 \tag{16}$$

(Note that a wave packet localized around x_L, with well-defined wave number k_L, would have a frequency close to ω_L, and would have zero group velocity and acceleration.)

Then

$$D(x,k_x,\omega) = \frac{1}{2}(x-x_L)^2\frac{\partial^2 D}{\partial x^2} + \frac{1}{2}(k_x-k_L)^2\frac{\partial^2 D}{\partial k_x^2}$$
$$+ (x-x_L)(k_x-k_L)\frac{\partial^2 D}{\partial x \partial k_x} + (\omega-\omega_L)\frac{\partial D}{\partial \omega} + \ldots \tag{17}$$

where the derivatives of D are evaluated at x_L,k_L,ω_L, and Eq. (15) becomes

$$A\frac{d^2\psi}{dx^2} + Bx\frac{d\psi}{dx} + C\,x^2\psi + \lambda\psi = 0 \tag{18}$$

where

$$A \equiv -1/2\frac{\partial^2 D}{\partial k_x^2},\quad B \equiv -i\frac{\partial^2 D}{\partial k_x \partial x},\quad C \equiv 1/2\frac{\partial^2 D}{\partial x^2},$$

and

$$\lambda \equiv (\omega - \omega_L)\frac{\partial D}{\partial \omega}\ , \text{ all evaluated at } \quad x_L, k_L, \text{ and } \omega_L;$$

and $\psi(x) \equiv \exp(-ik_L x)\phi(x + x_L)$.

This approach is similar to that used by other authors[10-12], who have, however, taken $k_L = 0$ to satisfy Eq. (16), and hence $B = 0$ in Eq. (18). Furthermore, the truncation of Eq. (17) has usually been justified by assuming small Larmor radius compared to scale length, and we wish to emphasize that this assumption is not the only one which will justify Eq. (17).

Taking $\psi(x) \to 0$ as $x \to \pm\infty$ as the boundary conditions, Eq. (18) has the solutions (see Appendix E)

$$\psi_n(x) = H_n(\eta^{1/2}x)\exp(-\beta x^2/2) \tag{19a}$$

$$\lambda_n = (2n\eta + \beta)A \tag{19b}$$

where

$$\eta \equiv [(B/2A)^2 - C/A]^{1/2},\ \beta \equiv \eta + B/2A,$$

and H_n is the order n Hermite polynomial, n=0,1,2... Then

$$\phi_n(x) = H_n[\eta^{1/2}(x-x_L)]\ \exp(-\beta(x-x_L)^2/2 + ik_L x) \tag{20a}$$

$$\omega_n = \omega_L - (n\eta + \beta/2)\ (\frac{\partial D}{\partial \omega})^{-1}\ \frac{\partial^2 D}{\partial k_x^2} \tag{20b}$$

A typical normal mode $\phi_n(x)$ for n=0 is shown in Fig. 1.

These solutions are valid provided we can neglect the higher derivatives of D in Eq. (17), and provided the Stokes lines of D have the proper

topological behavior in the complex x-plane with respect to the real axis to allow the boundary conditions to be satisfied.[13] We conjecture that Eq.(20) is a valid solution for the first few modes (n=0,1,2...) provided (see Appendix F)

$$L(k_L^2 + k_y^2)^{1/2} \gg 1 \tag{21}$$

(where L is the scale length of the plasma) except perhaps for special unusual cases. For instabilities requiring $k_\perp a_i \gtrsim 1$ (e.g. drift cone[14] or drift cyclotron[15]), $a_i \ll L$ is sufficient for inequality (21) to be satisfied, but it is certainly not necessary.

Since ω_n depends linearly on n, the fastest growing mode must be the n = 0 mode for some set (x_L, k_L, ω_L) satisfying Eq. (16), or else it cannot be found by the local method at all. [It has turned out in almost every case we have examined for which inequality (21) is satisfied, that the fastest growing mode can in fact be found by the local method using Eqs.(16) and (20), and is thus the n = 0 mode, but it is possible to construct models where this will not be true.] The fastest growing mode will then be

$$\phi(x) = \exp[-\beta(x-x_L)^2/2 + ik_L x] \tag{22a}$$

$$\omega = \omega_L - \frac{\beta}{2}\left(\frac{\partial D}{\partial \omega}\right)^{-1} \frac{\partial^2 D}{\partial k_x^2} \approx \omega_L \tag{22b}$$

where we require that x_L, k_L, ω_L satisfy

$$\mathrm{Im}\left[\delta\left(\frac{\partial D}{\partial \omega}\right)^{-1} \frac{\partial^2 D}{\partial k_x^2}\right] > 0$$

as well as Eq. (16).

A large class of solutions (x_L, k_L, ω_L) satisfying Eq. (16) have $k_L = 0$, since this guarantees that $\partial D/\partial k_x = 0$, and it is then only necessary to find x_L and ω_L satisfying $D = 0$ and $\partial D/\partial x = 0$. In this case $\partial^2 D/\partial k_x \partial x = 0$, and

$$\delta = \beta = \left(\frac{\partial^2 D}{\partial x^2} \Big/ \frac{\partial^2 D}{\partial k_x^2}\right)^{1/2}$$

B. Nonlocal Method

When inequality (21) is not satisfied a different approach is required. We Fourier transform $D(x,k_x,\omega)$

$$D(x,k_x,\omega) = (2\pi)^{-1} \int_{-\infty}^{\infty} dk'' \exp(ik''x)\, \tilde{D}(k'',k_x,\omega)$$

where

$$\tilde{D}(k'',k_x,\omega) \equiv \int_{-\infty}^{\infty} dx \exp(-ik''x)\, D(x,k_x,\omega) \tag{23}$$

Putting Eq. (23) into Eq. (15), multiplying by $\exp(-ik'x)$ for arbitrary k', and integrating over x, we obtain

$$\int dk''\, \tilde{D}(k'',k'-k'',\omega)\, \tilde{\phi}(k'-k'') = 0 \tag{24}$$

for all k'.

For the sinusoidal density profile given by Eq. (13), using Eqs. (6) and (12), Eq. (23) becomes

$$\tilde{D}(k'',k_x,\omega) = G_0(k_x,\omega)\, \delta(k'') + G_{+1}(k_x,\omega)\, \delta(k''-k_0) + G_{-1}(k_x,\omega)\, \delta(k'' + k_0) \tag{25}$$

where

$$G_0 \equiv - k^2 + \sum_s G_{0,s},$$

$$G_{\pm 1} \equiv \sum_s G_{\pm 1,s},$$

$$G_{0,s}(k_x,\omega) \equiv \frac{\omega_{ps}^2}{v_s^2} \exp(-k^2 a_s^2) \sum_\ell I_\ell(k^2 a_s^2)(\omega/\omega_{cs} - \ell)^{-1} \ell$$

$$G_{\pm 1,s}(k_x,\omega) \equiv \frac{\Delta_s \omega_{ps}^2}{2v_s^2} \exp[-(k^2+k_{\pm 1}^2)a_s^2/2]$$

$$\sum_\ell \exp(i\ell\alpha_{\pm 1})\, I_\ell(k\, k_{\pm 1} a_s^2)(\omega/\omega_{cs} - \ell)^{-1} (\ell \pm i k_y k_0 a_s^2)$$

$$k_{\pm 1}^2 \equiv (k_x \pm k_0)^2 + k_y^2,$$

$a_{\pm 1} \equiv i \log_e \left(\frac{k_x \pm k_0 + ik_y}{k_{\pm 1}}\right) - i \log_e \left(\frac{k_x + ik_y}{k}\right)$ is the angle between $\underset{\sim}{k}$ and $\underset{\sim}{k}_{\pm 1}$, and

$$\omega_{ps}^2 \equiv 4\pi n_{s0} q_s^2/m_s.$$

Then the integral in Eq. (24) becomes a discrete sum

$$G_0(k',\omega)\, \tilde{\phi}(k') + G_{+1}(k'-k_0,\omega)\, \tilde{\phi}(k'-k_0) + G_{-1}(k' + k_0,\omega)$$

$$\tilde{\phi}(k' + k_0) = 0 \qquad (26)$$

If we require $\phi(x)$ to be periodic with the same periodicity k_0 as the density [appropriate for a simulation with periodic boundary conditions; non-periodic $\phi(x)$ will be considered later], then $\tilde{\phi}(k_x)$ will vanish except at $k_x = pk_0$ for integer p:

$$\tilde{\phi}(k_x) = \sum_{p=-\infty}^{\infty} \phi_p\, \delta(k_x - pk_0)/2\pi$$

where

$$\phi_p \equiv (k_0/2\pi) \int_0^{2\pi/k_0} dx\ \phi(x)\ \exp(-ipk_0x)$$

and

$$\phi(x) = \sum_p \phi_p \exp(ipk_0x) \tag{27}$$

[It should be kept in mind that in general $\phi(x)$ is complex, so there is no relation between ϕ_p and ϕ_{-p}, as there would be if $\phi(x)$ were always real.]

Then, setting $k' = pk_0$, Eq. (26) becomes

$$A_{p,p-1}(\omega)\phi_{p-1} + A_{p,p}(\omega)\phi_p + A_{p,p+1}(\omega)\phi_{p+1} = 0 \tag{28}$$

where

$$A_{p,p'}(\omega) \equiv G_{(p-p')}(p'k_0,\omega).$$

Eq. (28) must be satisfied for all integers p, since Eqs. (24) and (26) are satisfied for arbitrary k'. So,we require

$$\det A_{p,p'}(\omega) = 0 \tag{29}$$

If, for the modes we are interested in, ϕ_p is negligibly small for all $p \geq p_{max}$, and all $p \leq p_{min}$ for some finite p_{max} and p_{min},then we can truncate the matrix $A_{p,p'}$ at $p > p_{max}$ and $p < p_{min}$. We can then solve Eq. (29) to find the normal mode frequencies ω (in general there will be $p_{max}-p_{min} + 1$ of them for each cyclotron harmonic), use Eq. (28) to find the ϕ_p for each normal mode, and then use Eq. (27) to find $\phi(x)$.

This method is easily generalized to periodic density profiles $n_s(x)$ other than the sinusoidal profile given by Eq. (13). For

$$n_s(x) = \sum_{j=-\infty}^{\infty} n_{s,j} \exp(ijk_0x) \tag{30}$$

Eq. (25) becomes

$$\tilde{D}(k'',k_x,\omega) = \sum_{j=-\infty}^{\infty} G_j(k_x,\omega)\ \delta(k'' - jk_0) \tag{31}$$

with G_j defined like $G_{\pm 1}$ in Eq. (25) but with $\Delta_s/2$ replaced by $n_{s,j}/n_{s0}$, and $\pm k_0$ replaced by $+ jk_0$. Eq. (28) then becomes

$$\sum_{p'} A_{p,p'}\ \phi_{p'} = 0$$

which again leads to Eq. (29). The advantage in using a sinusoidal density profile is that the matrix $A_{p,p'}$ is tri-diagonal, and Eq. (29) is easier to solve numerically than it would be for an arbitrary matrix. Adding terms with $2k_0$, for example, would result in a penta-diagonal matrix. On the other hand, an isolated slab of plasma with $a_i/L \sim 1$ would be more realistically modelled by a periodic density profile including higher harmonics of k_0.

It is also not difficult to generalize this method to potentials $\phi(x)$ which are not periodic in x. It can be shown using Floquet's theorem that, except for a measure zero subset of the parameter space, all normal mode potentials must be of the form (see Appendix G)

$$\phi(x) = \exp(iKx) \sum_{p} \phi_p \exp(ipk_0 x) \tag{32}$$

for some K, $0 < K < k_0$. Then we can still use Eqs. (28) and (29), but we must redefine $A_{p,p'}$ as

$$A_{p,p'}(\omega) \equiv G_{(p-p')}(p'k_0 + K,\omega) \tag{33}$$

The nonlocal method will only be practical if we can use $p_{max} - p_{min}$ not too large; otherwise the numerical solution of Eq. (29) will be too difficult. In order to estimate the conditions under which the nonlocal method is practical, we will temporarily assume that inequality

(21) is at least marginally satisfied, so that Eq. (20a) is at least qualitatively correct for the fastest growing mode. Then, using the definition of ϕ_p in Eq. (27), we find

$$\phi_p \propto \exp[-(pk_0-k_L)^2/2\beta - ipk_0x_L]$$

so we must choose $p_{max}-p_{min}$ to be at least a few times $|\beta|^{1/2}k_0^{-1}$. We estimate that β is on the order of $k_0(k_L^2 + k_y^2)^{1/2}$ from the definitions following Eqs. (18) and (19), since the scale length of D in x will be about k_0^{-1}, while the "scale length" of D in k_x will be about $(k_L^2 + k_y^2)^{1/2}$, and we can estimate the relative magnitudes of the various second derivatives of D from these scale lengths. It then follows that

$$p_{max} - p_{min} \gtrsim k_0^{-1}(k_L^2 + k_y^2)^{1/2} \tag{34}$$

and we need $p_{max} - p_{min} >> 1$ if inequality (21) is satisfied. So the nonlocal method is not a practical way to find the fastest growing mode in those cases where the local method is valid, whereas it is practical when the local method is marginally valid. We cannot say in general whether the nonlocal method is practical when inequality (21) is not even marginally satisfied. However, in all such cases which we have investigated numerically, $p_{max} - p_{min}$ can be chosen small (of order unity). Thus the local and nonlocal methods complement each other; each method works best in the regime where the other method is least efficient or least valid.

IV. Numerical Techniques

To find the normal modes and frequencies using the nonlocal method, the roots of Eq. (29) were found numerically in the region of interest of the complex ω plane, using a standard root-finding routine[16,17]

(modified by the authors) based on Muller's method.[18] Usually a small value of $k_y a_i$ (typically 0.5) was used initially, since the root-finder was most efficient near the real axis (rarely missing roots with Im $\omega/\omega_{ci} \lesssim 1$) and at small $k_y a_i$ the roots tended to be near the real axis. Then each root found at the initial $k_y a_i$ was followed out to larger $k_y a_i$; in this way it was possible to find roots far from the real axis at large $k_y a_i$, which the root-finder would very likely miss if $k_y a_i$ were large initially. For each root found, the eigenvector $(\phi_{pmin}, \ldots \phi_{pmax})$ was calculated from Eq. (28), and $\phi(x)$ [from Eq. (27)] was plotted out if desired. In practice p_{min} was always set equal to $-p_{max}$, since the matrix elements $A_{p,p'}$ for $p,p' < 0$ could be found using the identity $A_{p,p'} = A_{-p',-p}$ [following from the definitions of $A_{p,p'}$ and G_0, $G_{\pm 1}$ after Eqs. (28) and (25)]; thus little time would be saved by using $|p_{min}| \neq |p_{max}|$ if $p_{min} < 0 < p_{max}$. (Some time would be saved by using $|p_{min}| < |p_{max}|$ to find modes with $\phi_{-p_{max}} \ll \phi_{p_{max}}$.)

Since the time required to calculate det $A_{p,p'}$ is proportional to p_{max}, and the number of roots is proportional to p_{max}, the time required to find all the roots in a given range of k_y and ω was roughly proportional to p_{max}^2. A typical run for a loss cone deuterium plasma with $\omega_{pe} \approx \omega_{ce}$ and $k_0 a_i \approx 1$, in the range of k_y and ω typical of the drift cone instability ($0.5 \leq k_y a_i \leq 20$, $|\omega| / \omega_{ci} < 2.5$ at $k_y a_i = 0.5$), using $p_{max} = 3$, took one minute on the CDC 7600 computer. However, for $p_{max} \gtrsim 5$ the roots were so close together, especially near the cyclotron harmonics, that the root finder could not find many of them. In order to avoid this problem and to reduce the computation time, an approximation was sometimes employed in which the asymptotic form for the Bessel function[19]

$$e^{-x} I_\ell(x) \approx (2\pi x)^{-1/2} \exp(-\ell^2/2x), \quad (x >> \ell >> 1)$$

was used, and the sum over ℓ replaced by an integral, for the ion terms. Then the definitions after Eq. (25) become, for the ion terms (see Appendix H

$$G_{0,s}(k_x,\omega) = -(\omega_{ps}^2/v_s^2)\,[1 + \zeta_0 Z(\zeta_0)]$$

$$G_{\pm1,s}(k_x,\omega) = -(\omega_{ps}^2 \Delta_s/2v_s^2) \exp(-k_0^2 a_s^2/2)[1 + \zeta_{\pm1} Z(\zeta_{\pm1})] \tag{35}$$

where

$$\zeta_0 \equiv \omega/(\sqrt{2}kv_s)$$

$$\zeta_{\pm1} \equiv (\omega \pm ik_y k_0 a_s v_s)/(\sqrt{2}\, kv_s)$$

and $Z(\zeta)$ is the usual plasma dispersion function[20]

$$Z(\zeta) \equiv \pi^{-1/2}\int dt\, \exp(-t^2)/(t-\zeta)$$

This approximation is valid when $\mathrm{Im}\,\omega \gtrsim \omega_{ci}$, $ka_i \gtrsim 1$, and $k_0 \lesssim k$, and is equivalent to using straight line orbits for the ions when integrating the Vlasov equation over the zero-order orbits. When this approximation is used, the modes associated with the ion cyclotron harmonics (Bernstein waves) are eliminated, allowing higher p_{max} to be used without confusing the root-finder (no problems were encountered up to $p_{max} = 9$, as long as the $Z(\zeta)$ routine had high enough precision) and greatly reducing the computation time for a given p_{max} (typically by a factor of 10).

The electron terms were evaluated in the limit of zero Larmor radius, $ka_e << 1$:

$$G_{0,e}(k_x,\omega) = -\omega_{pe}^2 k^2/(\omega_{ce}^2 - \omega^2)$$

$$G_{\pm 1,e}(k_x,\omega) = -(\omega_{pe}^2 \Delta_e/2)\ [k\ k_{\pm 1}/(\omega_{ce}^2 - \omega^2) \pm i\ k_0 k_y/|\omega_{ce}|\omega] \tag{36}$$

and Δ_e was related to Δ_i (usually set equal to 0.99) by Eq. (14a), so

$$\Delta_e = \Delta_i \sum_{s = \text{ions}} (n_s/n_e)\ \exp(-k_0^2 a_s^2/2) \tag{37}$$

where the sum is over ion species [recall that a loss cone ion distribution, given by Eq. (11), is treated as two Maxwellian ion species].

Numerical solutions were obtained using the local method also. The first two parts of Eq. (16), D = 0, and $\partial D/\partial x = 0$, were solved simultaneously for complex ω_L and x_L, using a Newton-Raphson[21] root finder (k_L was assumed to be zero, so that $\partial D/\partial k_x = 0$ was always satisfied). This root-finder (in contrast to the root-finder used for the nonlocal method) worked well only when the initial values of ω_L and x_L were reasonably close to the correct values. Initial values were taken either from the results of the nonlocal method, or from previous calculations of ω_L and x_L using slightly different parameters. Because the root-finder was very quick, and the equations to be solved were relatively simple, it was possible to cover the parameter space fairly densely, using the local method. In fact the entire parameter space could be covered in less time using the local method than it took to solve a single case using the nonlocal method, and the results were more accurate using the local method when inequality (21) was well-satisfied. On the other hand, the nonlocal method was needed to obtain initial values of ω_L and x_L in a given regime of parameter space, as well as to find modes with $k_L \neq 0$, and modes with inequality (21) not satisfied.

Abridged flow charts and complete listings for the program ROOTS and its subroutines (used for the nonlocal method) and of the program LOCAL and its subroutines (used for the local method) are given in Appendix I.

V. Results

Runs were made for both methods using a mirror ratio R = 3, a mass ratio m_i/m_e = 3700 (appropriate for a deuterium plasma), a density gradient k_0a_i ranging from 0.1 to 6.3 [a_i refers to v_i/ω_{ci} where v_i is the thermal velocity of the first Maxwellian component in Eq. (11); the total ion thermal velocity is thus $v_i(1+R^{-1})^{1/2}$], and density $\omega_{pi}^2/\omega_{ci}^2$ ranging from 3 to 10^4. In each case, for the mode with the highest growth rate, ω and k_y were found and $\phi(x,y)$ was plotted. Sample results using the nonlocal method for one case, $\omega_{pi}^2/\omega_{ci}^2 = 30$, $k_0a_i = 1.13$, and $p_{max} = 3$, are shown in Fig. 2, a plot of ω vs. k_y, (note that there are $p_{max}-p_{min}+1 = 7$ brances for each cyclotron harmonic in Fig. 2, as discussed in Sec. IIB.)and Fig. 3, a contour plot of $\phi(x,y)$. When inequality (21) was satisfied and when p_{max} was high enough so that $\phi_p \approx 0$ for $|p| \gtrsim p_{max}$, the results using each of the two methods were found to be in excellent agreement, within 1% for ω, k_y, and x_L, and within 10% for the half-width of $\phi(x)$. (see Appendix J)

Figure 4 shows the growth rate of the fastest growing mode as a function of $\omega_{pi}^2/\omega_{ci}^2$ and k_0a_i, using either the local or the nonlocal results, whichever was most accurate for each set of parameters. Figure 5 shows the minimum p_{max} needed in order to use the nonlocal method, as a function of $\omega_{pi}^2/\omega_{ci}^2$ and k_0a_i. Figure 6 shows $|\phi(x)|^2$ and $|\phi_p|^2$ for various values of k_0a_i, for $\omega_{pi}^2/\omega_{ci}^2 = 1000$.

VI. Intepretation

The parameter space is divided into several regimes in each of which the fastest growing mode has qualitatively different characteristics (shown in Fig. 7) and is driven unstable by a different mechanism.

At $k_0a_i < 1.25$ (region A in Fig. 7), the most unstable mode is a drift cone mode. Its characteristics are in qualitative agreement with

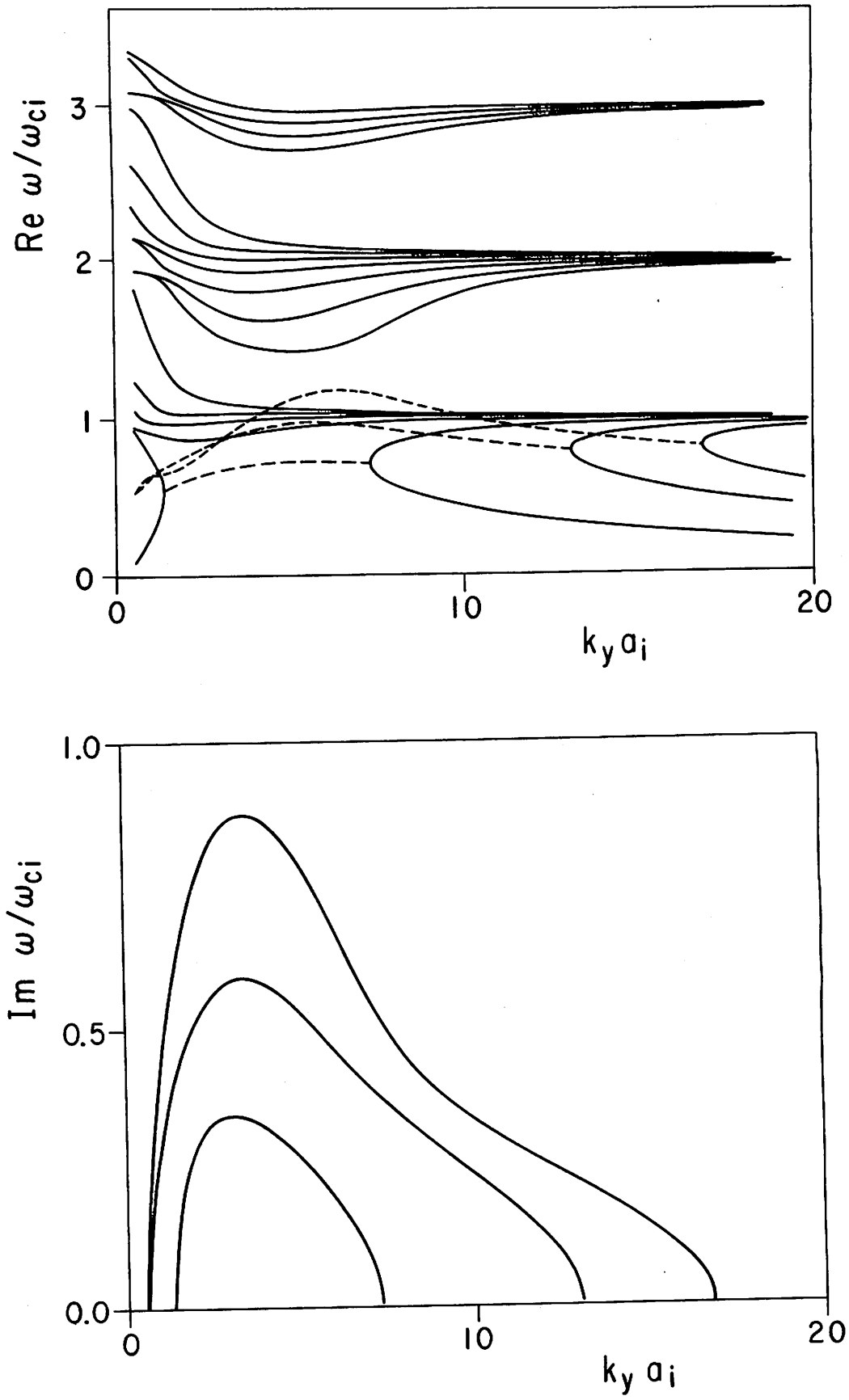


Figure 2. ω vs. k_y plot, using $\omega_{pi}^2/\omega_{ci}^2 = 30$, $m_i/m_e = 3700$, $R = 3$, $k_o a_i = 1.13$, $\Delta_i = 0.99$, $p_{max} = 3$. The real part of ω is shown in the top plot (dashed curves indicate the real parts of complex ω) and the imaginary part of ω (the growth rate) in the bottom plot.

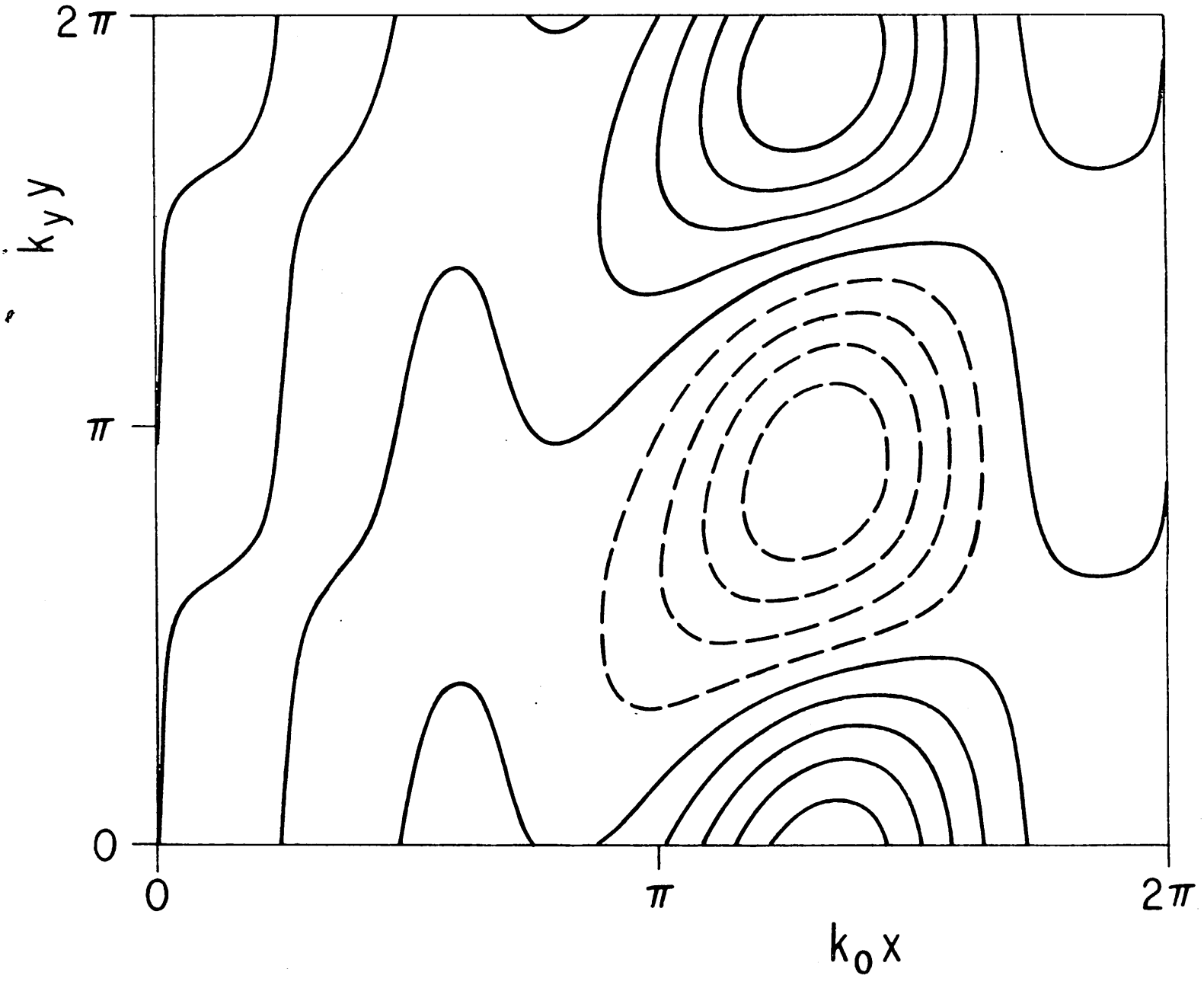


Figure 3. Contour plot of the real part of $\phi(x,y)$ for the fastest growing mode in Fig. 2, at $\omega/\omega_{ci} = 0.953 + 0.873i$, $k_y a_i = 3.55$. The plasma density is at a maximum at $k_o x = 0$ and 2π, and at a minimum at $k_o x = \pi$. Re $\phi(x,y)$ is normalized so that it ranges from -1 to +1. Contours are shown for 0.8, 0.6, 0.4, 0.2, and 0 (solid curves) and for -0.2, -0.4, -0.6, and -0.8. The figure would have to be compressed vertically by a factor of 2.56 to make the vertical and horizontal scales the same.

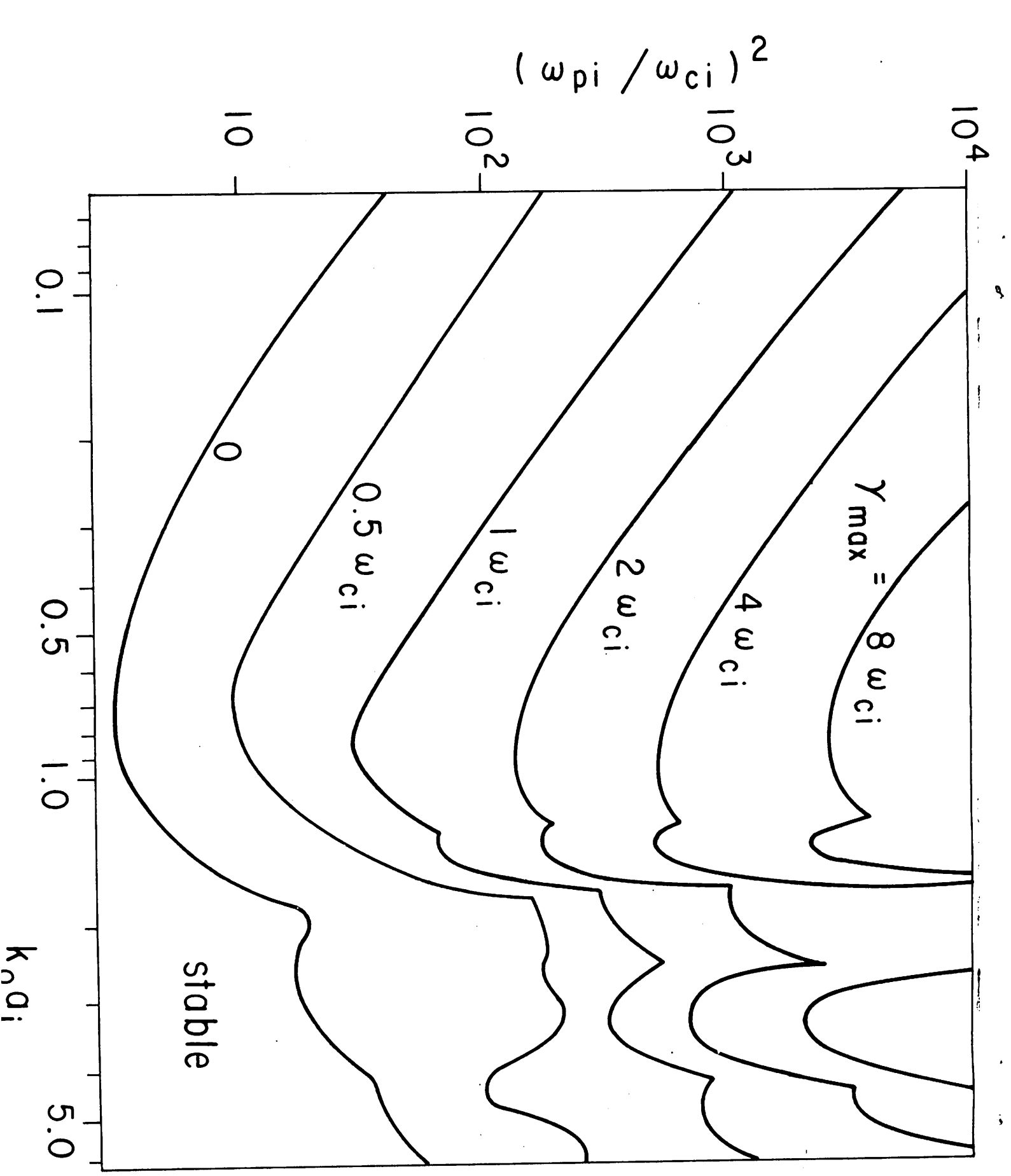


Figure 4. The growth rate γ_{max} of the fastest growing mode is shown as a function of $(\omega_{pi}/\omega_{ci})^2$ and $k_o a_i$, for $\Delta_i = 0.99$, $R = 3$, $m_i/m_e = 3700$.

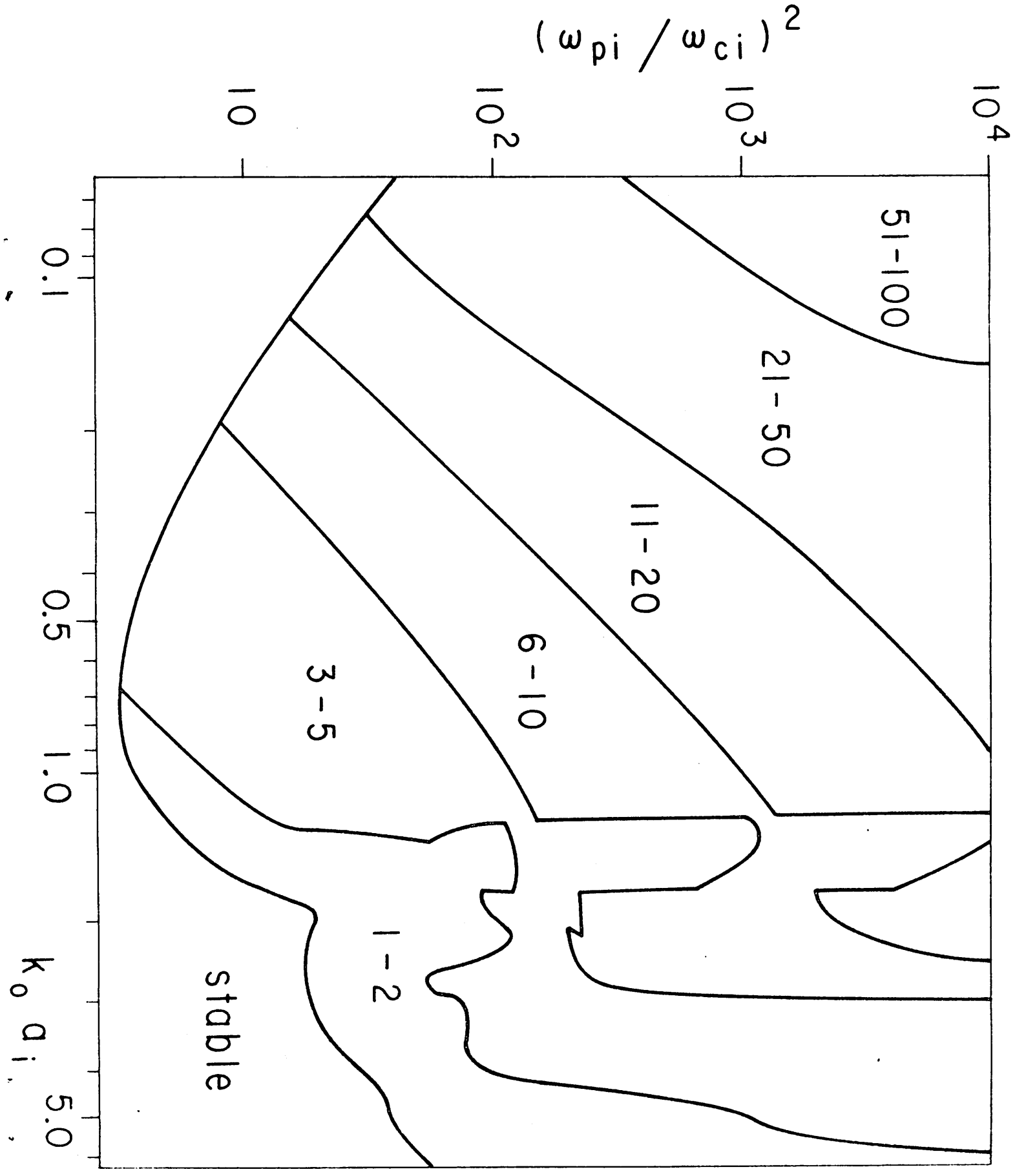


Figure 5. The minimum p_{max} needed to find the fastest growing mode using the nonlocal method is shown as a function of $(\omega_{pi}/\omega_{ci})^2$ and $k_o a_i$, for $\Delta_i = 0.99$, $R = 3$, $m_i/m_e = 3700$.

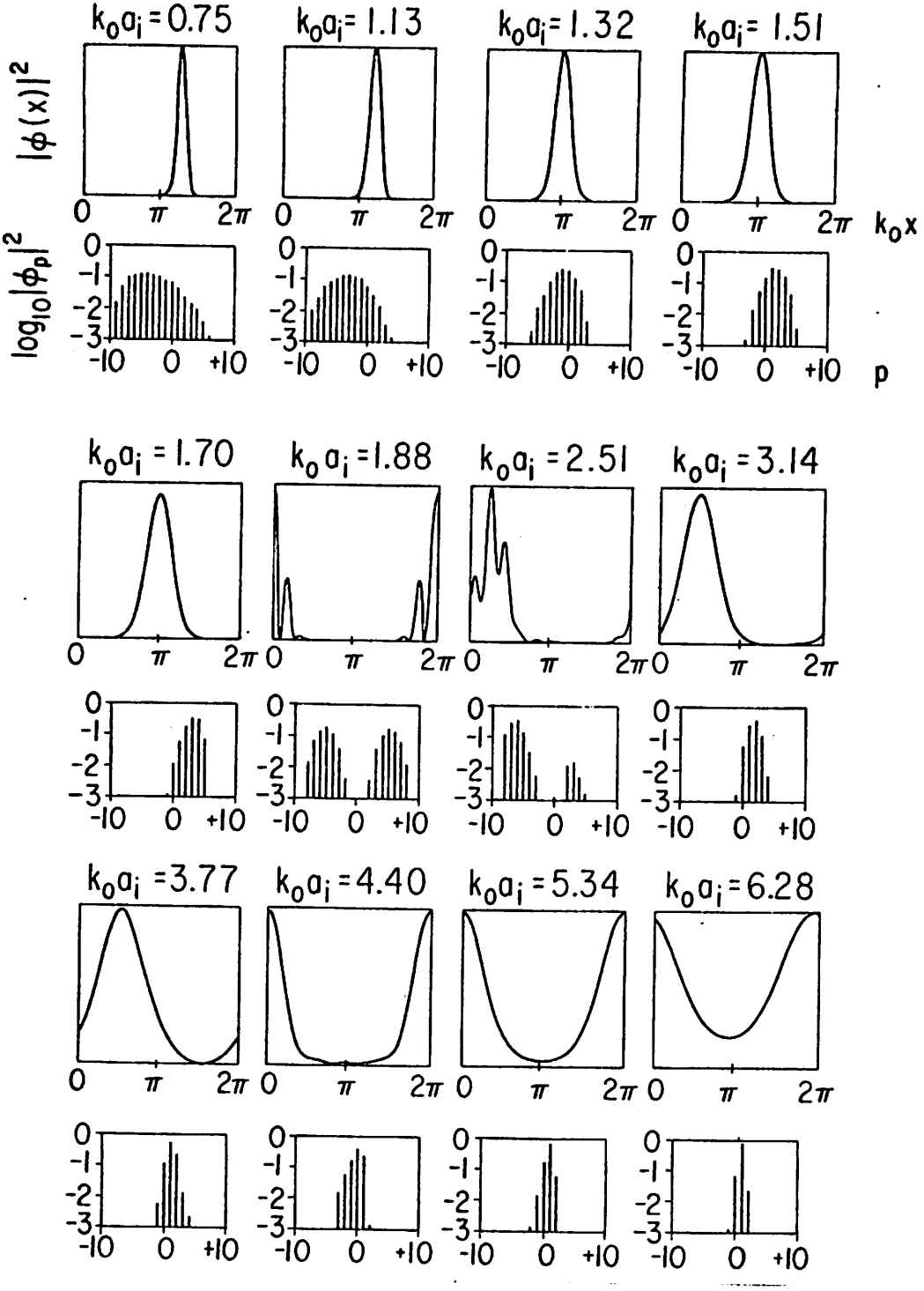


Figure 6. $|\phi(x)|^2$ and $|\phi_p|^2$ are shown for the fastest growing mode for various values of $k_o a_i$, using $(\omega_{pi}/\omega_{ci})^2 = 1000$, $\Delta_i = 0.99$, $R = 3$, and $m_i/m_e = 3700$. The plasma guiding center density is at a maximum at $k_o x = 0$ and 2π, and at a minimum at $k_o x = \pi$. For $k_o a_i > 1.80$, the plasma density (as opposed to guiding center density) has its maximum at $k_o x = \pi$, and its minimum at $k_o x = 0$ and 2π. The twelve cases shown are located respectively in regions A, A, B, B, B, C, D, E, E, F, F, and G of Fig. 7.

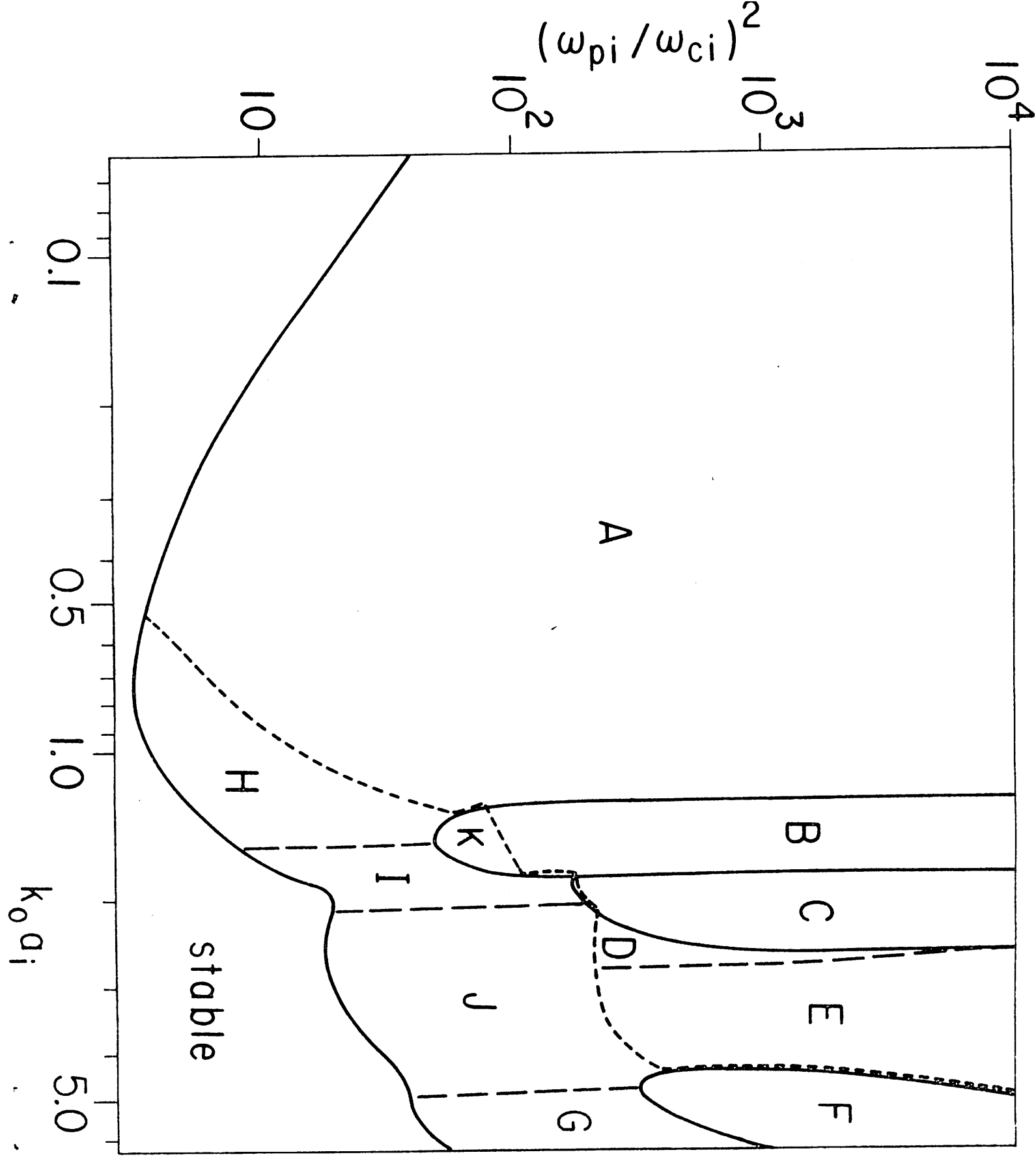


Figure 7. Regions of the parameter space are shown in which the fastest growing mode has different characteristics. The usual drift cone mode occurs in region A; other types of instabilities occur in the other regions, discussed in Sec. VI. R = 3, Δ_i = 0.99, m_i/m_e = 3700.

the mode described by Post and Rosenbluth[14] for an infinite medium with constant density gradient; it is modified by the fact that $\phi(x)$ is centered near the edge of the plasma (typically $k_0 x_L/2\pi \approx 0.6$) where the density gradient $(a_i/n)dn/dx$ (a small parameter in Ref. 14) is relatively large. For $k_0 a_i \gtrsim 0.4$, the ion density gradient begins to compete with the loss cone as a source of free energy for the instability, and the mode has some of the characteristics of the Mikhailovskii drift cyclotron instability[15].

At $k_0 a_i > 1.25$ and $\omega_{pi}^2/\omega_{ci}^2 \gtrsim 50$ (region B in Fig. 7), a new mode appears, and quickly becomes more unstable than the drift cone mode. This mode has $\phi(x)$ centered at $k_0 x/2\pi = 0.5$ (the point of minimum ion guiding center density), and has a purely imaginary frequency. It is an ion two-stream instability, driven by the double-humped ion velocity distribution $f_{0i}(\underset{\sim}{x},\underset{\sim}{v})$ at $k_{0x}/2\pi = 0.5$ [since $n_i(\underset{\sim}{x})$ is double-humped, and $f_{0i}(\underset{\sim}{x},\underset{\sim}{v}) = g_i(v_\perp, v_\parallel) n_i(\underset{\sim}{x}_{gc})$]. It is thus an artifact of the periodic guiding center density profile $n_i(x)$, and would not appear in a plasma with a single-humped density profile.

The ion-two-stream instability is most unstable at $k_0 a_i \approx 1.35$, and disappears at $k_0 a_i > 1.80$. The drift cone (or drift cyclotron) instability also disappears at about this value of $k_0 a_i$, since $\Delta_e = 0$ at $k_0 a_i = 1.80$ for $R = 3$ [in order to satisfy Eq.(14b)], and the drift cone mode depends on an electron density gradient. But for $\omega_{pi}^2/\omega_{ci}^2 \gtrsim 200$, another instability appears at $k_0 a_i \simeq 1.70$, and remains the most unstable mode until $k_0 a_i \approx 2.5$ (region C of Fig. 7). This mode has $\phi(x)$ centered at $k_0 x/2\pi = 0$, the point of maximum ion guiding center density, and is a purely growing ion instability, with $k_\perp \gtrsim k_y$; the maximum growth rate occurs when $k_y = 0$. The instability arises because the projection of the ion

velocity distribution in the x-direction $\int f_{0i}(\underset{\sim}{x},\underset{\sim}{v})dv_y dv_z$, has its loss cone deepened slightly when $d^2n_i/dx^2 < 0$ (true at x = 0), and this makes the ion distribution unstable by the Penrose criterion[22] to modes with wave vector in the x-direction. Eq. (16) can be solved analytically in the limits $k_y = 0$, Re $\omega_L = 0$, $k_L v_i \gtrsim |\omega_L| \gtrsim \omega_{ci}$, which are applicable in region C when $\omega_{pi} >> \omega_{ci}$. In these limits, $D(x,k_x,\omega)$ can be written in terms of the plasma dispersion function $Z(\zeta)$, using Eqs. (11), (23), (25), and (35), and $Z(\zeta)$ can be approximated by $i\pi^{1/2}$, since $\zeta = \omega_L/k_L v_i << 1$.[20] Then Eq. (16) has the solutions (see Appendix K)

$$\omega_L = 0.307\ i\ \omega_{\ell h}(1-R^{-1})^{-1/2}\ (e_2-e_1)^{3/2}\ [R^{1/2}(e_2+1)-(e_1+1)]^{-1}$$

$$k_L = 0.577(\omega_{\ell h}/v_i)\ (1-R^{-1})^{-1/2}\ (e_2-e_1)^{1/2}$$

$$x_L = 0 \qquad (38)$$

where

$$e_1 \equiv \exp(-k_0^2 a_i^2/2),\ e_2 \equiv \exp(-k_0^2 a_i^2/2R),$$

and

$$\omega_{\ell h} \equiv \omega_{pi}[1 + \omega_{pe}^2(1 + \Delta_e)/\omega_{ce}^2]^{-1/2}$$

is the lower hybrid frequency at x = 0,

and $\Delta_e = (Re_1-e_2)/(R-1)$ from Eq. (14a) assuming $\Delta_i = 1$. The growth rate is greatest for a given $\omega_{\ell h}$ when $k_0 a_i = [2R\ \ell n\ R/(R-1)]^{1/2}$. For R = 3, this corresponds to $k_0 a_i = 1.80$, and Eq. (38) gives $\omega_L = 0.06\ i\ \omega_{\ell h}$ and $k_L/k_0 = 0.27\ \omega_{\ell h}/\omega_{ci}$. Since Eq. (38) is derived using the local method and depends on inequality (21) for validity, we might expect that this mode will no longer exist when $\omega_{pi}^2/\omega_{ci}^2 \lesssim 200$, since then $k_L/k_0 < 2$. The nonlocal method confirms this expectation and gives results in good agreement with Eq. (38) when $\omega_{pi}^2/\omega_{ci}^2 > 200$ and $1.80 < k_0 a_i < 2.5$.

Since the mechanism for this instability is the deepening of the loss cone due to $d^2n/dx^2 < 0$, it does not depend on the periodic nature of the density profile, and we would expect it to occur even in an isolated slab of plasma. However, it would not occur in a plasma column with cylindrical symmetry; this may be seen by applying the Penrose criterion[22] at x = 0 to a plasma with distribution function $f_{0i}(\underset{\sim}{x},\underset{\sim}{v}) = g_i(v_\perp, v_\parallel, \underset{\sim}{x}_{gc})$ given by Eq. (11), and $n_i(\underset{\sim}{x}) = n_0 \exp(-x^2/L - y^2/L)$. Hence the instability of region C, like that of region B, is an artifact of the model density profile used. (see Appendix L)

When $k_0 a_i > 1.80$, $\Delta_e < 0$ (for R = 3), i.e. the electron density gradient is in the opposite direction of the ion guiding center density gradient, and a drift cone or drift cyclotron instability occurs with $\phi(x)$ centered at $0 < k_0 x/2\pi < 0.5$, rather than at $0.5 < k_0 x/2\pi < 1.0$, as occurs when $\Delta_e > 0$. This instability has higher growth rate than any of the purely growing ion instabilities when $2.5 \gtrsim k_0 a_i \gtrsim 4.0$ (regions D and E). At $k_0 a_i \approx 2.5$ (region D) finite $k_L \sim k_y$ must be used in Eq. (16) to describe the fastest growing mode by the local method, but at $k_0 a_i > 2.5$ (region E) $k_L = 0$ may be used. The ion density gradient is more important than the loss cone in driving this instability, but the loss cone is crucial in determining the electron density gradient Δ_e (which would be positive and much smaller in absolute value if there were no loss cone), on which the real part of the frequency (and indirectly the growth rate) depends.

For $k_0 a_i \gtrsim 4.0$, the electron density gradient is unimportant because Δ_e (which decreases exponentially with $k_0 a_i$) is negligibly small. Ion instabilities of the two-stream (or many-stream) type again become important (for $\omega_{pi}^2/\omega_{ci}^2 \gtrsim 50$), driven by the ion density profile [which for $k_0 a_i >> 1$ results in $f_{0i}(v_y)$ consisting of several

streams], and by the loss cone. These instabilities are purely growing at higher densities and lower $k_0 a_i$ (region F). At lower densities and higher $k_0 a_i$, when ion cyclotron effects (i.e. $\mathrm{Im}\,\omega \lesssim \omega_{ci}$) become important, the real frequencies are not zero (region G). Because $f_{0s}(\underset{\sim}{x},\underset{\sim}{v})$ is not very dependent on $\underset{\sim}{x}$ when $k_0 a_i \gg 1$, these modes tend to have $\phi(x)$ very spread out in x, with $k_y \approx k_x \approx k_0$. Hence they cannot be described by the local method, only by the nonlocal method.

Fastest-growing modes which cannot be described by the local method also exist at lower $k_0 a_i$, at sufficiently low densities (regions H,I,J,K). For regions H,I, and J, $\mathrm{Re}\,\omega \neq 0$; in region K, $\mathrm{Re}\,\omega = 0$. In regions I and K, like regions G and F, the instability is due entirely to the ions, and the electron density gradient (which is very small in these regions) plays no role. In regions H and J, the electron density gradient is important, and both the ions and electrons contribute to the mode. This may be seen by comparing the perturbed ion and electron densities for the fastest-growing mode in the different regions. In regions F, G, I, and K, the perturbed ion density is much greater, while in regions H and J, the perturbed ion and electron densities are comparable.

VII. Summary and Conclusions

Two methods were derived for finding the linear normal modes of electrostatic perturbations in a non-uniform Vlasov plasma. The methods complement each other, since the local method, using Eqs. (16) and (22) is valid only when the wavelength is much smaller than the scale length, while the nonlocal method, using Eqs. (27), (28), and (29), is most efficient when the wavelength is comparable to the scale length. Neither method depends on the Larmor radius being small compared to the scale length. Both methods were employed (each in its own regime of validity)

to find the normal modes for the drift cone and related instabilities in a loss cone plasma with density varying sinusoidally in space in a direction perpendicular to a uniform magnetic field, over a wide range of densities and density gradients.

An immediate application of these results is to computer simulations of the drift cone mode using periodic boundary conditions and a slab geometry, with sinusoidal density profile. Analytic solutions for the linear normal modes and frequencies for this model are useful in order to check the linear behavior of the simulation, before we explore the nonlinear behavior. Such simulations should have $k_0 a_i < 1.25$; otherwise we will introduce new instabilities that depend on the model, and would not occur in a cylindrical plasma column, for example, as discussed in Sec. VI.

If other models (including more realistic models of experiments) behave in a similar fashion to the model we have considered, then no qualitatively new behavior (not predicted by the small Larmor radius approximation) would occur if $(a_i/n)\ dn/dx < 1$. The condition is satisfied for the bulk of the plasma in the 2XIIB experiment.[3]

More generally, we have shown that it is possible to use a local method to find the normal modes and their growth rates, even when the Larmor radius is comparable to the scale length, provided the wavelength is much less than the scale length [and assuming we can find the equilibrium distribution $f_{0s}(\underline{x},\underline{v})$]. When the wavelength is comparable to the scale length, it is possible to use a matrix equation (not tri-diagonal in general) truncated at fairly low index p. Although the model we have solved does not resemble any experiment, it should be possible to use these methods to find the normal modes of experimental plasmas with Larmor radius comparable to scale length.

Historical Note

This report is the culmination of many years of work.

In 1969, Birdsall and Fuss (at Lawrence Livermore Lab.) began computer simulation of the drift-cyclotron loss-cone mode, using particles on a two-dimensional periodic mesh, starting with a sinusoidal density profile [Eq.(13) with $\Delta_i \simeq 1$, $k_o a_i/2\pi \simeq 0.1, 0.2$]. The checks between simulation and small-amplitude dispersion results (ω, k), calculated by N. Lindgren (Univ. of Calif., Berkeley) from existing local approximation theory[14], were fair, considering the difference in the model and theory.

Langdon (Univ. of Calif., Berkeley) then provided the exact or nonlocal theory, essentially Eq.(29), from which Birdsall and Fuss obtained a few sample solutions for ω, k (showing the multiple roots, as in Fig. 2) and ϕ (x,y) (showing the localization, as in Fig. 3), and these only for a ring distribution of velocities,

$$g_{i\perp}(v_\perp, \underset{\sim}{x}) = (2\pi v_o)^{-1} \delta(v_\perp - v_o)\, n_i(\underset{\sim}{x}),$$

rather than the loss cone distribution given by Eq.(11). While these results were presented in talks[4,5] and progress reports, they were incomplete.

Since 1973, Gerver derived the local method [eqs.(16) and (20)], and used these and the nonlocal dispersion relation, Eq.(29), with loss cone distributions of the form given by Eq.(11), to explore the broad range of parameters displayed in Fig. 4 through 7. This required considerable refinement of the numerical techniques used earlier, including the use of straight-line orbits [Eq.(35)], and extensive modifications of the root-finding routine. The physical identification of the various modes and the interpretations are also Gerver's.

Acknowledgments

We wish to express our appreciation to A.N. Kaufman and W.B. Kunkel for helpful discussions and suggestions, and to Richard Meyers for help in debugging the computer programs.

APPENDIX A

Derivation of Eq. (4):

Starting with Eq. (3) and the identities

$$\phi(\underset{\sim}{x}_s', t) = \int d\underset{\sim}{k}\ \tilde{\phi}(k) \exp(i\underset{\sim}{k} \cdot \underset{\sim}{x}_s' - i\omega t)$$

$$f_{0s}(\underset{\sim}{x}_s', \underset{\sim}{v}_s') = \int d\underset{\sim}{k}''\ \tilde{f}_s(\underset{\sim}{k}'', \underset{\sim}{v}_s') \exp(-i\underset{\sim}{k}'' \cdot \underset{\sim}{x}_s')$$

we obtain

$$\int d\underset{\sim}{k}'' \int d\underset{\sim}{k} \left[\sum_s \frac{4\pi q_s^2}{m_s} \int d\underset{\sim}{v} \int_{-\infty}^{0} d\tau\ i\underset{\sim}{k} \cdot \frac{\partial \tilde{f}_s}{\partial \underset{\sim}{v}_s'} (\underset{\sim}{k}'', \underset{\sim}{v}_s') \right.$$

$$\left. \tilde{\phi}(\underset{\sim}{k}) \exp(i\underset{\sim}{k} \cdot \underset{\sim}{x}_s' - i\underset{\sim}{k}'' \cdot \underset{\sim}{x}_s' - i\omega t) - k^2\, \tilde{\phi}(\underset{\sim}{k}) \right] = 0$$

$$\int d\underset{\sim}{k}\ \tilde{\phi}(\underset{\sim}{k}) \left(\int d\underset{\sim}{k}'\ \exp(-i\underset{\sim}{k}'' \cdot \underset{\sim}{x} + i\underset{\sim}{k} \cdot \underset{\sim}{x}) \left\{ \sum_s \frac{4\pi q_s^2}{m_s} \right.\right.$$

$$\left.\left. \int d\underset{\sim}{v} \int_{-\infty}^{0} d\tau\ i\underset{\sim}{k} \cdot \frac{\partial \tilde{f}_s}{\partial \underset{\sim}{v}_s} (\underset{\sim}{k}'', \underset{\sim}{v}_s') \exp[i(\underset{\sim}{k}'' - \underset{\sim}{k}) \cdot (\underset{\sim}{x} - \underset{\sim}{x}_s') - i\omega\tau] \right\} \right.$$

$$\left. - k^2 \right) = 0$$

Let $\underset{\sim}{k}' \equiv \underset{\sim}{k} - \underset{\sim}{k}''$, and we obtain Eq. (4).

Derivation of Eq. (7):

Starting with the identity

$$\tilde{f}_s(\underset{\sim}{k}, \underset{\sim}{v}) = \int d\underset{\sim}{x}\ f_s(\underset{\sim}{x}, \underset{\sim}{v}) \exp(i\underset{\sim}{k} \cdot \underset{\sim}{x})$$

$$= \int d\underset{\sim}{x}\ g_s(v_\perp, v_\parallel, \underset{\sim}{x}_{gc}) \exp(i\underset{\sim}{k} \cdot \underset{\sim}{x})$$

$$= \int d\underset{\sim}{x}_{gc}\ g_s(v_\perp, v_\parallel, \underset{\sim}{x}_{gc}) \exp(i\underset{\sim}{k} \cdot \underset{\sim}{x}_{gc})$$

$$\exp[-i\underset{\sim}{k} \cdot (\underset{\sim}{v} \times \hat{B}_0)\omega_{cs}^{-1}]$$

$$= \exp[-i\underset{\sim}{k} \cdot (\underset{\sim}{v} \times \hat{\underset{\sim}{B}}_0)\omega_{cs}^{-1}]\tilde{g}_s\ (v_\perp, v_\parallel, \underset{\sim}{k})$$

using $\underset{\sim}{x}_{gc} \equiv \underset{\sim}{x} + \underset{\sim}{v} \times \hat{\underset{\sim}{B}}_0\ \omega_{cs}^{-1}$

Derivation of Eq. (9):

We put Eq. (7) into Eq. (6) to obtain

$$D_s(\underset{\sim}{x},\underset{\sim}{k},\omega) = \frac{4\pi q_s^2}{m_s} \exp(-i\underset{\sim}{k} \cdot \underset{\sim}{x}) \int d\underset{\sim}{k}' \exp(i\underset{\sim}{k}' \cdot \underset{\sim}{x})$$

$$\int d\underset{\sim}{v} \int_{-\infty}^{0} d\tau\ i\underset{\sim}{k} \cdot \frac{\partial}{\partial \underset{\sim}{v}_s'} \{\exp[-i(\underset{\sim}{k} - \underset{\sim}{k}') \cdot (\underset{\sim}{v}_s' \times \hat{\underset{\sim}{B}}_0)\omega_{cs}^{-1}]$$

$$\tilde{g}_s(v_\perp, v_\parallel, \underset{\sim}{k} - \underset{\sim}{k}')\}\exp(i\underset{\sim}{k}' \cdot (\underset{\sim}{x}_s' - \underset{\sim}{x}) - i\omega\tau] \tag{A1}$$

We evaluate $\frac{\partial}{\partial \underset{\sim}{v}_s'}$ { }

$$\frac{\partial}{\partial \underset{\sim}{v}_s'} \{\quad\} = \exp[-i(\underset{\sim}{k} - \underset{\sim}{k}') \cdot (\underset{\sim}{v} \times \hat{\underset{\sim}{B}}_0)\omega_{cs}^{-1}]$$

$$\{\frac{\partial \tilde{g}_s}{\partial \underset{\sim}{v}_s'} - \frac{i\tilde{g}_s}{\omega_{cs}} \frac{\partial}{\partial \underset{\sim}{v}_s'} [(\underset{\sim}{k} - \underset{\sim}{k}') \cdot (\underset{\sim}{v}_s' \times \hat{\underset{\sim}{B}}_0]\} \tag{A2}$$

where the arguments of $\tilde{g}_s$ and its derivatives are always taken to be $(v_\perp, v_\parallel, \underset{\sim}{k} - \underset{\sim}{k}')$.

$$\frac{\partial \tilde{g}_s}{\partial \underset{\sim}{v}_s'} = \hat{v}_{\perp s} \frac{\partial \tilde{g}_s}{\partial v_\perp} + \hat{B}_0 \frac{\partial \tilde{g}_s}{\partial v_\parallel} \tag{A3}$$

$$\frac{\partial}{\partial \underset{\sim}{v}_s'} [(\underset{\sim}{k} - \underset{\sim}{k}') \cdot (\underset{\sim}{v}_s' \times \hat{\underset{\sim}{B}}_0)] = \frac{\partial}{\partial \underset{\sim}{v}_s'} \{\underset{\sim}{v}_s' \cdot [\hat{\underset{\sim}{B}}_0 \times (\underset{\sim}{k} - \underset{\sim}{k}')]\}$$

$$= \hat{\underset{\sim}{B}}_0 \times (\underset{\sim}{k} - \underset{\sim}{k}') \tag{A4}$$

where $\hat{v}_{\perp s}' \equiv \underset{\sim}{v}_{\perp s}'/|\underset{\sim}{v}_{\perp s}'|$, and $\underset{\sim}{v}_{\perp s}' \equiv \underset{\sim}{v}_{\perp s}' - (\underset{\sim}{v}_s' \cdot \hat{B}_0)\hat{B}_0$

Putting Eqs. (A2), (A3) and (A4) into Eq. (A1), we find

$$D_s(\underset{\sim}{x},\underset{\sim}{k},\omega) = \frac{4\pi q_s^2}{m_s} \exp(-i\underset{\sim}{k} \cdot \underset{\sim}{x}) \int d\underset{\sim}{k}' \exp(i\underset{\sim}{k}' \cdot \underset{\sim}{x})$$

$$d\underset{\sim}{v} \int_{-\infty}^{0} d\tau \exp[-i(\underset{\sim}{k} - \underset{\sim}{k}') \cdot (\underset{\sim}{v}_s' \times \hat{B}_0)\omega_{cs}^{-1}]$$

$$\{i\underset{\sim}{k} \cdot \hat{v}_{\perp s}' \frac{\partial \tilde{g}_s}{\partial v_\perp} + \underset{\sim}{k} \cdot [\hat{B}_0 \times (\underset{\sim}{k} - \underset{\sim}{k}')]\tilde{g}_s \omega_{cs}^{-1}$$

$$+ i k_\parallel \frac{\partial \tilde{g}_s}{\partial v_\perp}\} \exp[i\underset{\sim}{k}' \cdot (\underset{\sim}{x}_s' - \underset{\sim}{x}) - i\omega\tau] \qquad \text{(A5)}$$

We can use the fact that $\underset{\sim}{k} \cdot (\hat{B}_0 \times \underset{\sim}{k}) = 0$ to replace $\underset{\sim}{k} \cdot [\hat{B}_0 \times (\underset{\sim}{k}-\underset{\sim}{k}')]$ by $\underset{\sim}{k} \cdot (\hat{B}_0 \times \underset{\sim}{k}')$ in Eq. (A5). We then put in the expression for the particle orbits, Eq. (8), and obtain

$$D_s(\underset{\sim}{x},\underset{\sim}{k},\omega) = \frac{4\pi q_s^2}{m_s} \exp(-i\underset{\sim}{k} \cdot \underset{\sim}{x}) \int d\underset{\sim}{k}' \exp(i\underset{\sim}{k}' \cdot \underset{\sim}{x})$$

$$\int d\underset{\sim}{v} \exp[i(\underset{\sim}{k}' - \underset{\sim}{k}) \cdot (\underset{\sim}{v} \times \hat{B}_0)\omega_{cs}^{-1}] \int_{-\infty}^{0} d\tau$$

$$\exp[i\underset{\sim}{k} \cdot \underset{\sim}{v}_\perp \omega_{cs}^{-1} \sin \omega_{cs}\tau + i \underset{\sim}{k} \cdot (\underset{\sim}{v} \times \hat{B}_0) \omega_{cs}^{-1}$$

$$(1 - \cos \omega_{cs}\tau) + ik_\parallel v_\parallel - i\omega\tau]$$

$$\{[i \underset{\sim}{k} \cdot \hat{v}_\perp \cos \omega_{cs}\tau + i \underset{\sim}{k} \cdot (\hat{v}_\perp \times \hat{B}_0) \sin \omega_{cs}\tau] \frac{\partial \tilde{g}_s}{\partial v_\perp}$$

$$- \underset{\sim}{k} \cdot (\hat{B}_0 \times \underset{\sim}{k}') \omega_{cs}^{-1} \tilde{g}_s + ik_\parallel \frac{\partial \tilde{g}_s}{\partial v_\parallel} \}$$

$$= \frac{4\pi q_s^2}{m_s} \exp(-i\underset{\sim}{k} \cdot \underset{\sim}{x}) \int d\underset{\sim}{k}' \exp(i\underset{\sim}{k}' \cdot \underset{\sim}{x}) \int d\underset{\sim}{v}$$

$$\exp[i\underset{\sim}{k}' \cdot (\underset{\sim}{v} \times \hat{B}_0) \omega_{cs}^{-1}] \{[\underset{\sim}{k} \cdot (\underset{\sim}{k}' \times \hat{B}_0) \omega_{cs}^{-1} \tilde{g}_s$$

$$+ ik_\parallel \frac{\partial \tilde{g}_s}{\partial v_\parallel}]I_1 + i \frac{\partial \tilde{g}_s}{\partial v_\perp} I_2\} \qquad \text{(A6)}$$

where $I_1 \equiv \int_{-\infty}^{0} d\tau \exp[i\underset{\sim}{k} \cdot \underset{\sim}{v}_\perp \omega_{cs}^{-1} \sin \omega_{cs}\tau$

$-i\underset{\sim}{k} \cdot (\underset{\sim}{v} \times B_0) \omega_{cs}^{-1} \cos \omega_{cs}\tau + ik_\parallel v_\parallel \tau - i\omega\tau]$

$I_2 \equiv \int_{-\infty}^{0} d\tau \exp[i\underset{\sim}{k} \cdot \underset{\sim}{v}_\perp \omega_{cs}^{-1} \sin \omega_{cs}\tau - i\underset{\sim}{k} \cdot (\underset{\sim}{v} \times \hat{B}_0)\omega_{cs}^{-1}$

$\cos \omega_{cs}\tau + ik_\parallel v_\parallel \tau - i\omega\tau] [\underset{\sim}{k} \cdot \hat{v}_\perp \cos \omega_{cs}\tau + \underset{\sim}{k} \cdot (\hat{v}_\perp \times \hat{B}_0) \sin \omega_{cs}\tau]$

We define an azimuthal angle ϕ by

$$\underset{\sim}{k} \cdot \underset{\sim}{v}_\perp = k_\perp v_\perp \cos \phi$$

$$-\underset{\sim}{k} \cdot (\underset{\sim}{v} \times \hat{B}_0) = k_\perp v_\perp \sin \phi$$

Then

$$I_1 = \int_{-\infty}^{0} d\tau \exp[ik_\perp v_\perp \omega_{cs}^{-1} \sin(\phi + \omega_{cs}\tau) + ik_\parallel v_\parallel \tau - i\omega\tau]$$

$$I_2 = \int_{-\infty}^{0} d\tau \exp[ik_\perp v_\perp \omega_{cs}^{-1} \sin(\phi + \omega_{cs}\tau) + ik_\parallel v_\parallel \tau - i\omega\tau]$$

$$k_\perp \cos(\phi + \omega_{cs}\tau)$$

These integrals can be evaluated by using the Bessel function identity

$$\sum_\ell J_\ell(x) \exp(i\ell\theta) = \exp(i\, x \sin\theta) \qquad \text{(A7)}$$

$$I_1 = \sum_\ell J_\ell(k_\perp v_\perp/\omega_{cs}) \int_{-\infty}^{0} d\tau \exp(i\ell\phi + i\ell\omega_{cs} - i\omega\tau + ik_\parallel v_\parallel \tau]$$

$$= i \sum_\ell J_\ell(k_\perp v_\perp/\omega_{cs}) \exp(i\ell\phi) (\omega - \ell\omega_{cs} - k_\parallel v_\parallel)^{-1} \qquad \text{(A8)}$$

$$I_2 = (k_\perp/2) \sum_\ell J_\ell(k_\perp v_\perp/\omega_{cs}) \int_{-\infty}^{0} d\tau \sum_\pm \exp[i(\ell \pm 1)\phi$$

$$+ i(\ell \pm 1)\omega_{cs}\tau - i\omega\tau + ik_\parallel v_\parallel \tau]$$

$$= (k_\perp/2) \sum_\ell \sum_\pm J_{\ell\pm1}(k_\perp v_\perp/\omega_{cs}) \int_{-\infty}^{0} d\tau \exp[i\ell\phi$$

$$+ i\ell\omega_{cs}\tau - i\omega\tau + ik_\parallel v_\parallel \tau]$$

$$= i(k_\perp/2) \sum_\ell \sum_\pm J_{\ell\pm1}(k_\perp v_\perp/\omega_{cs}) \exp(i\ell\phi)$$

$$(\omega - \ell\omega_{cs} - k_\parallel v_\parallel)^{-1}$$

$$= i(\omega_{cs}/v_\perp) \sum_\ell \ell\, J_\ell(k_\perp v_\perp/\omega_{cs}) \exp(i\ell\phi)\ (\omega - \ell\omega_{cs} - k_\parallel v_\parallel)^{-1} \qquad \text{(A9)}$$

where we have used the Bessel function identity

$$(2\ell/x)J_\ell(x) = J_{\ell+1}(x) + J_{\ell-1}(x)$$

The $d\underset{\sim}{v}$ in Eq. (A6) can be replaced by $v_\perp dv_\perp d\phi$, and the integration over ϕ performed. We define α as the angle between $\underset{\sim}{k}_\perp'$ and $\underset{\sim}{k}_\perp$

$$\exp(i\alpha) = \hat{k}_\perp \cdot(\hat{k}_\perp' + i\,\hat{B}_0 \times \hat{k}_\perp') \qquad \text{(A10)}$$

Then

$$-\underset{\sim}{k}' \cdot(\underset{\sim}{v} \times \hat{B}_0) = k_\perp' v_\perp \sin(\phi - \alpha)$$

and we can use the identity (A7) to evaluate

$$\exp[i\underset{\sim}{k}' \cdot(\underset{\sim}{v} \times \hat{B}_0)\ \omega_{cs}^{-1}] = \exp[-ik_\perp' v_\perp\ \omega_{cs}^{-1} \sin(\phi - \alpha)]$$

$$= \sum_m J_m(k_\perp' v_\perp/\omega_{cs}) \exp[-im(\phi - \alpha)] \qquad \text{(A11)}$$

The ϕ dependence of I_1 and I_2, given by Eqs. (A8) and (A9), appears only in the factor $\exp(i\ell\phi)$. The only other ϕ dependence of the integrand in Eq. (A6) is in the term evaluated in Eq.(A11). So the integration over ϕ that must be done in Eq. (A6) is

$$\int d\phi \, \exp[-im(\phi\alpha)] \exp(i\ell\phi) = 2\pi \, \delta_{\ell m} \exp(i\ell\alpha) \qquad \text{(A12)}$$

Putting Eqs. (A8) through (A12) into Eq. (A6), we obtain Eq. (9).

APPENDIX B

Derivation of Eq. (10):

In Eq. (9), $\tilde{g}_s(v_\perp, v_\parallel, \underset{\sim}{k} - \underset{\sim}{k}')$ is negligible for $|\underset{\sim}{k} - \underset{\sim}{k}'| \gg L^{-1}$, where L is the scale length, and for $v_\perp \gg v_s$, where v_s is the thermal velocity. If $a_s \ll L$, where $a_s \equiv v_s/\omega_{cs}$ is the Larmor radius, then $\tilde{g}_s(v_\perp, v_\parallel, \underset{\sim}{k} - \underset{\sim}{k}')$ is negligible unless $(\underset{\sim}{k} - \underset{\sim}{k}') \cdot (\underset{\sim}{v} \times \hat{B}_0)\omega_{cs}^{-1} \ll 1$, so we can make the approximation

$$\exp[-i(\underset{\sim}{k} - \underset{\sim}{k}') \cdot (\underset{\sim}{v} \times \hat{B}_0)\omega_{cs}^{-1}] \approx 1 \tag{B1}$$

in Eq. (A5) without changing the integral very much. We use the definition of $\tilde{g}_s$ after Eq. (7) to obtain the identities

$$\int d\underset{\sim}{k}' \exp[i(\underset{\sim}{k}' - \underset{\sim}{k}) \cdot \underset{\sim}{x}] \, \partial \tilde{g}_s(\underset{\sim}{k} - \underset{\sim}{k}')/\partial v_\parallel$$

$$= \partial g_s(\underset{\sim}{x})/\partial v_\parallel$$

$$\int d\underset{\sim}{k}' \exp]i(\underset{\sim}{k}' - \underset{\sim}{k}) \cdot \underset{\sim}{x}] \partial \tilde{g}_s(\underset{\sim}{k} - \underset{\sim}{k}')/\partial v_\perp$$

$$= \partial g_s(\underset{\sim}{x})/\partial v_\perp$$

$$\int d\underset{\sim}{k}' \, [\hat{B}_0 \times i(\underset{\sim}{k}' - \underset{\sim}{k})] \exp[i(\underset{\sim}{k}' - \underset{\sim}{k}) \cdot \underset{\sim}{x}] \tilde{g}_s(\underset{\sim}{k} - \underset{\sim}{k}') = \hat{B}_0 \times \nabla g_s(\underset{\sim}{x})$$

which can be used together with Eq. (B1) to perform the integration over $\underset{\sim}{k}'$ in Eq. (A5), yielding

$$D_s(\underset{\sim}{x}, \underset{\sim}{k}, \omega) = (4\pi q_s^2/m_s) \exp(-i\underset{\sim}{k} \cdot \underset{\sim}{x}) \int dv \int_{-\infty}^{0} d\tau$$

$$\exp[i\underset{\sim}{k} \cdot (\underset{\sim}{x}_s' - \underset{\sim}{x}) - i\omega\tau] \, [i\underset{\sim}{k} \cdot \hat{v}_{\perp s}'(\partial g_s/\partial v_\perp)$$

$$+ i\underset{\sim}{k} \cdot (\hat{B}_0 \times \nabla g_s) \, \omega_{cs}^{-1} + ik_\parallel \, (\partial g_s/\partial v_\parallel)] \tag{B2}$$

Eq. (8) can be put into Eq. (B2), and the integration over τ and ϕ performed just as in Appendix A, yielding Eq. (10). The factor $\exp[i\underset{\sim}{k}' \cdot (\underset{\sim}{v} \times \hat{B}_0)\ \omega_{cs}^{-1}]$ which appears in Eq. (A6) must be replaced by $\exp[i\underset{\sim}{k} \cdot (\underset{\sim}{v} \times \hat{B}_0)\ \omega_{cs}^{-1}]$, which is justified by Eq. (B1); otherwise the derivation is completely analogous.

APPENDIX C

Derivation of Eq. (12):

For a Maxwellian $g_{s\perp}$

$$g_{s\perp}(v_\perp,\underset{\sim}{x}) = (2\pi v_s^2)^{-1} \exp(-v_\perp^2/2v_s^2) n_s(\underset{\sim}{x})$$

and a slab $n_s(\underset{\sim}{x})$ independent of y and z, we have

$$\int dv_\parallel \tilde{g}_s(v_\perp,\underset{\sim}{k} - \underset{\sim}{k}') = (2\pi v_s^2)^{-1} \exp(-v_\perp^2/2v_s^2)\tilde{n}_s(k_x' - k_x)$$

$$\delta(k_y - k_y')\delta(k_z')$$

where $k_z = 0$ by assumption, $\tilde{g}_s$ is defined after Eq. (7) and $\tilde{n}_s$ is defined after Eq. (12). Then Eq. (9) becomes (after integrating over k_y', k_z' and $v_\parallel$)

$$D_s(\underset{\sim}{x},\underset{\sim}{k},\omega) = (4\pi q_s^2/m_s) \int_{-\infty}^{\infty} dk_x' \exp[i(k_x' - k_x)x]$$

$$\int_0^\infty 2\pi v_\perp dv_\perp \sum_\ell \exp(i\ell\alpha)\, J_\ell(k_\perp v_\perp/\omega_{cs}) J_\ell(k_\perp' v_\perp/\omega_{cs})$$

$$(\omega - \ell\omega_{cs})^{-1} (2\pi v_s^2)^{-1} \exp(-v_\perp^2/2v_s^2)$$

$$[ik_y(k_x - k_x') + \ell\omega_{cs}/v_s^2]\, \tilde{n}_s(k_x' - k_x)$$

We define $k'' \equiv k_x' - k_x$, change the variable of integration from k_x' to k'', and use

$$v_s^{-2} \int_0^\infty v_\perp \exp(-v_\perp^2/2v_s^2)\, J_\ell(kv_\perp/\omega_{cs}) J_\ell(k' v_\perp/\omega_{cs})\, dv_\perp$$

$$= \exp[-(k^2 + k'^2)\, v_s^2/2\omega_{cs}^2]\, I_\ell(k\, k'\, v_s^2/\omega_{cs}^2)$$

[from identity 6.633.2 in Gradshteyn and Ryzhik[23]] to obtain Eq. (12).

APPENDIX D

Derivation of Eq. (14):

$$\rho_0(\underset{\sim}{x}) \sum_s q_s \int d\underset{\sim}{v}\ f_{0s}(\underset{\sim}{x},\underset{\sim}{v}) = 0 \text{ for all } \underset{\sim}{x} \tag{D1}$$

for no net equilibrium charge density.

$$f_{0s}(\underset{\sim}{x},\underset{\sim}{v}) = g_s(v_\perp, v_\parallel, \underset{\sim}{x}_{gc}) \tag{D2}$$

where $\underset{\sim}{x}_{gc} \equiv \underset{\sim}{x} + \underset{\sim}{v} \times \hat{B}_0\ \omega_{cs}^{-1}$

so $x_{gc} = x + v_y/\omega_{cs}$

$y_{gc} = y - v_x/\omega_{cs}$

Using Eq. (D2), the left hand side of Eq. (D1) becomes

$$\rho_0(\underset{\sim}{x}) = \sum_s q_s \int dv_x\ dv_y\ g_{s\perp}(v_\perp, \underset{\sim}{x}_{gc})$$

$$= \sum_s q_s (2\pi v_s^2)^{-1} \int dv_x\ dv_y\ \exp(-v_\perp^2/2v_s^2)\ n_s(\underset{\sim}{x}_{gc})$$

for Maxwellian $g_{\perp s}$. For a plasma slab, $n_s(\underset{\sim}{x})$ independent of y and z,

$$\rho_0(x) = \sum_s q_s (2\pi v_s^2)^{-1} \int dv_x\ dv_y\ \exp(-v_\perp^2/2v_s^2) n_s(x + v_y/\omega_{cs}) \tag{D3}$$

For a sinusoidal density profile,

$$n_s(x) = n_{0s}(1 + \Delta_s \cos k_0 x)$$

Eq. (D3) becomes

$$\rho_0(x) = \sum_s q_s (2\pi v_s^2)^{-1} \int dv_x\, dv_y\, \exp(-v_x^2/2v_s^2)$$

$$\exp(-v_y^2/2v_s^2)\; n_{0s}[1 + \Delta_s \cos(k_0 x + k_0 v_y/\omega_{cs})]$$

Integrating over v_x

$$\rho_0(x) = \sum_s n_{0s}\, q_s (2\pi v_s^2)^{-1/2} \int dv_y\, \exp(-v_y^2/2v_s^2)$$

$$[1 + \Delta_s \cos(k_0 x + k_0 v_y/\omega_{cs})]$$

$$= \sum_s n_{0s}\, q_s (2\pi v_s^2)^{-1/2} \int dv_y\, \exp(-v_y^2/2v_s^2)$$

$$[1 + \Delta_s \exp(ik_0 x)\exp(ik_0 v_y/\omega_{cs})/2$$

$$+ \Delta_s \exp(-ik_0 x)\exp(-ik_0 v_y/\omega_{cs})/2]$$

Since $\rho_0(x)$ vanishes for all x,

$$\int_0^{2\pi/k_0} \rho_0(x)\, dx = 2\pi k_0^{-1} \sum_s n_{0s} q_s = 0 \text{ or Eq. (14b) and}$$

$$\int_0^{2\pi/k_0} \rho_0(x) \exp(\pm ik_0 x) dx =$$

$$\pi k_0^{-1} \sum_s n_{0s}\, q_s\, \Delta_s (2\pi v_s^2)^{-1/2} \int dv_y \exp(-v_y^2/2v_s^2) \exp(\mp ik_0 v_y/\omega_{cs})$$

$$= \pi\, k_0^{-1} \sum_s n_{0s}\, q_s\, \Delta_s \exp(-k_0^2 v_y^2/2\omega_{cs}^2) = 0$$

or Eq. (14a).

APPENDIX E

Solution of Eq. (18):

$$A\frac{\partial^2\psi}{\partial x^2} + Bx\frac{\partial\psi}{\partial x} + Cx^2\psi + \lambda\psi = 0 \tag{E1}$$

Assume $\psi = (\sum_{m=0}^{n} a_m x^m)\exp(-\beta x^2/2)$ (E2)

Then $\frac{\partial\psi}{\partial x} = (\sum_{m=0}^{n-1} (m+1)a_{m+1}x^m)\exp(-\beta x^2/2)$

$$-(\beta \sum_{m=0}^{n} a_m x^{m+1})\exp(-\beta x^2/2)$$

$$= [\sum_{m=0}^{n-1} (m+1)a_{m+1}x^m - \beta\sum_{m=1}^{n+1} a_{m-1}x^m]\exp(-\beta x^2/2)$$

$$= \{\sum_{m=1}^{n+1} [(m+1)a_{m+1} - \beta a_{m-1}] x^m + a_1\}\exp(-\beta x^2/2) \tag{E3}$$

$$\partial^2\psi/\partial x^2 = \Bigg(\sum_{m=2}^{n+2} \{(m+1)[(m+2)a_{m+2} - \beta a_m]$$

$$- \beta(m a_m - \beta a_{m-2})\} x^m$$

$$+ [2(3a_3 - \beta a_1) - \beta a_1] x + 2a_2 - \beta a_0\Bigg) \exp(-\beta x^2/2)$$

$$= \{\sum_{m=2}^{n+2} [(m^2 + 3m + 2)a_{m+2} - \beta(2m + 1)a_m$$

$$+ \beta^2 a_{m-2}]x^m + 3(2a_3 - \beta a_1) x + (2a_2 - \beta a_0)\} \exp(-\beta x^2/2) \tag{E4}$$

$$x\,(\partial\psi/\partial x) = [\sum_{m=2}^{n+2} (ma_m - \beta a_{m-2})x^m + a_1 x] \exp(-\beta x^2/2) \tag{E5}$$

$$x^2\psi = \left(\sum_{m=2}^{n+2} a_{m-2}x^m\right) \exp(-\beta x^2/2) \tag{E6}$$

Putting Eqs. (E4), (E5) and (E6) into Eq. (E1) and dividing through by $\exp(-\beta x^2/2)$ yields

$$\sum_{m=2}^{n+2} x^m\{A[m^2 + 3m + 2)a_{m+2} - \beta(2m + 1)a_m$$

$$+ \beta^2 a_{m-2}] + B[ma_m - \beta a_{m-2}]$$

$$+ C a_{m-2} + \lambda a_m\}$$

$$+ [3A(2a_3 - \beta a_1) + B a_1 + \lambda a_1] x$$

$$+ [A(2a_2 - \beta a_0) + \lambda a_0] = 0 \tag{E7}$$

This can be satisfied for all x only if each term in the polynomial vanishes. So

$$A(2a_2 - \beta a_0) + \lambda a_0 = 0 \tag{E8}$$

$$3A(2a_3 - \beta a_1) + Ba_1 + \lambda a_1$$

$$= 6Aa_3 + (B + \lambda - 3A\beta)a_1 = 0 \tag{E9}$$

$$A(m^2 + 3m + 2)a_{m+2} + [Bn - \beta A(2m + 1) + \lambda]a_m$$

$$+ (C - \beta B + \beta^2 A) a_{m-2} = 0 \quad \text{for } n + 2 \geq m \geq 2 \tag{E10}$$

Setting $m = n + 2$ in Eq. (E10) yields (since $a_m = 0$ for $m > n$)

$$C - \beta B + \beta^2 A = 0 \tag{E11}$$

Then Eq. (E10) becomes

$$A(m+2)(m+1)a_{m+2} + [Bm - \beta A(2m+1) + \lambda]a_m = 0 \qquad \text{(E12)}$$

where $\beta = (B/2A) + [(B/2A)^2 - C/A]^{1/2}$ (E13)

[the + sign is taken in Eq. (E13) so that Re $\beta > 0$ and $\psi(x) \to 0$ as $x \to \pm \infty$].

Taking m=n in Eq. (E12) yields

$$Bn - \beta A(2n + 1) + \lambda = 0$$

or

$$\lambda = \beta A(2n + 1) - Bn \qquad \text{(E14)}$$

Taking $m = n - 1$ yields $a_{n-1} = 0$, which further implies $a_{n-3} = a_{n-5} = \ldots = 0$, so ψ must be either even or odd in x.

For $0 \leq m \leq n - 2$, Eqs.(E12) and (E14) give the recursion relation for the coefficients a_m

$$\frac{a_m}{a_{m+2}} = \frac{A(m+2)(m+1)}{-Bm + \beta A(2m+1) + \lambda}$$

$$= \frac{A(m+2)(m+1)}{B(n-m) - 2\beta A(n-m)}$$

$$= \frac{(m+2)(m+1)}{(n-m)(B/A - 2\beta)}$$

$$\frac{a_m}{a_{m+2}} = \frac{(m+2)(m+1)}{2(m-n)\eta} \qquad \text{(E15)}$$

where $\eta = \beta - B/2A$

$$= [(B/2A)^2 - C/A]^{1/2} \tag{E16}$$

Eq. (E15) is the recursion relation for the Hermite polynomial

$$\sum_m a_m x^m = H_n(\eta^{1/2} x) \tag{E17}$$

So the solutions are

$$\psi_n(x) = H_n(\eta^{1/2} x)\exp(-\beta x^2/2)$$

$$\lambda_n = \beta A(2n + 1) - Bn = (2n\eta + \beta)A \tag{E18}$$

with β and η defined by Eqs. (E13) and (E16).

Appendix F

Sketch of Proof that Local Method is Valid When Inequality (21) is Satisfied:

We wish to show that inequality (21)

$$L(k_L^2 + k_y^2)^{1/2} >> 1$$

is the criterion for being able to neglect the terms in Eq.(17) involving higher derivatives of D, so that Eq.(20) is a good approximation to at least some of the solutions of Eq.(15). We will consider a general dispersion function $D(x, k_x, \omega)$, not just the D appropriate to the model considered in Sec.III. However, we will not attempt to make our results as general as possible, but will make various physically reasonable assumptions about $D(x, k_x, \omega)$ when convenient. Sometimes we will make mathematical assumptions about $D(x,k_x, \omega)$ and the physical content of these assumptions will not be obvious; in such cases we cannot prove that the assumptions will be valid for all models of physical interest (or even for the model considered in Sec.III), but we conjecture that the assumptions will be valid for most models of physical interest.

To simplify notation, we will use $\psi(x) \equiv \exp(-ik_L x)\ \phi(x + x_L)$ and $Q(x,k_x, \omega) \equiv D(x + x_L, k_x + k_L, \omega)$. In this notation, Eq.(15) is

$$Q(x, -i \frac{\partial}{\partial x}, \omega)\ \psi(x) = 0 \qquad \text{(F1)}$$

and Eq.(16) is

$$Q(0,0, \omega_L) = \frac{\partial Q}{\partial x}(0,0, \omega_L) = \frac{\partial Q}{\partial k_x}(0,0,\omega_L) = 0$$

Anticipating our result that Eq.(17) is nearly valid without the higher derivatives of D, we expand Q and ψ in perturbation series, and write

$$Q(x, k_x, \omega_n) = Q^{(0)}(x, k_x, \omega_L) + \lambda_n + Q^{(1)}(x, k_x, \omega_n)$$

[The subscript n indicates we are looking at a normal mode whose frequency ω_n is given to lowest order by Eq.(20b)]

$$Q^{(0)}(x,k_x,\omega_L) \equiv -1/2\,\frac{\partial^2 Q}{\partial k_x^2}(0,0,\omega_L)\,k_x^2 - i\,\frac{\partial^2 Q}{\partial k_x \partial x}(0,0,\omega_L)\,xk_x + 1/2\,\frac{\partial^2 Q}{\partial x^2}(0,0,\omega_L)x^2$$

$$\lambda_n = \lambda_n^{(0)} + \lambda_n^{(1)} + \ldots = (\omega_n - \omega_L)\,\frac{\partial Q^{(0)}}{\partial \omega}(0,0,\omega_L)$$

$$\lambda_n^{(m)} \equiv \Delta\omega_n^{(m)}\,\frac{\partial Q^{(0)}}{\partial \omega}\,(0,0,\omega_L)$$

$$\omega_n = \omega_L + \Delta\omega_n^{(0)} + \Delta\omega_n^{(1)} + \ldots$$

$$\psi_n(x) = \psi_n^{(0)}(x) + \psi_n^{(1)}(x) + \ldots$$

Note that $Q^{(1)}$ contains all of the terms neglected in Eq.(17). As a result of these terms, ω_n and ψ_n have small corrections $\Delta\omega_n^{(1)} + \Delta\omega_n^{(2)} + \ldots$ and $\psi_n^{(1)} + \psi_n^{(2)} + \ldots$. Eq.(F1) may be written

$$(Q^{(0)} + Q^{(1)} + \lambda_n^{(0)} + \lambda_n^{(1)} + \ldots)(\psi_n^{(0)} + \psi_n^{(1)} + \ldots) = 0 \qquad \text{(F2)}$$

The zero-order terms of Eq.(F2) give Eq.(18), or in our notation

$$[Q^{(0)}(x, -i\,\frac{\partial}{\partial x}, \omega_L) + \lambda_n^{(0)}]\;\psi_n^{(0)}(x) = 0 \qquad \text{(F3)}$$

where $\psi_n^{(0)}$ and $\lambda_n^{(0)}$ are given by Eq.(19). The first-order terms of Eq.(F2) give

$$(Q^{(0)} + \lambda_n^{(0)})\;\psi_n^{(1)} + (Q^{(1)} + \lambda_n^{(1)})\;\psi_n^{(0)} = 0 \qquad \text{(F4)}$$

Following the usual procedure of quantum mechanical perturbation theory, we expand $\psi_n^{(1)}(x)$ in the eigenfunctions $\psi_m^{(0)}(x)$ of $Q^{(0)}$ [given by Eq.(19a)]:

$$\psi_n^{(1)}(x) = \sum_{m=0}^{\infty} c_{mn}\,\psi_m^{(0)}(x) \qquad \text{(F5)}$$

where $c_{mn} \equiv \int \psi_m^{(0)*}(x)\;\psi_n^{(1)}(x)\,dx \Big/ \int |\psi_m^{(0)}(x)|^2 dx$

Then, using $Q^{(0)}\,\psi_m^{(0)} = -\,\lambda_m^{(0)}\,\psi_m^{(0)}$,

Eq.(F4) becomes

$$\sum_{m=0}^{\infty} \left(\lambda_n^{(0)} - \lambda_m^{(0)} \right) c_{mn} \psi_m^{(0)} + \left(Q^{(1)} + \lambda_n^{(1)} \right) \psi_n^{(0)} = 0 \qquad \text{(F6)}$$

Eq.(F6) can be used to find $\lambda_n^{(1)}$ and c_{mn} for $m \neq n$. If we left-multiply Eq.(F6) by $\psi_n^{(0)*}$ and integrate over x, we obtain

$$\int \psi_n^{(0)*} Q^{(1)} \psi_n^{(0)} \, dx + \lambda_n^{(1)} \int |\psi_n^{(0)}|^2 \, dx = 0$$

$$\text{or} \quad \lambda_n^{(1)} = - \int \psi_n^{(0)*} Q^{(1)} \psi_n^{(0)} \, dx \Big/ \int |\psi_n^{(0)}|^2 \, dx \qquad \text{(F7)}$$

Left-multiplying Eq.(F6) by $\psi_m^{(0)*}$ for $m \neq n$ and integrating over x yields

$$c_{mn} \left(\lambda_n^{(0)} - \lambda_m^{(0)} \right) = - \int \psi_m^{(0)*} Q^{(1)} \psi_n^{(0)} dx \Big/ \int |\psi_m^{(0)}|^2 \, dx \qquad \text{(F8)}$$

Using Eq.(F7) we will show that (with certain assumptions) inequality (21) implies $\lambda_n^{(1)} \ll \lambda_n^{(0)}$ for small n. Similar arguments (which we will not go through), using Eq.(F8), would show that inequality (21) also implies $c_{mn} \ll 1$ for $m \neq n$, for small n. Thus we will show that if inequality (21) is satisfied, Eqs.(20a) and (20b) are good approximations to the exact normal mode potentials and frequencies, for small n.

We wish to estimate an upper bound for the expression $\int \psi_n^{(0)*} Q^{(1)} \psi_n^{(0)} dx$ appearing in Eq.(F7). We will consider n = 0; the same arguments apply for any $n \sim 1$.

From Eq.(19a)

$$\psi_0^{(0)}(x) = \exp\left(-\beta x^2/2\right)$$

We note that $\psi_0^{(0)}(x)$ is negligibly small for $x \gg (\mathrm{Re}\,\beta)^{-1/2}$, and that its fourier transform is negligibly small for $k_x \gg \beta^{1/2}$. This suggests that $\int \psi_0^{(0)*} Q^{(1)} \psi_0^{(0)} dx$ will depend on $Q^{(1)}(x, k_x, \omega)$ only for $|x| \lesssim (\mathrm{Re}\,\beta)^{-1/2}$ and $|k_x| \lesssim \beta^{1/2}$, and not for much larger values of x and k_x. Also, if $\lambda_0^{(1)} \ll \lambda_0^{(0)}$, then $\Delta\omega_0^{(1)} \ll \Delta\omega_0^{(0)}$, and we need only consider values of

ω satisfying $|\omega - \omega_L| \lesssim \Delta\omega_0^{(0)}$. Then we expect

$$\int \psi_0^{(0)*} Q^{(1)} \psi_0^{(0)} dx \lesssim Q_{max}^{(1)} \int \psi_0^{(0)*} \psi_0^{(0)} dx \qquad (F9)$$

where $Q_{max}^{(1)}$ is the maximum absolute value attained by $Q^{(1)}(x, k_x, \omega)$ for any $|x| \lesssim (\mathrm{Re}\,\beta)^{-1/2}$, $|k_x| \lesssim \beta^{1/2}$ and $|\omega - \omega_L| \lesssim \Delta\omega_0^{(0)}$. It is possible to imagine operators $Q^{(1)}$ for which Eq.(F9) would not be satisfied, e.g. $Q^{(1)}$ could be exponentially large at $|x| >> (\mathrm{Re}\,\beta)^{-1/2}$ for $|k_x| >> \beta^{1/2}$. However, we conjecture that Eq.(F9) is true for any physically reasonable choice of $Q^{(1)}$.

Then Eqs.(F7) and (F9) imply $\lambda_0^{(1)} \lesssim Q_{max}^{(1)}$. Using the definition following Eq.(F2),

$$Q^{(1)}(x,k_x,\omega) = \sum_{\substack{p,q,r \geq 0 \\ p+q+2r \geq 3}} \frac{\partial^{p+q+r} Q(0,0,\omega_L)}{\partial x^p \, \partial k_x^q \, \partial \omega^r} \frac{x^p k_x^q (\omega-\omega_L)^r}{p!\; q!\; r!}$$

Thus

$$\lambda_0^{(1)} \lesssim Q_{max}^{(1)} \lesssim \sum_{\substack{p,q,r \geq 0 \\ p+q+2r \geq 3}} \left| \frac{\partial^{p+q+r} Q(0,0,\omega_L)}{\partial x^p \, \partial k_x^q \, \partial \omega^r} \right| \frac{|\mathrm{Re}\beta|^{-p/2} |\beta|^{q/2} |\Delta\omega_0^{(0)}|^r}{p!\; q!\; r!} \qquad (F10)$$

We need an upper bound on $\left| \frac{\partial^{p+q+r} Q(0,0,\omega_L)}{\partial x^p \, \partial k_x^q \, \partial \omega^r} \right|$. We can estimate this by noting that Q is the sum of several terms

$$Q = \sum_j Q_j$$

where each term Q_j is due to a different physical effect, e.g. the vacuum term, the electron convection term, the ion Landau damping term, etc. Each Q_j is assumed to be nearly independent of x for $|x| \lesssim L_j$; thus L_j is an effective scale length for the plasma properties associated with Q_j (e.g.

density or density gradient of one species). $Q(0,0,\omega_L) = 0$ because all of the Q_j cancel out at $x = 0$, $k_x = 0$, $\omega = \omega_L$. In the vicinity of this point, at least two of the Q_j must be much greater than Q. Similarly, for the dominant terms Q_j, $\partial Q_j/\partial x$ and $\partial Q_j/\partial k_x$ will be much greater than $\partial Q/\partial x$ and $\partial Q/\partial k_x$. We will assume, however, that $\partial Q_j/\partial\omega \sim \partial Q/\partial\omega$, $\partial^2 Q_j/\partial k_x^2 \sim \partial^2 Q/\partial k_x^2$ and $\partial^2 Q_j/\partial x^2 \sim \partial^2 Q/\partial x^2$ for at least some of the dominant terms Q_j.

As may be seen from Eq.(12), k_x appears in D only as part of the expression $(k_x^2 + k_y^2)^{1/2}$, hence k_x appears in Q only as part of $[(k_x+k_L)^2 + k_y^2]^{1/2}$. So we expect each term Q_j to be nearly independent of k_x for $k_x << k \equiv (k_L^2 + k_y^2)^{1/2}$. Finally, we assume that all dominant terms Q_j are nearly independent of ω for $|\omega - \omega_L| \lesssim \Omega$, and that $\partial Q/\partial\omega \sim Q_j/\Omega$. [If, on the contrary, $\partial Q/\partial\omega << Q_j/\Omega$, we can still use the local method if we re-define $\lambda_n^{(0)}$ as $\Delta\omega^{(0)}\ \partial Q^{(0)}/\partial\omega + (1/2)\ (\Delta\omega_n^{(0)})^2\ \partial^2 Q^{(0)}/\partial\omega^2$, and $\lambda_n^{(1)}$ as $\Delta\omega_n^{(1)}\ \partial Q^{(0)}/\partial\omega + \Delta\omega_n^{(1)}\ \Delta\omega_n^{(0)}\ \partial^2 Q^{(0)}/\partial\omega^2$. Then Eqs.(19a), (19b) and (20a) will still be valid if $\lambda_n^{(1)} << \lambda_n^{(0)}$, but Eq.(20b) must be replaced by

$$\omega_n = \omega_L - (\partial D/\partial\omega)(\partial^2 D/\partial\omega^2)^{-1} \pm [(\partial D/\partial\omega)^2(\partial^2 D/\partial\omega^2)^{-2} - (2n\eta+\beta)(\partial^2 D/\partial k_x^2)(\partial^2 D/\partial\omega^2)^{-1}]^{1/2} \qquad \text{(F11)}$$

In other words, for each integer n there will be two normal modes, with nearly identical potentials given by Eq.(20a), and with frequencies given by the + and - choices of Eq.(F11). This situation occurs near the values of k_y and ω where two modes interact, e.g. a drift wave and a Bernstein wave.]

If $Q_j(x, k_x, \omega)$ is analytic in the range $|x| << L_j$, $k_x << k$, $|\omega - \omega_L| << \Omega$, we can write

$$Q_j(x, k_x, \omega) - Q_j(0,0,\omega_L) = \sum_{\substack{p,q,r\geq 0 \\ p+q+r\geq 1}} \frac{\partial^{p+q+r} Q_j(0,0,\omega_L)}{\partial x^p \partial k_x^q\, \omega \partial^r}\ \frac{x^p k_x^q (\omega-\omega_L)^r}{p!\ q!\ r!} \qquad \text{(F12)}$$

Since

$$|Q_j(x, k_x, \omega) - Q_j(0, 0, \omega_L)| \leq Q_j(0, 0, \omega_L)$$

for all $|x| \leq L_j$, $|k_x| \leq k$, $|\omega - \omega_L| \leq \Omega$, it follows from Eq.(F12) that

$$\left| \frac{\partial^{p+q+r} Q_j(0,0,\omega_L)}{\partial x^p \, \partial k_x^q \, \partial \omega^r} \right| \leq p!\, q!\, r!\, L_j^{-p} k^{-q} \Omega^{-r} \, |Q_j(0,0,\omega_L)|$$

If there are only a few dominant terms Q_j,

$$\left| \frac{\partial^{p+q+r} Q(0,0,\omega_L)}{\partial x^p \, \partial k_x^q \, \partial \omega^r} \right| \leq \left| \frac{\partial^{p+q+r} Q_j(0,0,\omega_L)}{\partial x^p \, \partial k_x^q \, \partial \omega^r} \right|$$

it follows that

$$\left| \frac{\partial^{p+q+r} Q(0,0,\omega_L)}{\partial x^p \, \partial k_x^q \, \partial \omega^r} \right| \leq p!\, q!\, r!\, L^{-p} k^{-q} \Omega^{-r} \, |Q_j(0,0,\omega_L)| \qquad \text{(F13)}$$

where L is the smallest L_j for any of the dominant terms Q_j, and Q_j is a typical dominant term. Putting Eq.(F13) into Eq.(F10) yields

$$\lambda_0^{(1)} \leq \sum_{\substack{p,q,r \geq 0 \\ p+q+2r \geq 3}} (L\,|\mathrm{Re}\beta|^{1/2})^{-p} \, (k\,|\beta|^{-1/2})^{-q} \, (\Omega |\Delta\omega_0^{(0)}|^{-1})^{-r} Q_j(0,0,\omega_L)$$

We want to compare $\lambda_0^{(1)}$ to $\lambda_0^{(0)}$ where

$$\lambda_0^{(0)} = \Delta\omega_0^{(0)} \, \partial Q/\partial\omega \sim \Delta\omega_0^{(0)} \, Q_j \, \Omega^{-1}$$

So $\lambda_0^{(1)} \ll \lambda_0^{(0)}$ if

$$L \ll |\mathrm{Re}\beta|^{-1/2} \qquad \text{(F14)}$$

and

$$k \ll |\beta|^{1/2} \qquad \text{(F15)}$$

and

$$\Delta\omega_0^{(0)} \ll \Omega \qquad \text{(F16)}$$

We will assume $\mathrm{Re}\beta \sim \beta$ (this may not be true for plasmas in which magnetic shear is important) so Eq.(F14) becomes

$$L \ll |\beta|^{-1/2} \tag{F17}$$

If $B^2 - 4AC \sim -4AC$ in Eq.(18) (This is certainly true for all of the modes found in Sec. V) then, from the definition of β after Eq.(19),

$$\beta \sim \left(\frac{\partial^2 Q}{\partial x^2} \Big/ \frac{\partial^2 Q}{\partial k_x^2}\right)^{1/2}$$
$$\sim \left(\frac{2Q_j}{L^2} \Big/ \frac{2Q_j}{k^2}\right)^{1/2} \sim kL^{-1} \tag{F18}$$

From Eq.(20b),

$$\Delta\omega_0^{(0)} = -(\beta/2)(\partial Q/\partial\omega)^{-1}(\partial^2 Q/\partial k_x^2)$$
$$\sim (k/2L)(Q_j/\Omega)^{-1}(2Q_j/k^2)$$
$$= \Omega(kL)^{-1} \tag{F19}$$

Using Eq.(F18) and F(18), Eq.(F15), (F16), and (F17) all reduce to

$$Lk \gg 1$$
$$\text{i.e. } L(k_L^2 + k_y^2)^{1/2} \gg 1 \tag{F20}$$

which is inequality (21).

Appendix G

Sketch of Proof that Normal Modes are in the Form of Eq. (32):

Floquet's theorem[24] tells us that a differential equation of order m

$$D(x, -i\frac{\partial}{\partial x}, \omega)\ \phi\ (x) = 0 \qquad \text{(G1)}$$

where

$$D(x, k_x, \omega) = P_m(x,\omega)\ k_x^{\,m} + P_{m-1}(x,\omega)\ k_x^{\,m-1} + \ldots + P_o(x,\omega) \qquad \text{(G2)}$$

with coefficients $P_i(x,\omega)$ periodic in x with period $2\pi\, k_o^{-1}$

$$P_i(x, \omega) = P_i(x + 2\pi k_o^{-1}, \omega) \text{ for } i = 1,2,\ldots m$$

has m independent solutions $\phi_j(x)$ which may be chosen so that

$$\phi_j(x + 2\pi k_o^{-1}) = \exp\,[2\pi i K_j(\omega) k_o^{-1}]\ \phi_j(x)$$

$$\text{for } j = 1,2,\ldots m \qquad \text{(G3)}$$

where each $K_j(\omega)$ depends on the functions $P_o(x,\omega)$, $P_i(x,\omega),\ldots P_m(x,\omega)$. Then the most general solution to Eq. (G1) is

$$\phi(x) = \sum_{j=1}^{m} A_j\ \phi_j(x) = \sum_{j=1}^{m} A_j\ \exp(iK_j x)\ \sum_{p=-\infty}^{\infty} \phi_{p,j}\ \exp(ipk_o x)$$

$$\text{where } \sum_p |\phi_{p,j}|^2 = 1$$

If we take as boundary conditions the requirement that $\phi(x)$ remain finite as $x \to \pm\infty$, then

$$A_j = 0 \quad \text{or} \quad \mathrm{Im}K_j(\omega) = 0$$

for every j, $1 \le j \le m$. If $\mathrm{Im}K_j(\omega) = 0$ for only one value of j, then the only $\phi(x)$ which satisfies the boundary conditions is

$$\phi(x) = \exp(iK_j x) \sum_{p=-\infty}^{\infty} \phi_{p,j}\ \exp(ip\kappa_o x) \qquad \text{(G4)}$$

If $\mathrm{Im}K_j(\omega) = 0$ for more than one value of j, then there is a degeneracy at this value of ω, but we are still free to choose the normal modes to be of the form given by Eq.(G4).

In fact $D(x, k_x, \omega)$ is not a polynomial function of k_x, as in Eq.(G2), but is a more complicated analytic function of k_x, given by Eq.(12), so Eq.(G1) is an integral equation rather than a differential equation. Is it possible that there is a normal mode solution $\phi(x)$ of Eq.(G1) [with $D(x, k_x, \omega)$ an arbitrary analytic function of k_x, still periodic in x] which could not be written in the form of Eq.(G4)? Suppose that such a mode $\phi(x)$ does exist. If $\phi(x)$ is differentiable, then there is some wave number k such that $\tilde{\phi}(k_x)$ is negligibly small for $|k_x| \gtrsim k$. There is also some integer m such that $D(x,k_x, \omega)$ can be approximated very well by Eq.(G2) for $|k_x| \lesssim k$. Then $\phi(x)$ would be very close to a normal mode of Eq.(G1) with $D(x,k_x, \omega)$ given by Eq.(G2), which would contradict our supposition that $\phi(x)$ cannot be written in the form of Eq.(G4).

APPENDIX H

Derivation of Eq. (35):

We start with the definitions of $G_{0,s}$ and $G_{\pm 1,s}$ after Eq. (25), and use

$$\exp(-k^2a_s^2)\, I_\ell(k^2a_s^2) \approx (2\pi)^{-1/2}(ka_s)^{-1}\exp(-\ell^2/2k^2a_s^2) \tag{H1}$$

(valid when $k^2a_s^2 \gg \ell \gg 1$)[19] and replace the sum over ℓ by an integral [valid when Im $\omega/\omega_{ci} \gtrsim 1$ so that $(\omega/\omega_{cs}-\ell)^{-1}$ varies slowly with ℓ, and $k_0 \lesssim k$ so that $\exp(i\ell\alpha_{\pm 1})$ varies slowly with ℓ; these conditions also ensure that $\ell \gg 1$ for most terms that contribute significantly, since typically $|\ell| \approx |\omega|/\omega_{cs}$].

We first consider $G_{0,s}$. With the above approximations,

$$G_{0,s}(k_x,\omega) = (2\pi)^{-1/2}(\omega_{ps}^2/v_s^2)(ka_s)^{-1} \int_{-\infty}^{\infty} d\ell\ \ell \exp(-\ell^2/2k^2a_s^2)$$
$$(\omega/\omega_{cs}-\ell)^{-1} \tag{H2}$$

We do the integration in Eq. (H2):

$$\int_{-\infty}^{\infty} d\ell\ \ell \exp(-\ell^2/2k^2a_s^2)(\omega/\omega_{cs}-\ell)^{-1} =$$
$$\int_{-\infty}^{\infty} d\ell(\ell-\omega/\omega_{cs})\ \exp(-\ell^2/2k^2a_s^2)(\omega/\omega_{cs}-\ell)^{-1}$$
$$+ (\omega/\omega_{cs}) \int_{-\infty}^{\infty} d\ell\ \exp(-\ell^2/2k^2a_s^2)(\omega/\omega_{cs}-\ell)^{-1}$$
$$= -(2\pi)^{1/2}\, ka_s\, -(\omega/\omega_{cs})\pi^{1/2}\, Z(\omega/\sqrt{2}k\, v_s) \tag{H3}$$

Eqs. (H2) and (H3) yield the first part of Eq. (35):

$$G_{0,s}(k_s,\omega) = -(\omega_{ps}^2/v_s^2)[1 + \zeta_0 Z(\zeta_0)] \tag{H4}$$

where $\zeta_0 \equiv \omega/\sqrt{2}\, k\, v_s$

To calculate $G_{\pm1,s}$, we note that

$$\exp[-(k^2 + k_{\pm1}^2)a_s^2/2] = \exp[-(k - k_{\pm1})^2 a_s^2/2]$$

$$\exp(-k\, k_{\pm1} a_s^2) \tag{H5}$$

$$(k - k_{\pm1})^2 \approx k_x^2 k_0^2/k\, k_{\pm1} + \mathcal{O}(k_0^3/k) \tag{H6}$$

$$\alpha_{\pm1} \approx \mp\, k_y k_0/k\, k_{\pm1} + \mathcal{O}(k_0^3/k^3) \tag{H7}$$

where Eqs. (H6) and (H7) are valid when $k_0 \ll k$, and may be derived from the definitions of $k_{\pm1}$ and $\alpha_{\pm1}$ after Eq. (25). Starting with the definition of $G_{\pm1,s}$ after Eq. (25), and using Eqs. (H1), (H5), (H6) and (H7), we obtain

$$G_{\pm1,s}(k_x,\omega) = (\omega_{ps}^2 \Delta_s/2v_s^2)(2\pi k\, k_{\pm1} a_s^2)^{-1/2}$$

$$\int_{-\infty}^{\infty} d\ell\, \exp(\mp i\ell k_y k_0/k\, k_{\pm1})\, \exp(-k_x^2 k_0^2 a_s^2/2k\, k_{\pm1})$$

$$\exp(-\ell^2/2k\, k_{\pm1} a_s^2)(\ell \pm i k_y k_0 a_s^2)\, (\omega/\omega_{cs} - \ell)^{-1} \tag{H8}$$

Rearranging terms, Eq. (H8) becomes

$$G_{\pm1,s} = (\omega_{ps}^2 \Delta_s/2v_s^2)(2\pi k\, k_{\pm1} a_s^2)^{-1/2} \exp(-k_0^2 a_s^2 k/2k_{\pm1})$$

$$\int_{-\infty}^{\infty} d\ell\, \exp(-\ell^2/2k\, k_{\pm1} a_s^2 \mp i\ell k_y k_0/k\, k_{\pm1} + k_0^2 k_y^2 a_s^2/2k\, k_{\pm1})$$

$$(\ell \pm i k_y k_0 a_s^2)\, (\omega/\omega_{cs} - \ell)^{-1} \tag{H9}$$

Changing the variable of integration from ℓ to $\ell' \equiv \ell \pm ik_y k_0 a_s^2$, the integral in Eq. (H9) becomes

$$\int_{-\infty}^{\infty} d\ell' \exp(-\ell'^2/2k\, k_{\pm 1} a_s^2)\ell'(\omega/\omega_{cs} - \ell' \pm ik_y k_0 a_s^2)^{-1}$$

This integral is of the same form as the integral in Eq. (H2), but with ω/ω_{cs} in Eq. (H2) replaced by $\omega/\omega_{cs} \pm ik_y k_0 a_s^2$, and it can be solved in the same way. To obtain the second part of Eq. (35), we must further replace $k_{\pm 1}$ everywhere by k, which is justified if $k_0 \ll k$. Note that we could not have replaced $k_{\pm 1}$ by k before writing down Eq. (H8), or we would have lost the factor $\exp(-k_x^2 k_0^2 a_s^2/2k\, k_{\pm 1})$ appearing in Eq. (H8).

It might be argued that it was not justified to neglect the higher order terms in Eqs. (H6) and (H7), since these terms, although small compared to the leading terms, may not be small compared to unity when they appear in the exponentials in Eq. (H8); hence they might significantly change Eq. (H8). However, a careful analysis including these higher order terms shows that they are only important when $G_{\pm 1,s}$ is exponentially small, in which case $G_{\pm 1,s}$ would not be important in the dispersion relation anyway.

APPENDIX I

Outlines of the main program ROOTS [which finds the normal modes $\phi(x)$ and frequencies ω, using the nonlocal method described in Sec. IIIB] and of the three large subroutines FOLLOW [which follows a given branch of $\omega(k_y)$ out from small k_y to large k_y], DISP (which calculates the dispersion function det $A_{p,p'}$ for a given ω and k_y) and MRAF [which finds the zeroes of det $A_{p,p'}(\omega)$ for a given k_y] are shown in the flow charts. These charts are intended only as rough guides to the programs. Details of the I/O and details of how various decisions are made (e.g. how FOLLOW decides when two roots are a complex pair or a fork; how FOLLOW decides when to increase or decrease the increment in k_{y_i}; how MRAF decides when $\phi_0 \approx 0$) are not shown. Also not shown are certain modes of operation which were not found to be very useful, e.g. a mode of operation where the frequency, rate of spreading and acceleration were found for wave packets localized at different values of x. For such details the listing may be consulted.

The subroutine MRAF evolved from a root-finding routine published by Rodman[16] and revised by Whitley[17], using a method developed by Muller and Traub[18], a second order iterative method analogous to the secant method. The published version was further revised by one of us (A.B.L.) and used as a subroutine in the first version of ROOTS (written by C.K.B., D.F., and A.B.L.). Several additions and changes to ROOTS and its subroutines were made by M.J.G., including:

1) Extension of the dispersion function routine DISP and its subroutines to include Maxwellian and loss cone distributions [only ring distributions, $g_s(v_\perp, v_\parallel, x) \propto \delta(v_\perp - v_0)$, were used in the first version], and to include many species (e.g. deuterium ring and tritium Maxwellian).

2) Use of possible symmetry of dispersion function around imaginary axis [i.e. $\det A_{p,p'}(\omega) = \det A_{p,p'}(-\omega^*)$] in finding roots (symmetry around real axis was used in published version of MRAF[16]).

4) Optional boundary to region of complex plane in which MRAF can search for roots. When the root-search goes beyond this boundary, the search ends (say at point x_i) and the function is divided by $(x-x_i)$. Eventually the boundary is surrounded by poles, and the root-search tends to stay away from the boundary and find all the roots inside.

5) Dividing up of complex plane into several regions, each searched separately by MRAF. This greatly reduces the number of roots missed by MRAF, if a large area of the complex plane, with a large number of roots, is to be searched.

6) Modifications in criteria for convergence (i.e. for having found a root) in MRAF, necessary when there are many roots close together.

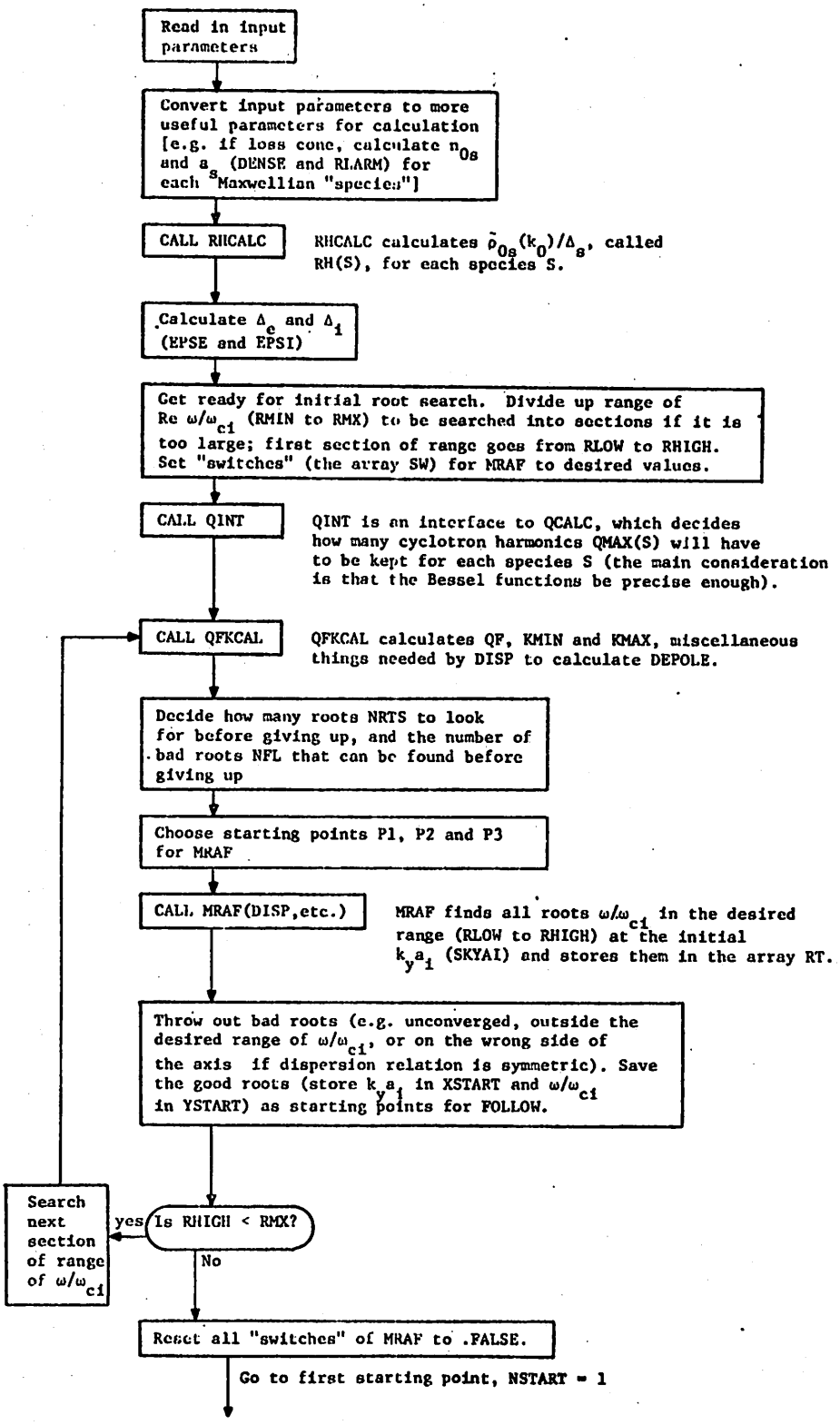
7) Use of eigenfunctions (ϕ_p) as a criterion for convergence in MRAF.

8) Option for using straight line ion orbits in calculating the dispersion function.

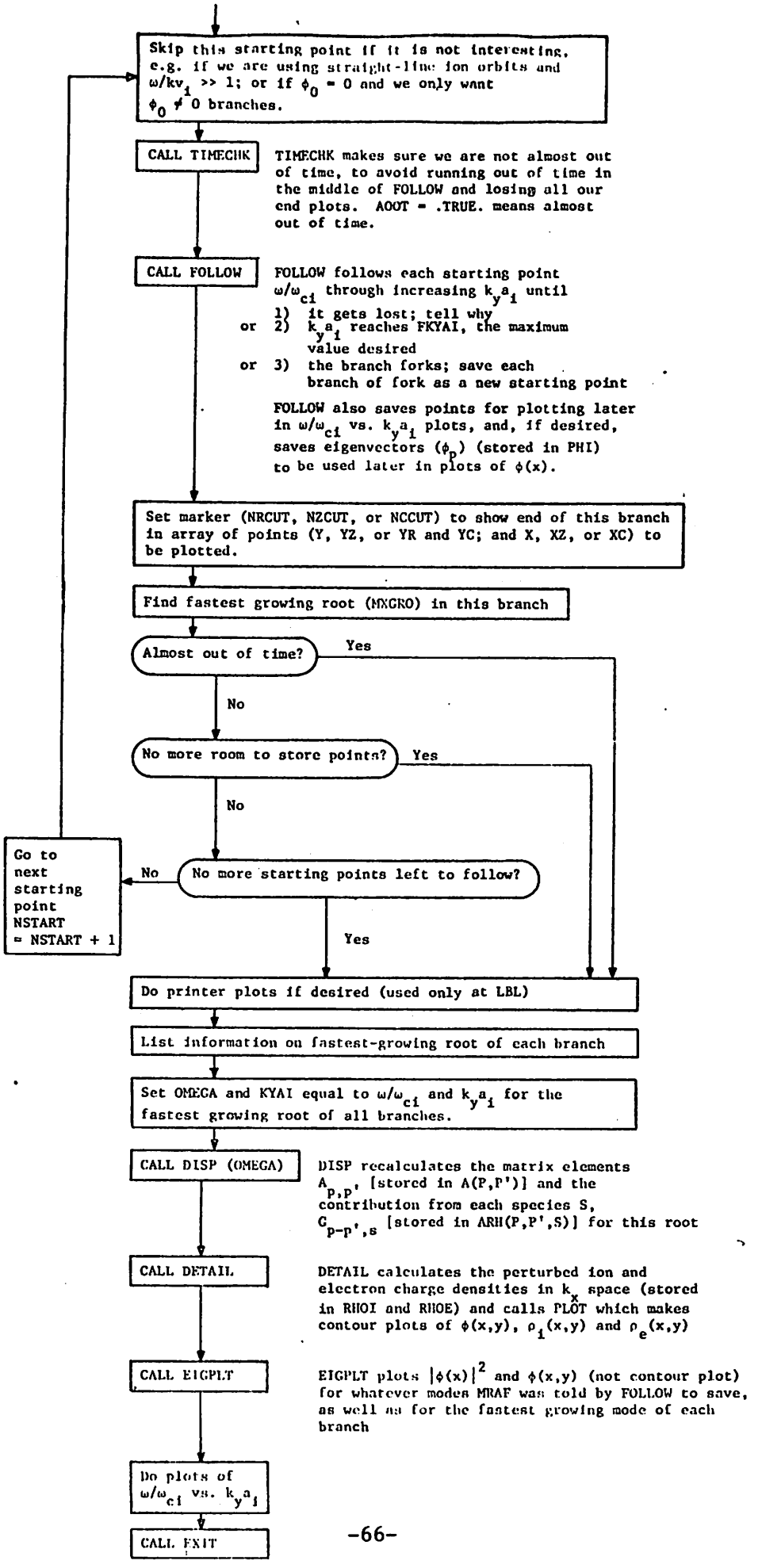
9) Implementation of subroutine FOLLOW to follow each branch $\omega(k_y)$ through increasing k_y. Use of eigenfunctions (ϕ_p) to sort out different branches.

10) Option which allowed ROOTS to find $\omega(k_y)$ for an infinite medium with constant (1/n) dn/dx, using the local approximation [i.e., $D(x=0,\omega,k_x=0) = 0$] rather than finding normal modes.

FLOW CHART OF ROOTS



(see previous page)

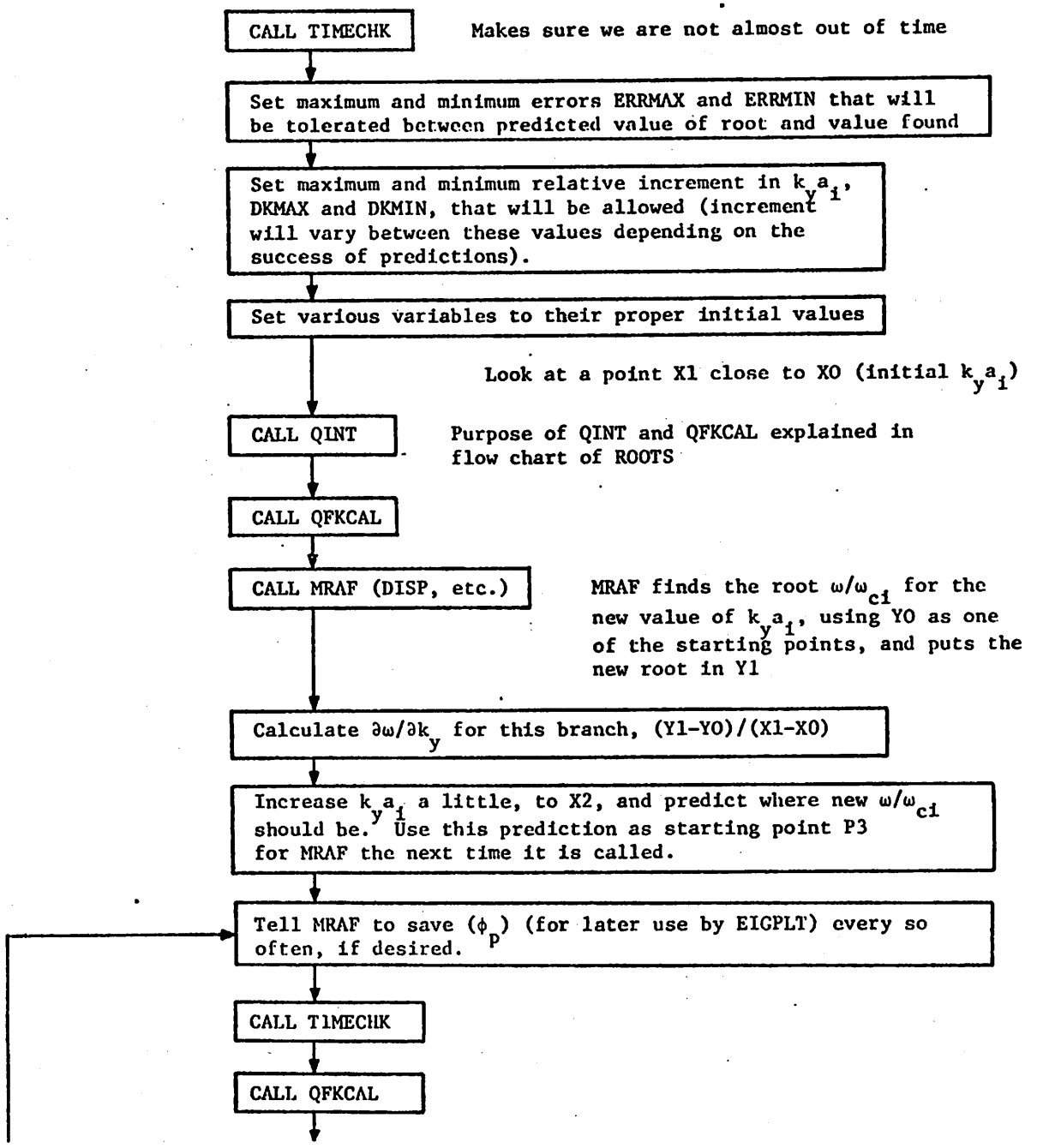


FLOW CHART OF FOLLOW (XO, YO, PHIZP)

XO is the $k_y a_i$ of the starting point

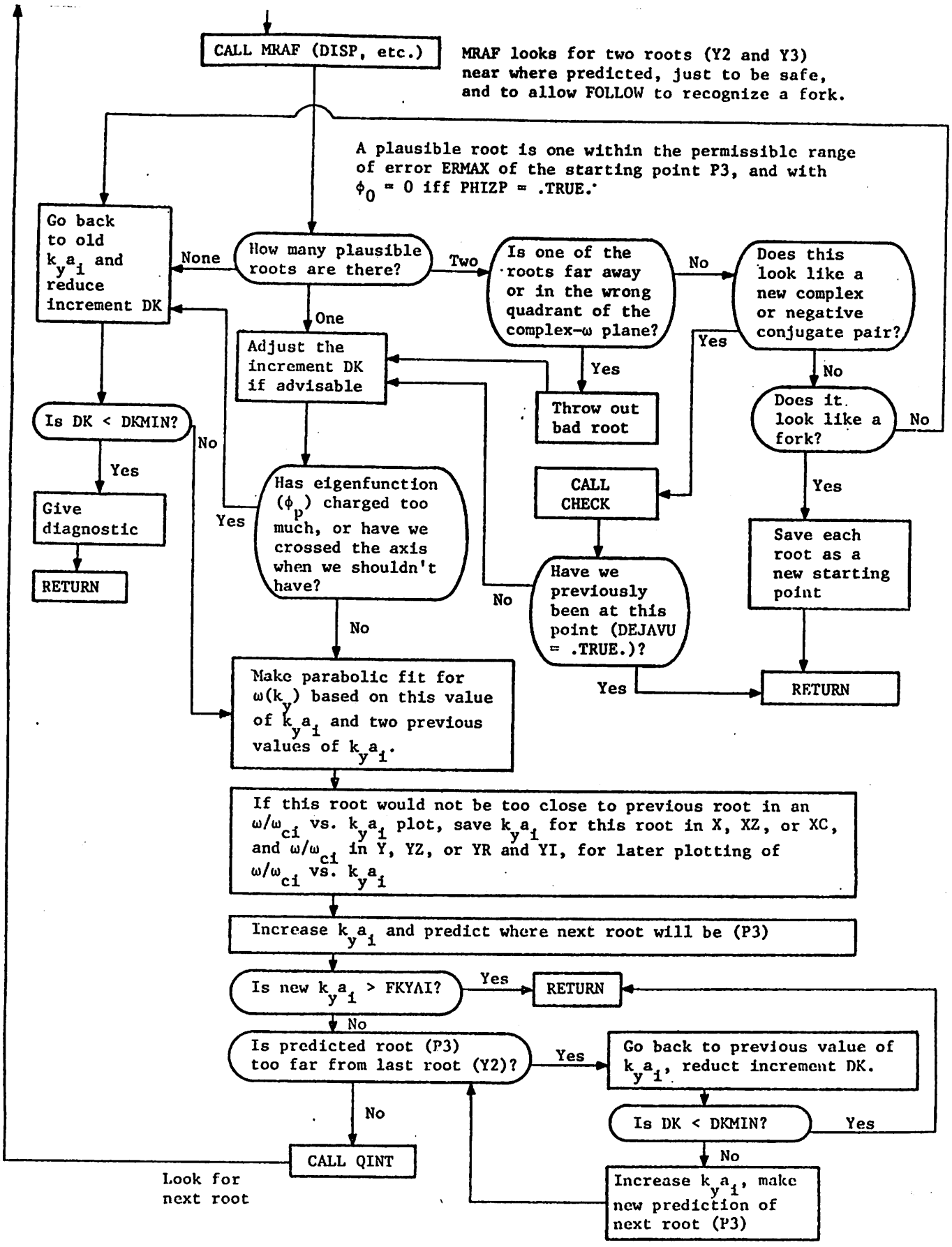
YO is the ω/ω_{ci} of the starting point

PHIZP = .TRUE. iff $\phi_0 = 0$ for this branch (useful, since this is a property which does not change for a given branch as $k_y a_i$ changes)

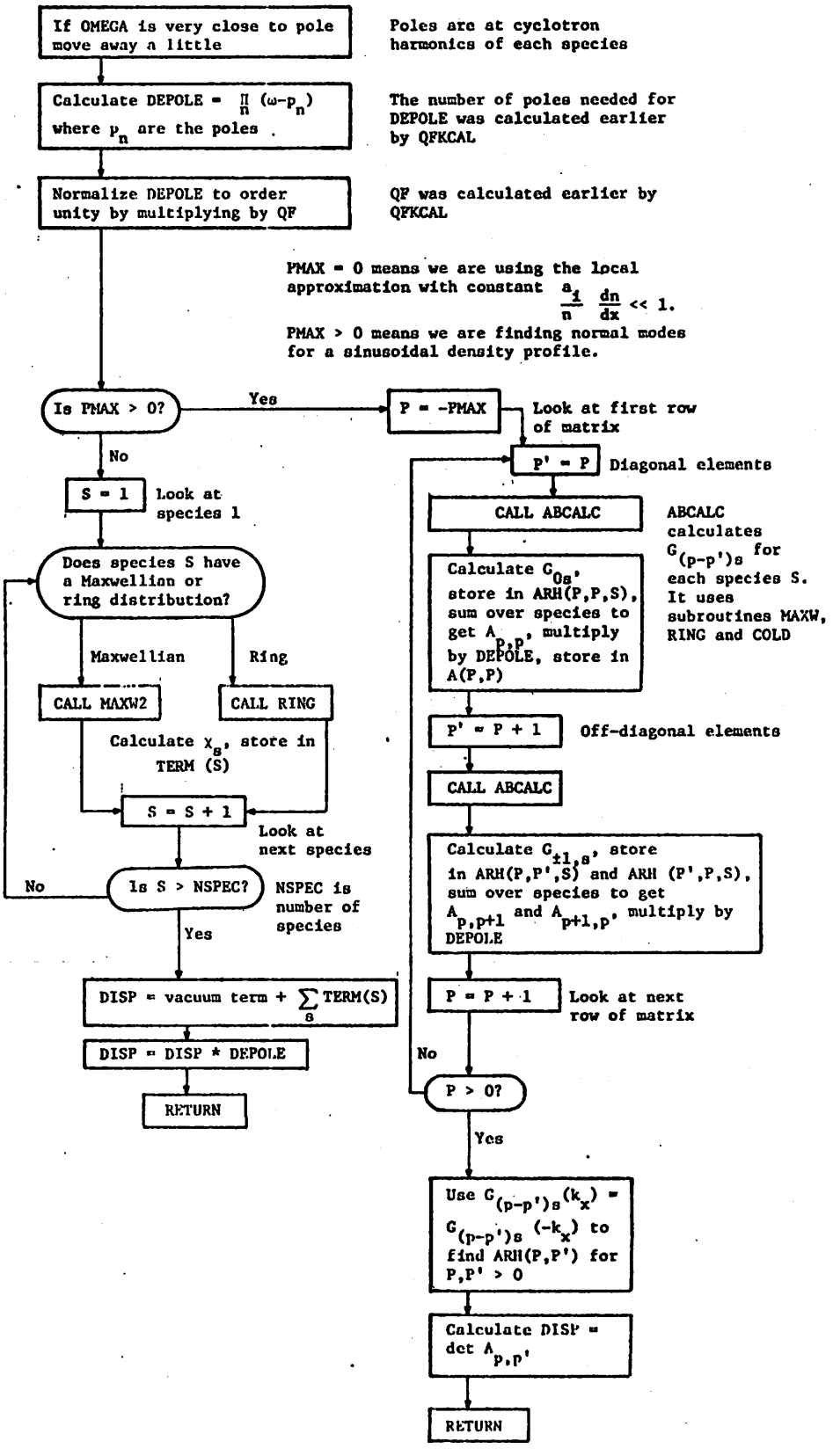


(see on next page)

(see previous page)

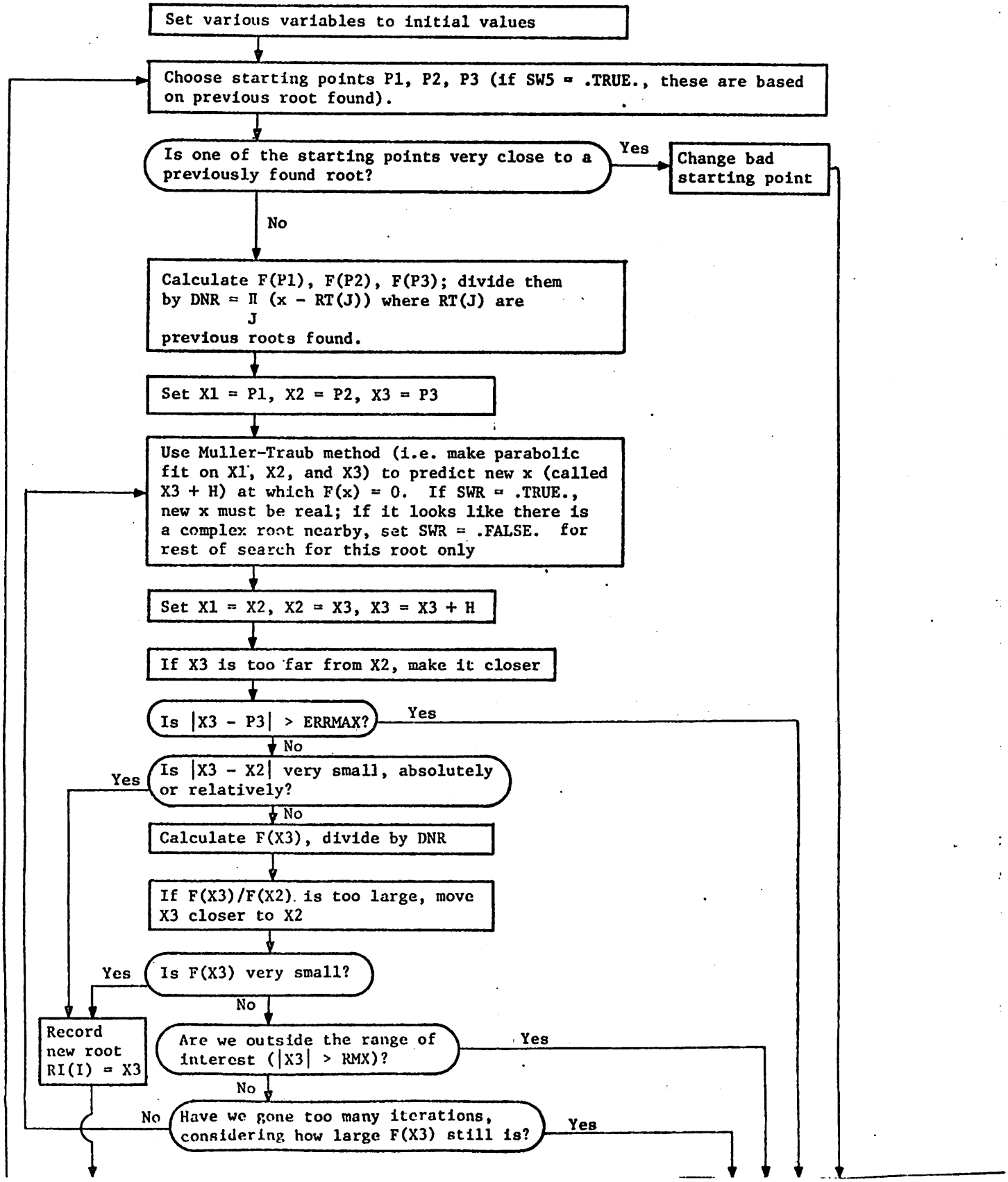


FLOW CHART OF DISP (OMEGA)



FLOW CHART OF MRAF (F,etc.)

F is the function which MRAF is finding the zeroes of

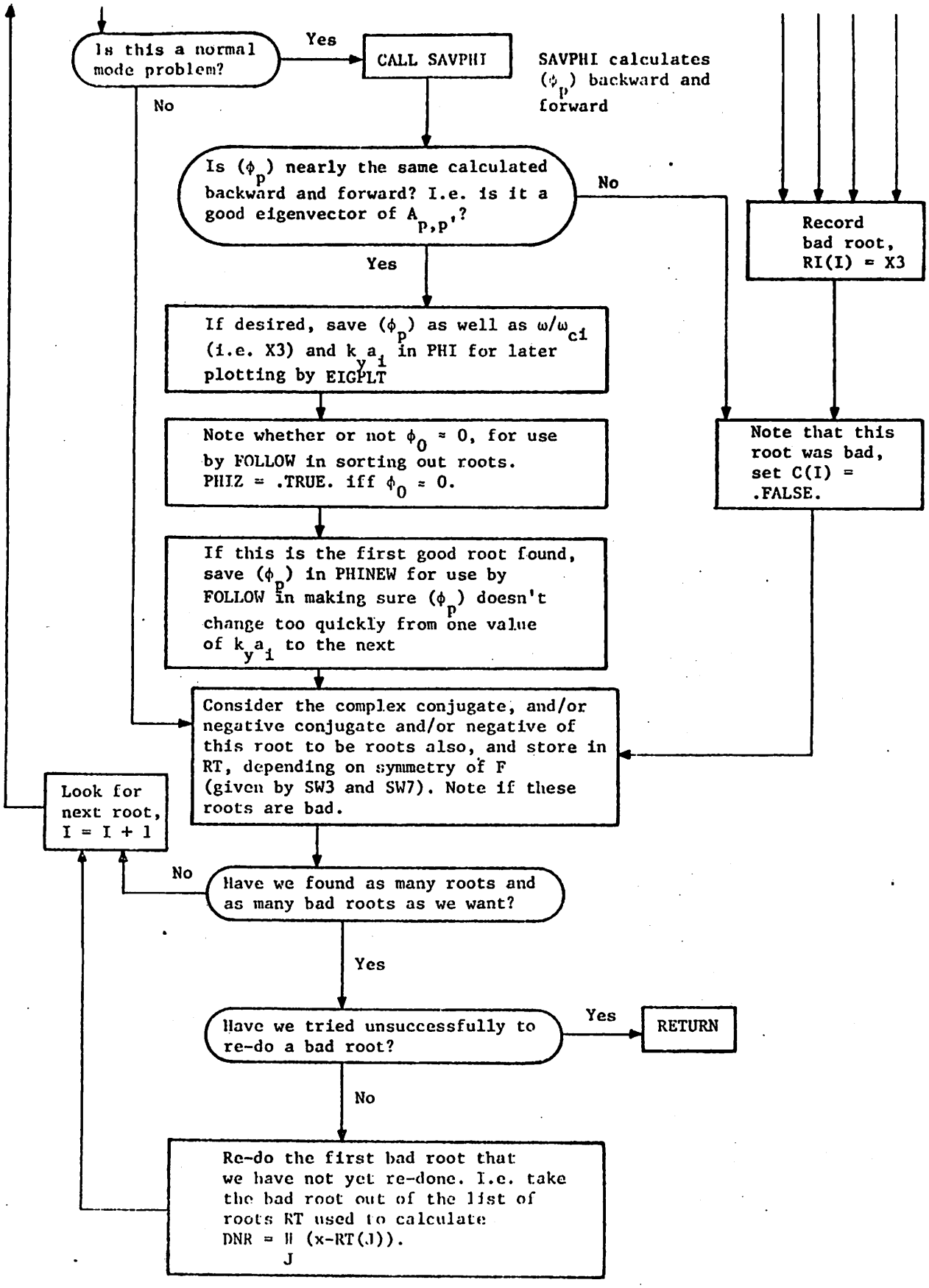


(see next page)

(see next page)

(see previous page)

(see previous page)



Listing of ROOTS and Subroutines

The version of ROOTS listed here runs on the A-machine (a CDC 7600) of the National Magnetic Fusion Energy Computer Center at Lawrence Livermore Laboratory, using the NEW1 compiler (which differs from the better known CHATR compiler in that it does complex division correctly), and the libraries ORDERLIB, CF76LIB, and TV80LIB. Features differing from standard Fortran include I/O (especially graphics routines), tests to see if the time limit has nearly been reached (OOTIM) and treatment of logical variables.

Although the code originated at Livermore, it spent a number of its formative years at Lawrence Berkeley Laboratory (growing from 600 to nearly 3000 cards) and it still contains a number of archaic features from its Berkeley days (e. g. the routines SGNL and WARN).

It is hoped that this code, which should be useful for a wide range of problems, will be made available to the users of the A-machine in the not too distant future, after it has been cleaned up and documented.

```
C-170A    BOX B05     13.28.56    06/17A      1976
000001          PROGRAM ROOTS(INPUT,HSP,OUTPUT,TAPE10=OUTPUT)
000002    C
000003    C     THE FOLLOWING   CHANGES SHOULD BE MADE IN *ROOTS* AND THE SUBROUTINES
000004    C     EVENTUALLY, BUT I HAVE NOT HAD TIME TO MAKE THEM YET, AND PROBABLY WILL NOT
000005    C     FOR QUITE A WHILE -MJG.
000006    C      1)  *RING2*, THE SUBROUTINE USED FOR RING DISTRIBUTIONS WITH THE
000007    C          LOCAL APPROXIMATION, IS NOT WORKING CORRECTLY, AND SHOULD BE CORRECTED
000008    C      2) DIFFUSION HAS NOT YET BEEN IMPLEMENTED WITH THE NON-LOCAL (PMAX.GT.0)
000009    C          DISPERSION RELATIONS, ONLY WITH THE LOCAL (PMAX=0) DISPERSION RELATIONS
000010    C      3)  THE UNMAGNETIZED PARTICLES OPTION (UNMAG=.TRUE.) HAS NOT YET BEEN
000011    C          IMPLEMENTED FOR RING DISTRIBUTIONS, ONLY FOR MAXWELLIAN AND LOSS CONE
000012    C          DISTRIBUTIONS.
000013    C      4)  SO FAR *MRAF* HAS WORKED QUITE WELL IN FINDING THE STARTING POINTS
000014    C          BUT IT MIGHT NOT BE SO EFFICIENT  (I.E. IT MIGHT MISS ROOTS) IN
000015    C          FARTHER REACHES OF PARAMETER SPACE. IN THIS CASE IT MIGHT BE
000016    C          NECESSARY TO USE A NYQUIST TECHNIQUE, SUCH AS CALLEN USED IN HIS
000017    C          THESIS, TO DETERMINE HOW MANY ROOTS THERE ARE AND THEIR APPROXIMATE
000018    C          LOCATIONS, RATHER THAN THE BLIND HUNTING OF *MRAF*.
000019    C             *MRAF* IS PRESENTLY NOT VERY GOOD AT FINDING ROOTS FAR FROM
000020    C          THE REAL AXIS, HENCE SKYAI MUST BE SET LOW, TO BE SURE OF NOT
000021    C          MISSING ANY ROOTS. IF *MRAF* WERE IMPROVED BY THE METHOD SUGGESTED
000022    C          ABOVE, SKYAI COULD BE SET HIGHER, AND CONSIDERABLE TIME SAVED
000023    C          (LIKE 75 PERCENT), SINCE MOST TIME IS NOW SPENT AT LOW VALUES OF KYAI.
000024    C      5) *MRAF* SOMETIMES BLOWS UP WHEN NRTS IS SET TOO HIGH. THIS HAS BEEN
000025    C          PREVENTED BY ARBITRARILY NOT LETTING NRTS BE MORE THAN 100, BUT
000026    C          THE UNDERLYING PROBLEMS SHOULD BE FOUND AND CORRECTED.
000027    C      6)  MAKE *ZEE* MORE EFFICIENT AND MORE ACCURATE. RIGHT NOW IT IS NEITHER
000028    C          FOR ABS(X) BETWEEN 5.0 AND 5.7, AND ROOTS OCCASIONALLY GET LOST
000029    C      7)  AESTHETIC IMPROVEMENTS, E.G. MAKE *MRAF* AND *FOLLOW* A GENERAL
000030    C          ROOT-FINDER AND ROOT-FOLLOWER, NOT LIMITED TO THIS PARTICULAR
000031    C          PROGRAM AND PROBLEM.  SORT OUT THE COMMON BLOCKS SO EACH SUBROUTINE
000032    C          ONLY HAS ACCESS TO WHAT IT NEEDS.  DON T PUT VARIABLES IN COMMON
000033    C          WHEN THEY REALLY SHOULD BE ARGUMENTS OF SUBROUTINES, AND VICE VERSA.
000034    C
000035          EXTERNAL DISP
000036          LOGICAL C(200)
000037          LOGICAL PHIZ(200), ZEH(100)
000038          LOGICAL WARN, AGOT
000039          LOGICAL SW(6), SW7, SW71,SW31,OPT(10)
000040          LOGICAL UNMAG,SINGLX, GOOD,PRDAMP,SNGLX
000041          COMPLEX OMEGA,DISP,ZERO,ZERO,A(20,20),PHI(18,22)
000042          COMPLEX RT(200), YSTART,Y2
000043          COMPLEX P1,P2,P3
000044          COMPLEX H,ROOT,FUNC,FROOT
000045          COMPLEX DFDW,D2FDK2,DFDX,D2WDK2,DWDX
000046          INTEGER RTC
000047          INTEGER P,PMAX,QMAX,EIGV,RMAX,RMAX1
000048          REAL MASS,KOAI,KOSQ,KYAI,KYSQ,MASQ,IQ,KOAE,KXSQ
000049          DIMENSION X(2000),Y(2000),XC(1000),YR(1000),YI(1000),XZ(4000),
000050         .YZ(4000),XPLOT(501),YPLOT(501)
000051          DIMENSION NRCUT(100), NCCUT(100), NZCUT(100),MXGRO(100),SIDE(100)
000052          DIMENSION RH(4), XLR(100), XLC(100)
000053          DIMENSION ABSPHI(20), ANGPHI(20)
000054          COMMON/CIO/ICR,IHSP,IGRAPH
000055          COMMON/CMRAF/SW,RTC,ITC,EP3,FRMAX,XRMAX,H,ROOT,FUNC,FROOT,EIGV,
000056         .SW7,START,RHIGH,NFL,CYAI
000057          COMMON/CMRAF2/ ERRMAX,NRTSU,SW31,SW71
000058          COMMON/CRBESJ/ QMAX(4),NMAX(4)
000059          COMMON/PMX/NPP
000060          COMMON/CNSP/ NSP,NSP1
000061          COMMON/DIS/QF,KYSQ,KOSQ,EPSF(4),     MASS,KYAI,KOAI,MASQ,PMAX,OMPSQ
000062         .,AEAI,KMAX(4),AMASQ(4),EPS(4),KMIN(4)
000063          COMMON/CPARAM/ RMAX,AIL,EPSI,EPSE,NSPECP,DFUI,DFUE,EP,SINGLX,XL(10
000064         .),NXL
000065          COMMON/CRGAM/R,GAM,RE,GAME,R1,RE1,AEAI1
000066          COMMON/CFAST/ DK,FKYAI,    NPTC,YI,YR,XC,                KYD,      YMX,
000067         .YZ,XZ,Y,X,NPT,NPTZ,ZEH,IGV,AGOT,NFOL,FKYAI2,RMIN
000068          COMMON/CFOL/YSTART(100),XSTART(100),NSTART,YEND(100),XEND(100),
000069         .NSTRT1
000070          COMMON/CCHECK/XCRIT(100),YCRIT(100),NCRIT,DEJAVU,Y2,NN,PZCRIT(100)
000071          COMMON/CQN/MINQ
000072          COMMON/CSPEC/ RLARM(4),AMASS(4),CHARG(4),JAY(4),DENSE(4),NSPEC
000073          COMMON/CDFU/ DFU(4),DREL(4)
000074          COMMON/PHIC/ A, NRTSV, PHI
000075          COMMON/FOLEIG/ GOOD(18),IBRANCH(18)
000076          COMMON/CTVY/TVY
000077          COMMON/TVPOOL/XMIN,XMAX,YMIN,YMAX,TVXMIN,TVXMAX,TVYMIN,TVYMAX
000078          COMMON/TVTUNE/LPENON,LPENOFF,ITALICS,IWINK,INTENSE,IRIGHT,IUP
000079          COMMON/TVGUIDE/ TVMODE,TEXTURE,ITV
000080          COMMON/CPHZ/PHIZ
000081          COMMON/CXK/XKO
000082          COMMON/CMAG/UNMAG
000083          COMMON/CKXSQ/ KXSQ
000084          COMMON/CBETA/ BETA
000085          COMMON/CSNGLX/ SNGLX
000086          DATA ICR,IHSP,IGRAPH/2,10,100/
000087          DATA RLARM/1.,3*0./, AMASS/4*1./, CHARG/2*1.,2*(-1.)/, JAY/4*0/,
000088         .DENSE/1.,0.,1.,0./
000089          DATA AMASQ/4*1./
000090          DATA NR,NC,NZ/3*1/
000091          DATA KMIN/4*1/
```

```
000092            DATA DFU/4*0./
000093            DATA TEXTURE/0./
000094            DATA PRDAMP/.TRUE./
000095            CALL DROPFILE
000096            ELAPSED = 0.
000097            CALL TIMECHK(ADOT,0.,ELAPSED)
000098            CALL KEEP80(10HROOTPLOTAA)
000099            KXSQ = 0.
000100            CALL FRAME(1)
000101            NRCUT(1) = 0
000102            NZCUT(1) = 0
000103            NCCUT(1) = 0
000104            PI = 3.14159265359
000105      C     IF WE WISH TO USE THE LOCAL APPROX. AT A SINGLE VALUE OF X/L, THEN
000106      C     WE SET PMAX = 0 AND SINGLX = .TRUE. THEN THE DENSITY GRADIENT IS
000107      C     GIVEN BY EPS(I) OR EP, WHILE AIL, EPSI, NXL, AND UNMAG ARE IGNORED.
000108      C     IF SINGLX = .FALSE. AND PMAX = 0, THEN WE USE A LOCAL DISPERSION RELATION
000109      C     AT NXL(UP TO 10) DIFFERENT  VALUES OF X/L, GIVEN BY XL, WHILE EPS(I)
000110      C     EP ARE IGNORED. IF UNMAG = .FALSE., THE USUAL LOCAL APPROX. DISP. REL. IS
000111      C     USED, VALID FOR SMALL AIL. IF UNMAG = .TRUE., WE USE A DISP.REL. VALID
000112      C     FOR ALL AIL, BUT ONLY FOR GROWTH RATE GREATER THAN ION CYCLOTRON FREQUENCY
000113      C     (I.E. ALL WARM PARTICLES ARE UNMAGNETIZED, NOT YET IMPLEMENTED FOR
000114      C     RING DISTRIBUTIONS.)
000115      C     IF PMAX.GT.0, WE USE THE NON-LOCAL DISP.REL. (DEVELOPED BY LANGDON)
000116      C     A SINUSOIDAL DENSITY PROFILE, AND       SINGLX,NXL,XL,AND EP ARE
000117      C     IGNORED.
000118      C     OPT(1),OPT(2),ETC. ARE VARIOUS OPTIONS WHICH MAY BE TRUE(1) OR FALSE(0).
000119      C     OPT(1) MEANS THAT ROOTS WITH PHI(0) = 0. WILL NOT BE FOLLOWED.
000120      C     OPT(2) MEANS THAT, IF UNMAGNETIZED IONS ARE USED, ONLY ROOTS WITH
000121      C      PHASE VELOCITY LESS THAN ION THERMAL VELOCITY WILL BE FOLLOWED.
000122            READ(ICR,121)SINGLX,UNMAG,NXL,(OPT(I),I=1,10)
000123          121 FORMAT(2A1,I2,10I1)
000124            IF(SINGLX.EQ.1HT) SINGLX = .TRUE.
000125            IF(UNMAG.EQ.1HT) UNMAG = .TRUE.
000126            IF(UNMAG.EQ.1HF) UNMAG = .FALSE.
000127            IF(SINGLX.EQ.1HF) SINGLX = .FALSE.
000128            IF(.NOT.SINGLX) READ(ICR,122) (XL(I),I=1,NXL)
000129          122 FORMAT(10F5.5)
000130            SNGLX = SINGLX
000131            IF(SINGLX) NXL = 1
000132      C     RMAX IS NUMBER OF CYCLOTRON HARMONICS WANTED, PMAX*KO IS ESSENTIALLY
000133      C     A CUTOFF IN KX, SEE MRAF FOR DEF. OF EIGV (BUT EVEN IF EIGV = 2, PHI(X)
000134      C     IS PLOTTED FOR FASTEST-GROWING ROOT OF EACH BRANCH), SKYAI AND FKYAI ARE
000135      C     LIMITS OF KYAI, DK IS INCREMENT OF KYAI, AIL IS RATIO OF LARMOR RADIUS TO
000136      C     LENGTH OF PLASMA, AEAI IS RATIO OF ELECTRON TO ION LARMOR RADIUS,
000137      C     OMPSQ IS (PLASMA FREQ./GYROFREQ.)**2, MASS IS ION TO ELECTRON MASS RATIO
000138      C     DFUI AND DFUE ARE ION AND ELECTRON DIFFUSION COEFFICIENTS.EP IS THE
000139      C     DENSITY GRADIENT IF WE ARE USING THE LOCAL APPROXIMATION WITH THE SAME
000140      C     GRADIENT FOR ALL SPECIES.
000141      C     IF NSW1.NE.0, INFORMATION ON EACH ROOT IS PRINTED OUT, FOR KYAI = SKYAI
000142      C     IF NPRNT.NE.0, A PRINTER PLOT IS MADE (IN ADDITION TO CRT PLOT).
000143      C     IF NOCRT.NE..0, NO CRT PLOTS ARE MADE.
000144      C     IF THERE IS ONLY ONE SPECIES OF IONS, RING,MAXWELLIAN, OR LOSS CONE
000145      C     THEN SET NSPEC = 0. IF THERE IS MORE THAN ONE SPECIES, OR A MORE
000146      C     COMPLICATED DISTRIBUTION, THEN NSPEC IS THE NUMBER OF MAXWELLIAN
000147      C     COMPONENTS OR RING COMPONENTS, INCLUDING THOSE OF ELECTRONS.(E.G. LOSS
000148      C     CONE IONS AND MAXW. ELECTRONS WOULD REQUIRE NSPEC = 2 + 1 = 3)
000149      C     IF NSPEC.GT.0, THEN MASS, R, GAM, RE, GAME, AEAI,EP,DFUE,AND DFUI ARE
000150      C     IGNORED, BUT DUMMY VALUES MUST BE READ IN.
000151      C     ELECTRONS MUST BE READ IN LAST, MUST HAVE LESS THAN ONE TENTH THE MASS
000152      C     OF SPECIES 1, AND MUST HAVE CHARGE -1
000153      C     IRMIN IS THE LOWEST FREQUENCY LOOKED AT (IN TERMS OF THE CYCLOTRON FREQ.)
000154            READ(ICR,2) RMAX,PMAX,EIGV,SKYAI,DK,FKYAI,NSW1,NPRNT,NOCRT,NSPEC,I
000155           .RMIN
000156          2 FORMAT(3I5,3F5.3,5I5)
000157            NPP = 2*PMAX + 1
000158      C     IF WE ARE EXECUTING SEVERAL RUNS ON ONE JOB, WE MAY BE OUT OF TIME
000159      C     AT THE BEGINNING OF THIS RUN, SO CHECK TO SEE IF THIS IS THE CASE.
000160            CALL TIMECHK(ADOT,0.,ELAPSED)
000161            IF(ADOT) CALL EXIT
000162            NSPECP = NSPEC
000163            READ(ICR,5) AIL,AEAI,OMPSQ,EPSI,MASS,EP,DFUI,DFUE,BETA
000164          5 FORMAT(9F5.0)
000165      C     RE AND GAME REFER TO ELECTRONS. SEE RHCALC.
000166            READ(ICR,12) R,GAM,RE,GAME
000167           12 FORMAT(4F5.3)
000168      C     IF WE ARE USING MANY SPECIES OF PARTICLES (NSPEC.GT.0), READ IN THE
000169      C     LARMOR RADIUS, MASS, CHARGE, PARTICLE DENSITY AND PARAMETER J FOR EACH
000170      C     SPECIES, WHERE F(VPERP) = CONST.*(VPERP**J)*MAXWELLIAN
000171      C     ALSO, THE DENSITY GRADIENT EPS, AND DIFFUSION COEFFICIENT DFU.
000172      C     FOR NONLOCAL (PMAX.GT.0) CASE, EPS IS THE DENSITY VARIATION FOR EACH ION
000173      C     SPECIES. IF EPS IS LEFT BLANK, IT IS SET EQUAL TO EPSI.
000174            IF(NSPEC.LE.0) GO TO 82
000175            IF(NSPEC.GT.4) NSPEC = 4
000176            DO 83 I = 1, NSPEC
000177            READ(ICR,84) RLARM(I),AMASS(I),CHARG(I),DENSE(I),JAY(I),EPS(I),DFU
000178           .(I)
000179           84 FORMAT(4F10.5,I5,2F10.5)
000180            AMASQ(I) = AMASS(I)**2
000181            IF(EPS(I).EQ.0..AND.PMAX.GT.0) EPS(I) = EPSI
000182      C     NSP IS THE NUMBER OF ION SPECIES, USED IN CALCULATING EPSE. IF SPECIES
000183      C     I IS ELECTRONS, THEN NSP = I - 1
000184            IF(AMASS(I).LT..1*AMASS(1).AND.CHARG(I).EQ.-1.) NSP = I - 1
000185           83 CONTINUE
000186            GO TO 89
```

```
000187        82 NSPEC = 4
000188           NSP = 2
000189           DFU(1) = DFUI
000190           DFU(2) = DFUI
000191           DFU(3) = DFUE
000192           DFU(4) = DFUE
000193           DO 108 I = 1,4
000194           IF(PMAX.GT.0.AND.I.LE.NSP) EPS(I) = EPSI
000195       108 IF(PMAX.EQ.0) EPS(I) = EP
000196           AMASS(3) = 1./MASS
000197           AMASQ(3) = AMASS(3)**2
000198           IF(R.LE.0.) GO TO 98
000199           DENSE(1) = R/(R - GAM)
000200           DENSE(2) = -GAM/(R - GAM)
000201           RLARM(2) = 1./SQRT(R)
000202           GO TO 90
000203        98 JAY(1) = -1
000204        90 IF(AEAI.EQ.0.) GO TO 89
000205           RLARM(3) = AEAI
000206           IF(RE.LE.0.) GO TO 99
000207           DENSE(3) = RE/(RE - GAME)
000208           DENSE(4) = -GAME/(RE - GAME)
000209           RLARM(4) = AEAI/SQRT(RE)
000210           AMASS(4) = AMASS(3)
000211           AMASQ(4) = AMASQ(3)
000212           GO TO 89
000213        99 JAY(3) = -1
000214        89 FKYAI2 = FKYAI
000215           IF (PMAX.EQ.0) EIGV = 0
000216           IF(EIGV.LT.2) EIGV = 0
000217           IGV = EIGV
000218           AEAI1 = AEAI
000219           IF(R.GT.0.) R1 = 1./SQRT(R)
000220           IF(RE.GT.0.) RE1 = 1./SQRT(RE)
000221           KOAI = 2.0*PI*AIL
000222           KOSQ=KOAI*KOAI
000223           NSP1 = NSP + 1
000224     C     CALCULATE THE ELECTRON GUIDING CENTER DENSITY GRADIENT EPSE NEEDED TO
000225     C     NEUTRALIZE IONS WITH GUIDING CENTER DENSITY GRADIENT EPSI.
000226     C     FIRST FIND RATIO OF OSCILLATING PART OF PARTICLE DENSITY TO  OSCILLATING
000227     C     PART OF GUIDING CENTER DENSITY  FOR EACH SPECIES.
000228       101 DO 93 I = 1, NSPEC
000229           KOAE = KOAI*RLARM(I)
000230           RM = 1.
000231           IF(JAY(I).LT.0) RM = 0.
000232        93 CALL RHCALC(RM,0.,KOAE,RH(I))
000233     C     THEN FIND THIS RATIO (TIMES TOTAL ION CHARGE DENSITY) FOR IONS AS A WHOLE
000234           RHEI = 0.
000235           DO 94 I = 1, NSP
000236        94 RHEI = RHEI + DENSE(I)*CHARG(I)*RH(I)*EPS(I)
000237     C     THEN FIND FOR ELECTRONS AS A WHOLE.
000238           RHEE = 0.
000239           DO 95 I = NSP1, NSPEC
000240        95 RHEE = RHEE - DENSE(I)*CHARG(I)*RH(I)
000241           EPSE = RHEI/RHEE
000242           DO 96 I = 1, NSP
000243        96 EPSF(I) = .5*EPS(I)*OMPSQ*CHARG(I)**2*DENSE(I)/AMASS(I)
000244           DO 97 I = NSP1, NSPEC
000245        97 EPSF(I) = .5*EPSE*OMPSQ*CHARG(I)**2*DENSE(I)/AMASS(I)
000246     C     PRINT OUT PARAMETERS
000247       103 CALL PARAM(99)
000248           PRINT 7,IRMIN,RMAX
000249         7 FORMAT(40H ROOTS FOUND BETWEEN CYCLOTRON HARMONICS,I3,4H AND,I3)
000250           MASQ = MASS*MASS
000251           NPT = 0
000252           NPTC = 0
000253           NPTZ = 0
000254           NFOL = 0
000255           YMX = FLOAT(RMAX) + .5
000256           DO 114 IXL = 1, NXL
000257           ERRMAX = 1.E+20
000258           RMX =FLOAT(RMAX) + .5
000259           RHIGH = RMX
000260           RMIN = IRMIN
000261           IF(NSW1.NE.0) SW(1) = .TRUE.
000262     C.....COMPLEX CONJUGATES ARE ALSO ROOTS, IF THERE IS NO DIFFUSION
000263           SW(3) = .TRUE.
000264           DO 107 I = 1, NSPEC
000265       107 IF(DFU(I).GT.0.) SW(3) = .FALSE.
000266           IF(UNMAG) SW(3) = .FALSE.
000267           SW(4)=.TRUE.
000268           SW(5) = .TRUE.
000269           IF(UNMAG) SW(5) = .FALSE.
000270     C     PMAX = 0 MEANS WE ARE USING   A LOCAL APPROXIMATION. IF NOT USING  A
000271     C     LOCAL APPROX., THEN A ROOT AT OMEGA IMPLIES A ROOT AT -OMEGA*.
000272           IF(PMAX.GT.0) SW7 = .TRUE.
000273           KYAI = SKYAI
000274           KYSQ=KYAI*KYAI
000275           CYAI = KYAI
000276           CHI = KYSQ + PMAX*PMAX*KOSQ
000277           MINQ = 2*RMAX
000278           CALL QINT(CHI)
000279     C     IF THE RANGE OF FREQ. WE ARE INTERESTED IN (RMX - RMIN) IS TOO LARGE
000280     C     FOR *MRAF* TO HANDLE, WE BREAK IT INTO REGIONS OF CH CYCLOTRON HARMONICS
000281     C     AND LOOK AT ONE REGION AT A TIME.
000282           CH = 5.
000283           RLOW = RMIN
000284       105 RHIGH = RMX
```

```
000285          IF(RHIGH - RLOW.GT.CH+.5) RHIGH = RLOW + CH + .5
000286     C    CALCULATE QF, KMAX, AND KMIN (USED IN *DISP*)
000287          CALL QFKCAL(QF,KMAX,KMIN,RHIGH,RLOW)
000288     C    HOW MANY POLES ARE THERE.
000289          NPOLES = 0
000290          DO 92 I = 1, NSPEC
000291       92 NPOLES = NPOLES + KMAX(I) - KMIN(I) + 1
000292     C    IF THE NUMBER OF ROOTS(CONVERGED OR NOT) EXCEEDS NRTS, OR THE NUMBER
000293     C    OF UNCONVERGED ROOTS(NOT COUNTING NEGATIVES AND CONJUGATES) EXCEEDS NFL,
000294     C    IT IS NOT WORTHWHILE TO LOOK FOR MORE ROOTS (DETERMINED EMPIRICALLY)
000295          NRTS = 2*NPP*(NPOLES + 3*NSPEC - 3)
000296          NFL = NPP*NPOLES + NPOLES
000297          IF(UNMAG) NRTS = 10*NPP
000298          IF(UNMAG) NFL  = 10*NPP
000299          IF(NRTS.GT.200) NRTS = 200
000300          IF(NRTS.GT.100) NRTS = 100
000301          IF(PMAX.GT.0) GO TO 117
000302          IF(SINGLX) GO TO 116
000303     C    DREL(I) IS THE RELATIVE DENSITY, N(X)/NO FOR SPECIES I
000304     C    EPS(I) IS THE LOCAL DENSITY GRADIENT OF SPECIES I
000305          CALL XLSET(XL(IXL),XKO,EPSI,EPSE,NSP,NSP1,NSPEC,EPS,DREL,RH,KOAI)
000306          GO TO 117
000307       116 DO 118 I = 1, NSPEC
000308       118 DREL(I) = 1.
000309       117 CONTINUE
000310     C    STARTING VALUES FOR ROOT SEARCH. IF THESE ARE TOO CLOSE TOGETHER,
000311     C    IT TAKES TOO LONG TO CONVERGE TO A ROOT.
000312          START = 0.
000313          IF(UNMAG) GO TO 112
000314          P1 = -1.952 + RLOW
000315          P2 = -0.951 + RLOW
000316          P3 =  0.051 + RLOW
000317          GO TO 113
000318       112 P1 = .001
000319          P2 = EPSI*RH(1)*KOAI*KYAI + .0019
000320          P3 = SQRT(EPSI*RH(1)*KOAI)*KYAI + .0027
000321       113 SW(6) = SW(3)
000322        14 CALL  MRAF(DISP,RT,C,NRTS,P1,P2,P3,1.E-06,1.E-12,25)
000323          IF(SW(1)) PRINT 10, (C(I),I=1,NRTS)
000324        10 FORMAT(1X,2015)
000325          IF(SKYAI.GE.FKYAI) GO TO 114
000326          IF(SW(1)) CALL PAGE
000327          DO 21 I = 1, RTC
000328          IF (.NOT. C(I)) GO TO 21
000329          IF(REAL(RT(I)).LT.-1.E-06.AND.SW7) GO TO 21
000330          IF(REAL(RT(I)).LT.0..AND.SW7) RT(I) = CMPLX(0.,AIMAG(RT(I)))
000331          IF(AIMAG(RT(I)).LT.-1.E-06.AND.SW(3)) GO TO 21
000332          IF(CABS(RT(I)).LT.RLOW - 1.) GO TO 21
000333          IF(RHIGH.LT.RMX.AND.CABS(RT(I)).GT.RHIGH - 1.5) GO TO 21
000334          IF(CABS(RT(I)).GT.RMX) GO TO 21
000335          NSTART = NSTART + 1
000336     C    USE THE GOOD ROOTS AS STARTING POINTS FOR *FOLLOW*.
000337          XSTART(NSTART) = KYAI
000338          YSTART(NSTART) = RT(I)
000339          ZEH(NSTART) = PHIZ(I)
000340          IF(NSTART.GE.100) GO TO 106
000341        21 CONTINUE
000342          RLOW = RLOW + CH
000343          IF(RHIGH.LT.RMX) GO TO 105
000344          NSTRT1 = NSTART
000345     C    WE HAVE FOUND ALL THE STARTING POINTS. NOW PREPARE TO FOLLOW EACH ROOT.
000346       106 RHIGH = 1.E+06
000347          SW31 = SW(3)
000348          SW71 = SW7
000349          SW(3) = .FALSE.
000350          SW(4) = .FALSE.
000351          SW(5) = .FALSE.
000352          SW(6) = .FALSE.
000353          SW7 = .FALSE.
000354        56 NFOL = NFOL + 1
000355          SW(1) = .FALSE.
000356          SW(2) = .FALSE.
000357     C    IF WE HAVE RUN OUT OF SPACE TO STORE ROOTS, OR HAVE FOLLOWED ALL THE
000358     C    ROOTS, DO END PROCEDURES.
000359          IF(NFOL.GT.100) GO TO 55
000360          IF(NFOL.GT.NSTART) GO TO 125
000361     C    IF WE ONLY WANT TO FOLLOW SOME OF THE ROOTS, STICK IN EXTRA CARDS HERE
000362          IF(ZEH(NFOL).AND.PMAX.GT.0.AND.OPT(1)) GO TO 56
000363          IF(REAL(YSTART(NFOL)).GT.SKYAI.AND.XSTART(NFOL).EQ.SKYAI.AND.UNMAG
000364         ..AND.OPT(2)) GO TO 56
000365          IF(REAL(YSTART(NFOL)).LT.IRMIN) GO TO 56
000366          IF(.NOT.SW71.AND.UNMAG) GO TO 128
000367          IF(REAL(YSTART(NFOL)).LT.IRMIN) GO TO 56
000368     C    FOLLOW THE PHI(0)'0 ROOTS BEFORE FOLLOWING THE PHI(0)=0 ROOTS, SINCE
000369     C    THE FASTEST-GROWING ROOT IS MORE LIKELY TO HAVE PHI(0).NE.0.
000370          IF(.NOT.ZEH(NFOL).OR.NSTART.GE.100) GO TO 128
000371          DO 138 I = NFOL, NSTART
000372          IF(.NOT.ZEH(I)) GO TO 139
000373       138 CONTINUE
000374          GO TO 128
000375       139 NSTART = NSTART + 1
000376          XSTART(NSTART) = XSTART(NFOL)
000377          YSTART(NSTART) = YSTART(NFOL)
000378          ZEH(NSTART) = ZEH(NFOL)
000379          GO TO 56
000380       128 CALL TIMECHK(ADOT,0.,ELAPSED)
```

```
000381          PRINT 57, NFOL,YSTART(NFOL),XSTART(NFOL),ELAPSED
000382       57 FORMAT (5H ROOT,I4,19H STARTS AT OMEGA = ,2E22.13,8H KYAI = ,E22.1
000383         .3,5X,8H TIME IS,F8.3)
000384          IF(CABS2(YSTART(NFOL)).GT.1.E-12) GO TO 58
000385          PRINT 59, NFOL
000386       59 FORMAT(16H OMEGA = 0. ROOT, I4,25H NOT FOLLOWED EXPLICITLY.    )
000387          GO TO 56
000388       58 CALL FOLLOW(XSTART(NFOL), YSTART(NFOL),ZEH(NFOL))
000389          PRINT 76, NFOL,Y2  ,KYAI
000390       76 FORMAT(5H ROOT,I4,17H ENDS AT OMEGA =  ,2E22.13,8H KYAI =  ,E22.13)
000391     C    CLASSIFY THE ROOTS ACCORDING TO WHETHER IT IS REAL OR COMPLEX, AND AC-
000392     C    CORDING TO WHETHER PHI(0) = 0 (PHIZ = .TRUE.) OR NOT. PHI(0) = 0
000393     C    FOR ABOUT HALF THE ROOTS, AND THIS IS A CONVENIENT WAY OF CLASSIFYING
000394     C    THEM SO AS NOT TO CONFUSE ONE ROOT WITH ANOTHER NEARBY.
000395          IF(ZEH(NFOL)) GO TO 61
000396          IF(NPT.EQ.NRCUT(NR)) GO TO 60
000397          NR = NR + 1
000398     C    NRCUT, NZCUT, AND NCCUT TELL THE CRT PLOTTING ROUTINES WHEN IN THE LIST OF
000399     C    POINTS ONE BRANCH ENDS AND ANOTHER BEGINS, SO CONSECUTIVE POINTS FROM
000400     C    DIFFERENT BRANCHES WILL NOT BE JOINED.
000401     C    NR,NRCUT, NPT FOR REAL ROOTS, PHIZ = .FALSE.
000402     C    NZ, NZCUT, NPTZ FOR REAL ROOTS, PHIZ = .TRUE.
000403     C    NC, NCCUT, NPTC FOR COMPLEX ROOTS
000404          NRCUT(NR) = NPT
000405          GO TO 60
000406       61 IF(NPTZ.EQ.NZCUT(NZ)) GO TO 60
000407          NZ = NZ + 1
000408          NZCUT(NZ) = NPTZ
000409          IF(PMAX.EQ.0.AND..NOT.SINGLX) XLR(NZ) = XL(IXL)
000410       60 IF(NPTC.EQ.NCCUT(NC)) GO TO 63
000411          NC = NC + 1
000412          NCCUT(NC) = NPTC
000413          IF(PMAX.EQ.0.AND..NOT.SINGLX) XLC(NC) = XL(IXL)
000414     C     FIND THE FASTEST-GROWING ROOT IN THIS BRANCH
000415          INIT = NCCUT(NC-1) + 1
000416          MAXGRO = INIT
000417          DO 130 I = INIT, NPTC
000418      130 IF(YI(I),GT.YI(MAXGRO)) MAXGRO = I
000419          SIDE(NC-1) = SIGN(1.,YR(MAXGRO))
000420          MXGRO(NC-1) = MAXGRO
000421       63 IF(NCRIT.GT.100.OR.NPT.GE.2000) GO TO 550
000422          IF(NPTZ.GE.4000.OR.NPTC.GE.1000) GO TO 550
000423     C     ABOT MEANS ALMOST OUT OF TIME
000424          IF(WARN(0).OR.ABOT) GO TO 109
000425          GO TO 56
000426      109 PRINT 110
000427      110 FORMAT (19H ALMOST OUT OF TIME)
000428          GO TO 55
000429      550 PRINT 104
000430      104 FORMAT (28H NO MORE ROOM TO STORE ROOTS)
000431      125 IF(PMAX.GT.0) GO TO 55
000432          NFOL = NFOL - 1
000433          IF(NXL.GT.1) PRINT 129, XL(IXL+1)
000434      129 FORMAT(6H X/L =,F5.2)
000435      114 CONTINUE
000436       55 RMX = YMX
000437          EIGV = IGV
000438          CALL TIMECHK(ABOT,0.,ELAPSED)
000439          PRINT 78, ELAPSED
000440       78 FORMAT(5X,8H TIME IS,F8.3)
000441     C     FIND THE FASTEST-GROWING ROOT
000442          IF(NPTC.EQ.0) GO TO 43
000443          MAXGRO = MXGRO(1)
000444          NC1 = NC - 1
000445          DO 42 I = 1, NC1
000446          MXGROI = MXGRO(I)
000447       42 IF(YI(MXGROI)  .GT.YI(MAXGRO)) MAXGRO = MXGRO(I)
000448     C     FIND THE MAXIMUM GAMMA/KYSQ
000449          MGAMK2 = 1
000450          GAMK2 = YI(1)/XC(1)**2
000451          DO 157 I = 2,NPTC
000452          IF(YI(I)/XC(I)**2.GT.GAMK2) MGAMK2 = I
000453      157 IF(MGAMK2.EQ.I) GAMK2 = YI(I)/XC(I)**2
000454          IF(SW(1)) CALL PAGE
000455          DO 123 I = 2, NC
000456      123 IF(NCCUT(I).GE.MAXGRO) GO TO 124
000457      124 XLGMX = XLC(I)
000458          PRINT 44, YR(MAXGRO),YI(MAXGRO),XC(MAXGRO),XLGMX
000459       44 FORMAT(29H FASTEST-GROWING ROOT, OMEGA=,2E10.3,6H KYAI=,
000460         .E10.4,7H X/L = ,F6.3)
000461          PRINT 158, GAMK2, XC(MGAMK2)
000462      158 FORMAT("MAXIMUM GAMMA/KYSQ =",E12.4,"AT KYAI =",E12.4)
000463          IF(SW71.OR..NOT.UNMAG) GO TO 43
000464          IF(.NOT.PRDAMP) GO TO 160
000465          IF(NPTZ.EQ.0) GO TO 43
000466          DO 156 I = 1, NPTZ
000467      156 IF(YZ(I).GT.RMX) RMX = YZ(I)
000468          GO TO 43
000469      160 IF(SINGLX) GO TO 43
000470          IF(YR(MAXGRO).LT.0.) XLGMX = -XLGMX
000471          RMX = RMAX + .5
000472          DO 127 I = 1, NPTC
000473          IF(YR(I).LT.0) YR(I) =  -YR(I)
000474      127 IF(YR(I).GT.RMX) RMX = YR(I)
000475     C     DO THE PRINTER PLOTS, IF DESIRED.
000476       43 IF(NPRNT.EQ.0) GO TO 137
```

```
000477          YINC = (RMX - RMIN)/50.
000478          XMIN = IFIX(SKYAI)
000479          RANGE = FKYAI - XMIN
000480          XRANGE = 500.
000481          IF(RANGE.LE.250.) XRANGE = 250.
000482          IF(RANGE.LE.100.) XRANGE = 100.
000483          IF(RANGE.LE.75.) XRANGE = 75.
000484          IF(RANGE.LE.50.) XRANGE = 50.
000485          IF(RANGE.LE.20.) XRANGE = 20.
000486          IF(RANGE.LE.10.) XRANGE = 10.
000487          IF(RANGE.LE. 5.) XRANGE =  5.
000488          XMAX = XMIN + XRANGE
000489          XINC = XRANGE/100.
000490          IF(PRDAMP) GO TO 154
000491          IF(UNMAG) GO TO 80
000492      154 IF(.NOT.SW(1)) CALL PAGE
000493          IF(NPT.EQ.0) GO TO 79
000494          CALL PRNPLT(X,Y,XMAX,XINC,RMX,YINC,0,0,NPT)
000495          PRINT 39
000496          CALL PAGE
000497       79 IF(NPTZ.EQ.0) GO TO 80
000498          CALL PRNPLT(XZ,YZ,XMAX,XINC,RMX,YINC,0,0,NPTZ)
000499          PRINT 39
000500       39 FORMAT(11H REAL ROOTS)
000501          CALL PARAM(99)
000502       80 IF(NPTC.EQ.0) GO TO 48
000503          IF(.NOT.SW(1)) CALL PAGE
000504          CALL PRNPLT(XC,YR,XMAX,XINC,RMX,YINC,0,0,NPTC)
000505          CALL PARAM(99)
000506          PRINT 40
000507       40 FORMAT(28H REAL PART OF COMPLEX ROOTS,)
000508          CALL PAGE
000509          YMAX = IFIX(YI(MAXGRO) + 1.)
000510          YINC = YMAX/50.
000511          CALL PRNPLT(XC,YI,XMAX,XINC,YMAX,YINC,0,0,NPTC)
000512          CALL PARAM(99)
000513          PRINT 41
000514       41 FORMAT(29H IMAG. PART OF COMPLEX ROOTS,)
000515      C   FIND PHI(P) FOR THE FASTEST-GROWING MODE ON EACH BRANCH
000516      137 NC1 = NC - 1
000517          IF(NPRNT.NE.0) CALL PAGE
000518          CALL SETCH(1.,42.,1,0,1,0,0)
000519          IF(IGV.LT.2) GO TO 145
000520          DO 131 I = 1,NC1
000521          MXGROI = MXGRO(I)
000522          OMEGA = CMPLX(YR(MXGROI),YI(MXGROI))
000523          KYAI = XC(MXGROI)
000524          CYAI = KYAI
000525          KYSQ = KYAI*KYAI
000526          CHI = KYSQ + PMAX*PMAX*KOSQ
000527          MINQ = 2.*REAL(OMEGA)
000528          CALL QINT(CHI)
000529          ZERO = DISP(OMEGA)
000530          CALL SAVPHI
000531          PHI(NRTSV,NPP+1) = OMEGA
000532          PHI(NRTSV,NPP+2) = CMPLX(KYAI,0.)
000533          IF(MXGROI.EQ.MAXGRO) CALL DETAIL(NRTSV)
000534          CALL TIMECHK(ADOT,0.,ELAPSED)
000535          PRINT 78, ELAPSED
000536          SUMPHI = 0.
000537          DO 133 J = 1,NPP
000538          ABSPHI(J) = CABS(PHI(NRTSV,J))
000539          SUMPHI = SUMPHI + ABSPHI(J)**2
000540      133 IF(ABSPHI(J).NE.0.) ANGPHI(J)  = 57.29*AIMAG(CLOG(PHI(NRTSV,J)))
000541          GOOD(NRTSV) = .TRUE.
000542          IBRANCH(NRTSV) = I
000543          NRTSV = NRTSV + 1
000544          PHNORM = 1./SQRT(SUMPHI)
000545          DO 132 J = 1, NPP
000546      132 ABSPHI(J) = ABSPHI(J)*PHNORM
000547          PRINT 136, I, OMEGA, KYAI
000548          WRITE(IGRAPH,136) I,OMEGA,KYAI
000549      136 FORMAT(31H FASTEST-GROWING WAVE OF BRANCH, I3, 9H OMEGA = ,2E21.12
000550         ./8H KYAI = ,E22.13)
000551          PRINT 134, (ABSPHI(J),J=1,NPP)
000552          WRITE(IGRAPH,134) (ABSPHI(J),J=1,NPP)
000553      134 FORMAT(6E12.4)
000554          PRINT 135, (ANGPHI(J),J=1,NPP)
000555          WRITE(IGRAPH,135) (ANGPHI(J),J=1,NPP)
000556      135 FORMAT(6F12.2)
000557          WRITE(IGRAPH,159)
000558      159 FORMAT(" ")
000559      131 CONTINUE
000560          GO TO 147
000561      145 IF(SINGLX) GO TO 48
000562          DO 146 I = 1, NC1
000563          MXGROI = MXGRO(I)
000564          KYAI = XC(MXGROI)
000565          OMEGA = CMPLX(YR(MXGROI),YI(MXGROI))
000566          KYSQ = KYAI*KYAI
000567          XX = XLC(I+1)*SIDE(I)
000568          IF(XX.LT.0.) XX = XX + 1.
000569          PRINT 148, I, OMEGA, KYAI, XX
```

```
000570          WRITE(IGRAPH,148) I, OMEGA, KYAI, XX
000571        148 FORMAT(31H FASTEST-GROWING WAVE OF BRANCH, I3, 9H OMEGA = ,2E20.12
000572           ./8H KYAI = ,E20.12,5H X/L=,F6.3)
000573          IF(.NOT.UNMAG) GO TO 146
000574          CALL XLSET(XX,XKO,EPSI,EPSE,NSP,NSP1,NSPEC,EPS,DREL,RH,KOAI)
000575          ZERO = DISP(OMEGA)
000576          DFDW = 1.E+04*DISP(OMEGA + .0001)
000577          KXSQ = .002
000578          D2FDK2 = 1.E+03*DISP(OMEGA)
000579          KXSQ = 0.
000580          XX = XX + .0001
000581          CALL XLSET(XX,XKO,EPSI,EPSE,NSP,NSP1,NSPEC,EPS,DREL,RH,KOAI)
000582          DFDX = 1.E+04*DISP(OMEGA)*AIL
000583          TRZERO = 1.E+12*CABS2(ZERO)
000584          IF(CABS2(DFDW).LT.TRZERO.OR.CABS2(D2FDK2).LT.TRZERO.OR.CABS2(DFDX)
000585         ..LT.TRZERO) GO TO 151
000586          D2WDK2 = D2FDK2/DFDW
000587          DWDX = DFDX/DFDW
000588          SPREAD = SQRT(CABS(D2WDK2)/AIMAG(OMEGA))
000589          CHANGE = CABS(OMEGA/DWDX)
000590          ERROR = AIMAG(OMEGA)*SPREAD/CHANGE
000591          GAMMA = AIMAG(OMEGA) - ERROR
000592          WRITE(IGRAPH,150) SPREAD,CHANGE,GAMMA
000593          PRINT 150, SPREAD,CHANGE, GAMMA
000594        150 FORMAT(                    " DURING ONE GROWTH PERIOD, A WAVE PACKET
000595         .SPREADS A DISTANCE OF",E12.3," ION LARMOR RADII"/" THE FREQUENCY
000596         .FOR A GIVEN KYAI CHANGES OVER A DISTANCE OF",E12.3," ION LARMOR RA
000597         .DII"/" GAMMA PROBABLY CLOSER TO ",E12.3/)
000598          GO TO 153
000599        151 PRINT 152, ZERO,DFDW,D2FDK2,DFDX
000600        152 FORMAT(" SOMETHING WRONG,DISP = ",2E22.13/"DFDW,D2FDK2,DFDX = ",6E
000601           .18.11)
000602        153 CONTINUE
000603        146 CONTINUE
000604        147 CALL FRAME(1)
000605      C   PLOT THE EIGENFUNCTIONS PHI(X) IF DESIRED.
000606         48 CALL EIGPLT
000607          CALL TIMECHK(ACOT,0.,ELAPSED)
000608          PRINT 78, ELAPSED
000609          IF(NOCRT.NE.0) CALL EXIT
000610      C DO THE CRT PLOT OF THE REAL PART OF ROOTS.
000611          XMIN = IFIX(SKYAI)
000612          XMAX = IFIX(FKYAI+1.)
000613          IF(XMAX.EQ.FKYAI + 1.) XMAX = FKYAI
000614          IF(XMAX - XMIN.GT.100.) XMAX = 10.*IFIX(XMAX/10. + 1.)
000615          IF(XMAX.EQ.FKYAI + 10.) XMAX = FKYAI
000616          YMAX = IFIX(RMX - .5) + 1.5
000617          IF(YMAX.EQ.RMX + 1.) YMAX = RMX
000618          YMIN = IFIX(RMIN) -1.E-06
000619          NX2 = XMAX - XMIN
000620          IF(NX2.GT.100) NX2 = NX2/10
000621          NX3 = 1
000622          IF(NX2.LT.10) NX3 = 2
000623          IF(NX2.LT. 5) NX3 = 5
000624          NY2 = 2.*(YMAX - YMIN)
000625          CALL MAPS (XMIN,XMAX,YMIN,YMAX,0.11328,1.0,0.30,1.0)
000626          CALL PAGE
000627          IF(.NOT.SW31) TEXTURE = 3.
000628          IF(PRDAMP) GO TO 155
000629          IF(UNMAG) GO TO 66
000630        155 IF(NR.LE.1) GO TO 64
000631          NR = NR - 1
000632          DO 65 J = 1, NR
000633          IMIN = NRCUT(J) + 1
000634          IMAX = NRCUT(J+1)
000635          NUM = MINO(IMAX - IMIN + 1, 500)
000636          PRINT 77, J, NUM
000637         77 FORMAT(14H PLOTTING ROOT,I4,I20,7H POINTS)
000638          K = 0
000639          DO 72 I = IMIN,IMAX
000640          K = K + 1
000641          YPLOT(K) = Y(I)
000642          IF(YPLOT(K).LT.YMIN) YPLOT(K) = YMIN
000643          XPLOT(K) = X(I)
000644         72 IF(K.GT.500) GO TO 65
000645         65 CALL TRACET(TEXTURE,XPLOT,YPLOT,NUM)
000646         64 IF(NZ.LE.1) GO TO 66
000647          NZ = NZ - 1
000648          DO 67 J = 1,NZ
000649          IMIN = NZCUT(J) + 1
000650          IMAX = NZCUT(J+1)
000651          NUM = MINO(IMAX - IMIN + 1, 500)
000652          IF(SINGLX.OR.PMAX.GT.0) PRINT 77, J, NUM
000653          IF(.NOT.SINGLX.AND.PMAX.EQ.0) PRINT 126, J, NUM, XLR(J+1)
000654        126 FORMAT(14H PLOTTING ROOT,I4,I20,7H POINTS, 6H X/L =,F5.2)
000655          K = 0
000656          DO 73 I = IMIN, IMAX
000657          K = K + 1
000658          YPLOT(K) = YZ(I)
000659          IF(YPLOT(K).LT.YMIN) YPLOT(K) = YMIN
000660          XPLOT(K) = XZ(I)
000661         73 IF(K.GT.500) GO TO 67
000662         67 CALL TRACET(TEXTURE,XPLOT,YPLOT,NUM)
000663         66 IF(NC.LE.1) GO TO 68
000664          NC = NC - 1
000665      C   USE DOTTED LINE FOR COMPLEX ROOTS.
```

```
000666           TEXTURE = 2.
000667           IF(.NOT.SW31) TEXTURE = 0.
000668           DO 69 J = 1, NC
000669           IMIN = NCCUT(J) + 1
000670           IMAX = NCCUT(J+1)
000671           NUM = MINO(IMAX - IMIN + 1, 500)
000672           IF(SINGLX.OR.PMAX.GT.0) PRINT 77, J, NUM
000673           IF(.NOT.SINGLX.AND.PMAX.EQ.0) PRINT 126, J, NUM, XLC(J+1)
000674           K = 0
000675           DO 74 I = IMIN, IMAX
000676           K = K + 1
000677           YPLOT(K) = YR(I)
000678           IF(YPLOT(K).LT.YMIN) YPLOT(K) = YMIN
000679           XPLOT(K) = XC(I)
000680        74 IF(K.GT.500) GO TO 69
000681        69 CALL TRACET(TEXTURE,XPLOT,YPLOT,NUM)
000682        68 TEXTURE = 0.
000683           TVRNGE = TVXMAX - TVXMIN
000684           FACTX= TVRNGE/(XMAX-XMIN)
000685           TVRNGE = TVYMAX - TVYMIN
000686           FACTY = TVRNGE/(YMAX - YMIN)
000687     C    LIST THE PARAMETERS UNDER THE CRT PLOT.
000688           CALL PARAM(98)
000689           CALL SETCH (1.,7.,1,0,1,0,0)
000690           WRITE(IGRAPH,46)
000691        46 FORMAT(19H REAL PART OF ROOTS)
000692           CALL FRAME(1)
000693     C    DO THE CRT PLOT OF THE IMAGINARY PART OF ROOTS.
000694           IF(NPTC.EQ.0) GO TO 47
000695            NY2  = IFIX(YI(MAXGRO) + 1.)
000696           YMAX = NY2
000697           YMIN = 0.
000698           NY2 = 2.*NY2
000699           CALL MAPS (XMIN,XMAX,YMIN,YMAX,0.11328,1.0,0.30,1.0)
000700           IF(NC.LT.1) GO TO 70
000701           DO 71 J = 1,NC
000702           IMIN = NCCUT(J) + 1
000703           IMAX = NCCUT(J+1)
000704           NUM = MINO(IMAX - IMIN + 1, 500)
000705           K = 0
000706           DO 75 I = IMIN, IMAX
000707           K = K + 1
000708           YPLOT(K) = YI(I)
000709           IF(YPLOT(K).LT.YMIN) YPLOT(K) = YMIN
000710           XPLOT(K) = XC(I)
000711        75 IF(K.GT.500)GO TO 71
000712        71 CALL TRACET(TEXTURE,XPLOT,YPLOT,NUM)
000713        70 FACTY = TVRNGE/(YMAX - YMIN)
000714           CALL PARAM(98)
000715           CALL SETCH(1.,9.,1,0,1,0,0)
000716           WRITE(IGRAPH,44) YR(MAXGRO),YI(MAXGRO),XC(MAXGRO),XLGMX
000717           WRITE(IGRAPH,158) GAMK2,XC(MGAMK2)
000718           WRITE(IGRAPH,41)
000719        47 CALL FRAME(1)
000720           CALL PLOTE
000721           CALL TIMECHK(AOUT,0.,ELAPSED)
000722           PRINT 78, ELAPSED
000723           CALL EXIT
000724           END
```

```
000725
000726           SUBROUTINE XLSET(XL,XKO,EPSI,EPSE,NSP,NSP1,NSPEC,EPS,DREL,RH,KOAI)
000727     C    DOES EVERYTHING THAT HAS TO BE DONE WHEN A NEW VALUE OF X/L IS USED.
000728           REAL KOAI
000729           DIMENSION EPS(NSPEC),DREL(NSPEC),RH(NSPEC)
000730           DATA PI/3.141592653589/
000731           XKO = 2.*PI*XL
000732           DO 115 I = 1, NSP
000733           DREL(I) = 1. + EPSI*RH(I)*COS(XKO)
000734       115 EPS(I) = -EPSI*RH(I)*KOAI*SIN(XKO)/DREL(I)
000735           DO 100 I = NSP1,NSPEC
000736           DREL(I) = 1. + EPSE*RH(I)*COS(XKO)
000737       100 EPS(I) = -EPSE*RH(I)*KOAI*SIN(XKO)/DREL(I)
000738           RETURN
000739           END
```

```
000740
000741          SUBROUTINE DETAIL(N)
000742      C  CALCULATES DETAILED INFORMATION ON ONE ROOT,PHI, RHOE, AND RHOI.
000743      C   BEFORE CALLING THIS ROUTINE, IT IS NECESSARY TO CALL DISP AND SAVPHI
000744      C   WHICH CALCULATE ARH AND PHI, USED IN THIS SUBROUTINE. OMEGA AND KYAI MUST
000745      C   BE STORED IN PHI(N,NPP+1) AND PHI(N,NPP+2) RESPECTIVELY, BEFORE CALLING
000746      C   THIS SUBROUTINE.
000747            LOGICAL DTCALC
000748            COMPLEX A(20,20), PHI(18, 22),PHNORM, RHNORM
000749            COMPLEX ARH(20,20,4),                OMEGA, RHOE(20), RHOI(20)
000750            COMPLEX PHI1(20)
000751            REAL KYAI
000752            COMMON/CIO/ ICR,IHSP,IGRAPH
000753            COMMON/CDETAL/ DTCALC, ARH
000754            COMMON /PHIC/ A,      NRTSV, PHI
000755            COMMON/CSPEC/ RLARM(4),AMASS(4),CHARG(4),JAY(4),DENSE(4),NSPEC
000756            COMMON/PMX/NPP
000757            COMMON/CNSP/ NSP,NSP1
000758            COMMON/CTVY/TVY
000759            COMMON/TVTUNE/LPENON,LPENOFF,ITALICS,IWINK,INTENSE,IRIGHT,IUP
000760            DATA PI/3.1415926535/
000761            PRINT 19, A(1,1),A(1,2)
000762            NP1 = NPP - 1
000763            DO 20 I = 2,NP1
000764         20 PRINT 19, A(I,I-1),A(I,I),A(I,I+1)
000765            PRINT 19, A(NPP,NP1), A(NPP,NPP)
000766         19 FORMAT(6E13.4)
000767            DO 22 J = 1,NSPEC
000768            PRINT 23, J
000769         23 FORMAT(/"MATRIX FOR SPECIES",I3/)
000770            PRINT 19, ARH(1,1,J),ARH(1,2,J)
000771            DO 21 I = 2,NP1
000772         21 PRINT 19, ARH(I,I-1,J),ARH(I,I,J),ARH(I,I+1,J)
000773         22 PRINT 19, ARH(NPP,NP1,J),  ARH(NPP,NPP,J)
000774            KYAI = REAL(PHI(N,NPP+2))
000775      C    CALCULATE RHOE(P),  RHOI(P)
000776            DO 15 I = 1, NPP
000777            RHOE(I) = 0.
000778         15 RHOI(I) = 0.
000779            NP1 = NPP - 1
000780            DO 14 J = NSP1, NSPEC
000781            RHOE(  1) = ARH(  1,  1,J)*PHI(N,  1) + ARH(  1,  2,J)*PHI(N,  2)
000782           .+RHOE(1)
000783         14 RHOE(NPP) = ARH(NPP,NP1,J)*PHI(N,NP1) + ARH(NPP,NPP,J)*PHI(N,NPP)
000784           .+RHOE(NPP)
000785            DO 16 J = 1, NSP
000786            RHOI(  1) = ARH(  1,  1,J)*PHI(N,  1) + ARH(  1,  2,J)*PHI(N,  2)
000787           .+RHOI(1)
000788         16 RHOI(NPP) = ARH(NPP,NP1,J)*PHI(N,NP1) + ARH(NPP,NPP,J)*PHI(N,NPP)
000789           .+RHOI(NPP)
000790            DO 1 I = 2, NP1
000791            DO 17 J = NSP1, NSPEC
000792         17 RHOE(I) = ARH(I,I-1,J)*PHI(N,I-1) + ARH(I,I,J)*PHI(N,I) + ARH(I,I+
000793           .1,J)*PHI(N,I+1) + RHOE(I)
000794            DO 18 J = 1, NSP
000795         18 RHOI(I) = ARH(I,I-1,J)*PHI(N,I-1) + ARH(I,I,J)*PHI(N,I) + ARH(I,I+
000796           .1,J)*PHI(N,I+1) + RHOI(I)
000797          1 CONTINUE
000798      C    NORMALIZE PHI AND RHO
000799            SUM = 0.
000800            DO 3 I = 1, NPP
000801          3 SUM = SUM + CABS2(PHI(N,I))
000802            PHNORM = CMPLX(SQRT(SUM),0.)
000803            RHNORM = CMPLX(4.*PI,0.)*PHNORM
000804            DO 4 I = 1, NPP
000805            PHI(N, I) = PHI(N,I)/PHNORM
000806            RHOI(I) = RHOI(I)/RHNORM
000807          4 RHOE(I) = RHOE(I)/RHNORM
000808      C    THIS GIVES RHO*AI**2 IN SAME UNITS AS PHI
000809            WRITE(IGRAPH,13)
000810         13 FORMAT(9X,"PHI(P)",16X,"RHOE(P)",17X,"RHOI(P)",12X,"P")
000811            DO 6 I = 1, NPP
000812            IP = I - (NPP+1)/2
000813            WRITE(IGRAPH,7) PHI(N,I),RHOE(I),RHOI(I),IP
000814          7 FORMAT(6E12.4,I5)
000815          6 CONTINUE
000816            DO 10 I = 1, NPP
000817         10 PHI1(I) = PHI(N,I)
000818            CALL PLOT(PHI1,NPP,PHIMAX)
000819            CALL SETCH(1.,10.,1,0,1,0,0)
000820            WRITE(IGRAPH,11)
000821         11 FORMAT(/"POTENTIAL FROM X = 0 TO X = L, FROM Y = 0 TO Y = 2PI/KY")
000822            WRITE(IGRAPH,9) PHI(N,NPP+1),KYAI
000823            CALL PARAM(98)
000824          9 FORMAT(9H OMEGA = ,2E15.6,8H KYAI = ,E15.6)
000825            CALL PLOT(RHOE,NPP,RHEMAX)
000826            RHEMAX = RHEMAX/PHIMAX
000827            NAME = 9HELECTRON
000828             CALL SETCH(1.,10.,1,0,1,0,0)
000829            WRITE(IGRAPH,12) NAME,RHEMAX
000830         12 FORMAT(/1X,A9,"CHARGE DENSITY, MAXIMUM = ",E12.4,14H *PHIMAX/AI**2
000831           .)
000832            WRITE(IGRAPH,9) PHI(N,NPP+1),KYAI
000833            CALL PARAM(98)
000834            CALL PLOT(RHOI,NPP,RHIMAX)
000835            NAME = 9H      ION
000836            CALL SETCH(1.,10.,1,0,1,0,0)
000837            WRITE(IGRAPH,12) NAME,RHIMAX
000838            WRITE(IGRAPH,9) PHI(N,NPP+1),KYAI
000839            CALL PARAM(98)
000840            CALL FRAME(1)
000841            CALL SETCH(1.,42.,1,0,1,0,0)
000842            RETURN
000843            END
```

```
000844
000845            SUBROUTINE PLOT(F,NP,FMAX)
000846            EXTERNAL PLFUNC
000847            COMPLEX F(20), FTWIDL(20)
000848            DIMENSION C(11)
000849            COMMON/CONPLOT/ NPP,FTWIDL,FMAX1
000850            DATA K1,K2,NX,NY /-11,6,100,50/
000851            NPP = NP
000852            CALL FRAME(1)
000853            C(1) = -1.0
000854            C(2) = 1.0
000855            DO 1 I = 1,NP
000856          1 FTWIDL(I) = F(I)
000857            CALL MAPS(0.,1.,0.,1.,0.11328,1.0,0.30,1.0)
000858            FMAX1 = 1.
000859            FMAX = 0.
000860            DO 2 IY = 1,50
000861            Y = FLOAT(IY)/50.
000862            DO 3 IX = 1,100
000863            X = FLOAT(IX)/100.
000864            GENERIC PLFUNC
000865            FXY = PLFUNC(X,Y)
000866          3 IF(FXY.GT.FMAX) FMAX = FXY
000867          2 CONTINUE
000868            FMAX1 = FMAX
000869            CALL LCURVS1(K1,C,K2,NX,NY)
000870            RETURN
000871            END
000872
000873            SUBROUTINE LCURVS1(K1,C,K2,NX,NY)
000874            EXTERNAL PLFUNC
000875            DIMENSION C(1)
000876            CALL LCURVS(K1,C,K2,PLFUNC,NX,NY)
000877            RETURN
000878            END
000879
000880            GENERIC PLFUNC
000881            FUNCTION PLFUNC(X,Y)
000882            COMPLEX F(20), EIKY, EIPKO, FP
000883            INTEGER PMAX
000884            COMMON/CONPLOT/ NP,F,FMAX
000885            DATA PI/3.141592653589/
000886            PMAX = (NP-1)/2
000887            FP = 0.
000888            DO 1 J = 1,NP
000889            P = J - PMAX - 1
000890            PKO = 2.*X*PI*P
000891            EIPKO = CEXP(CMPLX(0.,PKO))
000892          1 FP = FP + F(J)*EIPKO
000893            YKY = 2.*PI*Y
000894            EIKY = CEXP(CMPLX(0.,YKY))
000895            PLFUNC = REAL(FP*EIKY)/FMAX
000896            RETURN
000897            END
```

```
000898
000899            SUBROUTINE FOLLOW(XO,YO,PHIZP)
000900      C     THIS SUBROUTINE FOLLOWS A GIVEN ROOT THROUGH INCRESING VALUES OF KYAI.
000901      C     THE STARTING POINTS FOR THE SEARCH FOR THE ROOT AT EACH NEW VALUE OF
000902      C     KYAI (DONE BY *MRAF*) ARE DETERMINED BY MAKING A PARABOLIC FIT FOR THE
000903      C     LAST 3 VALUES OF KYAI AND EXTENDING THIS PARABOLA TO THE NEW KYAI.
000904      C     IF NO ROOT IS FOUND WITHIN ERRMAX OF THE STARTING POINT P3, THE INCREMENT
000905      C     DELT2 IS CUT IN HALF AND WE TRY AGAIN. THIS IS ALSO DONE IF THE ONLY
000906      C     ROOT FOUND IS IMPLAUSIBLE(BECAUSE THE EIGENFUNCTION CHANGES TOO DRASTICALLY
000907      C     OR OMEGA CHANGES TOO MUCH).THE INCREMENT IS INCREASED IF THE ROOT IS
000908      C     FOUND VERY CLOSE TO THE STARTING POINT (LESS THAN ERRMIN) AND THERE IS VERY
000909      C     LITTLE CHANGE IN THE EIGENFUNCTION, BUT THE RELATIVE INCREMENT DK(=DELT2/
000910      C     KYAI) IS NEVER MORE THAN DKMAX. IF DK FALLS BELOW DKMIN, WE GIVE UP
000911      C     FOLLOWING THIS ROOT AND RETURN.
000912            EXTERNAL DISP
000913            LOGICAL SW(6), SW7, CV(2), WARN, DEJAVU, PHIZ(200), PHIZP,ZEH(100)
000914            LOGICAL REALZ, FLIP, GOOD(18), PAIR,AOOT,SW71,SW31
000915            LOGICAL CMPPR, CMCRIT, FLIPC
000916            LOGICAL UNMAG
000917            COMPLEX              P1,P2,P3,ROOT,FROOT,FUNC,H,YO,Y1,Y2,A,B,C,Y3
000918            COMPLEX YY(2), YINIT
000919            COMPLEX PHIOLD(20), PHINEW(20), PHIVLD(20)
000920            COMPLEX YSTART
000921            COMPLEX YREV
000922            INTEGER EIGV,PMAX
000923            REAL KYAI, KYSQ, KOSQ
000924            DIMENSION X(2000),Y(2000),XC(1000),YR(1000),YI(1000),XZ(4000),
000925           .YZ(4000)
000926            COMMON/CIO/ ICR,IHSP,IGRAPH
000927            COMMON/CFAST/DK1,FKYAI1,  NPTC,YI,YR,XC,              KYD,     YMAX,
000928           .YZ,XZ,Y,X,NPT,NPTZ,ZEH,IGV,AOOT,NFOL,FKYAI2,YMIN
000929            COMMON/CMRAF/SW,RTC,ITC,EP3,FRMAX,XRMAX,H,ROOT,FUNC,FROOT,EIGV,
000930           .SW7,START,RMX,NFL,CYAI
000931            COMMON/DIS/QF,KYSQ,KOSQ,EPSF(4),     MASS,KYAI,KOAI,MASQ,PMAX,OMPSQ
000932           .,AEAI,KMAX(4),AMASQ(4),EPS(4),KMIN(4)
000933            COMMON/CPHCK/ PHINEW,PHZP
000934            COMMON/PMX/NP
000935            COMMON/FOLEIG/ GOOD, IBRANCH(18)
000936            COMMON/CMRAF2/ ERRMAX,NRTSU,SW31,SW71
000937            COMMON/CFOL/YSTART(100),XSTART(100),NSTART,YEND(100),XEND(100),
000938           .NSTRT1
000939            COMMON/CCHECK/XCRIT(100),YCRIT(100),NCRIT,DEJAVU,Y2,N,PZCRIT(100)
000940            COMMON/CCMP/ CMPPR, CMCRIT(100)
000941            COMMON/CPHZ/ PHIZ
000942            COMMON/CQN/MINQ
000943            COMMON/CMAG/ UNMAG
000944            DATA GOOD/18*.FALSE./
000945            DATA FRAMES/3./
000946            DATA AOOT/.FALSE./
000947            DATA NSTART/0/
000948            DATA NCRIT/0/
000949            DATA NGROW /0/
000950            DATA FKYAI2/25./
000951            DATA SUM2/0./
000952            CALL TIMECHK(AOOT,FRAMES,ELAPSED)
000953            IF(AOOT) RETURN
000954      C     FKYAI1 IS THE VALUE OF FKYAI READ IN. FKYAI2 IS A SMALLER VALUE USED
000955      C     FOR REAL ROOTS WHICH WE ARE NOT INTERESTED IN FOLLOWING SO FAR.
000956            FKYAI = FKYAI2
000957            IF(XO.GE.FKYAI2) FKYAI = FKYAI1
000958            IF(FKYAI2.GT.FKYAI1) FKYAI = FKYAI1
000959            BASE = .5*XO
000960            BASE2 = .25*FKYAI + XO
000961            PHZP = SGNL(1.,PHIZP)
000962            FLIP = .FALSE.
000963            FLIPC = .FALSE.
000964            PAIR = .FALSE.
000965            DK = DK1
000966            ERRMAX = .003
000967            ERRMIN = .0005
000968            SUM2MX = NP*.01
000969            SPREAD = .001
000970            YINIT = YO
000971            DKMAX = DK1
000972            DKMIN = 1.E-10
000973            IF(SW31) GO TO 110
000974            IF(AIMAG(YO).LE.0.) GO TO 15
000975            FKYAI = FKYAI1
000976            NGROW = NGROW + 1
000977            PRINT 55, NGROW
000978            GO TO 15
000979          110 IF(REALZ(YO)) GO TO 15
000980            FKYAI = FKYAI1
000981            NCRIT = NCRIT + 1
000982            PRINT 55, NCRIT
000983           55 FORMAT(13H GROWING ROOT,I4,7H BEGINS)
000984      C     IF THE ROOT IS ALREADY COMPLEX AT INITIAL KYAI, GIVE HARMLESS NONSENSE
000985      C     VALUES TO XCRIT, YCRIT, AND PZCRIT (USED BY *CHECK*) FOR THIS ROOT.
000986            PZCRIT(NCRIT) = 0.
000987            YCRIT(NCRIT) = -1.
000988            XCRIT(NCRIT) = -1.
000989            CMCRIT(NCRIT) = .TRUE.
000990           15 IF(.NOT.SW71) GO TO 134
000991            YREV = CMPLX(AIMAG(YO),REAL(YO))
000992            IF(REALZ(YREV)) GO TO 134
000993            NCRIT = NCRIT + 1
000994            PRINT 135, NCRIT
```

```
000995       135 FORMAT(13H  MOVING ROOT,I4,7H BEGINS)
000996           PZCRIT(NCRIT) = 0.
000997           YCRIT(NCRIT) = -1.
000998           XCRIT(NCRIT) = -1.
000999           CMCRIT(NCRIT) = .FALSE.
001000     C    FIND FIRST AND SECOND DERIVATIVES OF ROOT WITH RESPECT TO KYAI
001001       134 DELT1 = 1.E-05*XO
001002           DELT2 = 1.E-04*XO
001003           IF(SW31.OR.NFOL.LE.NSTRT1) GO TO 116
001004           DELT1 = 1.E-10*XO
001005           DELT2 = 1.E-09*XO
001006           DK = 1.E-09 ;
001007       116 P3 = YO
001008           P2 = YO - (0.,.001)
001009           P1 = YO - (.001,0.)
001010        16 KYAI = XO + DELT1
001011           IF(WARN(0)) RETURN
001012           KYSQ = KYAI*KYAI
001013           CYAI = KYAI
001014           CHI = KYSQ + PMAX*PMAX*KOSQ
001015           MINQ = 2.*REAL(P3)
001016           CALL QINT(CHI)
001017           CALL QFKCAL(QF,KMAX,KMIN,P3+3.,P3)
001018           CALL MRAF(DISP,YY,CV,2,P1,P2,P3,1.E-06,1.E-12,25)
001019     C    IF NO PLAUSIBLE ROOT IS FOUND, REDUCE DELT1 AND TRY AGAIN.
001020           IF(.NOT.CV(1)) GO TO 29
001021           Y1 = YY(1)
001022           IF(.NOT.CV(2)) GO TO 59
001023           IF(PHZP.EQ.SGNL(1.,PHIZ(1)))GO TO 59
001024           IF(PHZP.NE.SGNL(1.,PHIZ(2)))GO TO 291
001025           Y1 = YY(2)
001026        59 IF(AIMAG(Y1).LT.-1.E-06.AND.SW31) GO TO 292
001027           IF(REAL(Y1).LT.-1.E-06.AND.SW71) GO TO 294
001028           GO TO 48
001029       290 IWHY =10 $ GO TO 29
001030       291 IWHY =11 $ GO TO 29
001031       292 IWHY =12 $ GO TO 29
001032       294 IWHY = 14 $ GO TO 29
001033        29 IF(DELT1.LT.1.E-10) GO TO 58
001034           DELT1 = DELT1/10.
001035           IF(SW(1)) PRINT 56, DELT1, IWHY
001036        56 FORMAT(17H DELT1 REDUCED TO, E22.13,11H FOR REASON,I4)
001037           IF(SW(1).AND.IWHY.EQ.1) PRINT 57,PHIZP,PHIZ(1)
001038        57 FORMAT(9H PHIZP IS,L5,11H PHIZ(1) IS,L5)
001039           GO TO 16
001040        48 IF(IGV.EQ.0) GO TO 68
001041           DO 31 I = 1,NP
001042           PHIVLD(I) = PHINEW(I)
001043        31 PHIOLD(I) = PHINEW(I)
001044        68 A = (0.,0.)
001045           B = (Y1 - YO)/DELT1
001046           C = YO + B*DELT1
001047           ERROR = .001
001048           Y2 = Y1
001049           GO TO 27
001050         8 IF(WARN(0)) RETURN
001051     C    EVEN IF EIGV WAS ORIGINALLY 3 OR 4, WE ONLY LET IT BE EQUAL TO ITS
001052     C    ORIGINAL VALUE EVERY SO OFTEN, OTHERWISE WE WILL BE PLOTTING A
001053     C    RIDICULOUS NUMBER OF EIGENFUNCTIONS.
001054           IF(IGV.EQ.0) GO TO 53
001055           IF((KYAI/BASE).LT.(1.+DKMAX)) GO TO 40
001056           EIGV = IGV
001057           GO TO 53
001058        40 EIGV = 2
001059        53 IF(KYAI.LT.BASE2) GO TO 41
001060           BASE2 = BASE2 + .25*FKYAI
001061     C    CHECK THE TIME EVERY SO OFTEN, TO MAKE SURE WE ARE NOT ALMOST OUT OF TIME
001062           CALL TIMECHK(AOUT,FRAMES,ELAPSED)
001063           PRINT 54, ELAPSED, KYAI
001064        54 FORMAT(5X,8H TIME IS,F8.3,10H AT KYAI =,F8.3)
001065           IF(AOUT) RETURN
001066        41 CALL QFKCAL(QF,KMAX,KMIN,P3+3.,P3)
001067           CALL MRAF(DISP,YY,   CV,2,P1,P2,P3,1.E-06,1.E-12,25)
001068           Y2 = YY(1)
001069           Y3 = YY(2)
001070           IF(.NOT.CV(1)) GO TO 90
001071     C    WE CHANGE LOGICAL VARIABLES TO REAL VARIABLES, SO WE CAN USE THEM
001072     C    IN IF(A.EQ.B) STATEMENTS.
001073           PHZ1 = SGNL(1.,PHIZ(1))
001074           PHZ2 = SGNL(1.,PHIZ(2))
001075           IF(.NOT.SW31) GO TO 121
001076           RL2 = SGNL(1.,REALZ(Y2))
001077           RL3 = SGNL(1.,REALZ(Y3))
001078           RLI = SGNL(1.,REALZ(YINIT))
001079       121 IF(.NOT.SW71) GO TO 120
001080           YREV = CMPLX(AIMAG(Y2),REAL(Y2))
001081           AIM2 = SGNL(1.,REALZ(YREV))
001082           YREV = CMPLX(AIMAG(Y3),REAL(Y3))
001083           AIM3 = SGNL(1.,REALZ(YREV))
001084           YREV = CMPLX(AIMAG(YINIT),REAL(YINIT))
001085           AIMI = SGNL(1.,REALZ(YREV))
001086       120 IF(.NOT.CV(2)) GO TO 14
001087     C    IF TWO PLAUSIBLE ROOTS, GO TO 20
001088           IF(PHZ1.EQ.PHZP.AND.PHZ2.EQ.PHZP) GO TO 20
001089     C    NO PLAUSIBLE ROOTS
001090           IF(PHZ1.EQ.PHZ2) GO TO 91
```

```
001091      C    ONE PLAUSIBLE ROOT AND ONE IMPLAUSIBLE ONE, THROW OUT THE LATTER.
001092             IF(PHZ1.EQ.PHZP) GO TO 14
001093             Y2 = Y3
001094             PHIZ(1) = PHIZ(2)
001095             PHZ1 = PHZ2
001096             GO TO 14
001097      C    IF ONE OF THE ROOTS IS MUCH CLOSER THAN THE OTHER TO THE PREDICTED R
001098      C    USE THAT ONE.
001099           20 IF(CABS2(Y2 - P3).LT..01*CABS2(Y3 - P3)) GO TO 14
001100             IF(.NOT.SW71) GO TO 117
001101             CMPPR = .FALSE.
001102             IF(AIM2.NE.AIM3) GO TO 122
001103             IF(AIM2.GT.0..AND.REAL(Y3)*REAL(Y2).LT.0.) GO TO 123
001104             IF(AIM1.GT.0..AND.AIM2.GT.0.) GO TO 117
001105             IF(AIM1.LT.0..AND.AIM2.LT.0.) GO TO 124
001106             IF(AIM2.GT.0.) GO TO 123
001107             IF(REAL(Y2).LT.0.) Y2 = CMPLX(0.,AIMAG(Y2))
001108             IF(REAL(Y3).LT.0.) Y3 = CMPLX(0.,AIMAG(Y3))
001109             GO TO 13
001110           123 IF(REAL(Y3).GT.0..AND.REAL(Y2).LT.0.) Y2 = Y3
001111      C    SEE IF WE HAVE A NEGATIVE CONJUGATE PAIR
001112             IF(CABS2(CONJG(Y3)+YY(1)).GT..01*CABS2(Y3-YY(1))) GO TO 126
001113             GO TO 127
001114      C    IN THE CASE OF AN IRREVERSIBLE DISP. REL., WE ASSUME THERE ARE NO
001115      C    MULTIPLE ROOTS, SO IF THERE ARE TWO EQUALLY PLAUSIBLE ROOTS, WE REDUCE
001116      C    DK AND TRY AGAIN.
001117           117 CMPPR = .TRUE.
001118             IF(.NOT.SW31) GO TO 100
001119      C    IF THE ROOT CHANGES FROM REAL TO COMPLEX OR VICE VERSA, WE DONT
001120      C    TRUST THE NEW ROOT UNLESS THERE ARE TWO OF THEM.
001121             IF(RL2.NE.RL3) GO TO 92
001122             IF(RL2.GT.0..AND.AIMAG(Y3).GT.0..AND.AIMAG(Y2).LT.0.) Y2 = Y3
001123             IF(RL1.EQ.RL2) GO TO 14
001124             IF(REALZ(Y2)) GO TO 13
001125      C    SEE IF WE HAVE A CONJUGATE PAIR.
001126             IF(CABS2(CONJG(Y3)-YY(1)).GT..01*CABS2(Y3-YY(1))) GO TO 93
001127      C    SEE IF EIGENFUNCTION CHANGES TOO MUCH, BUT SET PAIR = .TRUE. SO WE WILL
001128      C    RETURN HERE.
001129      C    IF THE TWO ROOTS, Y2 AND Y3, ARE CLOSE TO ZERO, AND THEY HAVE
001130      C    NOT SWITCHED FROM BEING PURELY REAL TO BEING PURELY IMAGINARY
001131      C    OR VICE VERSA, THEN WE DO NOT HAVE A NEW PAIR, BUT RATHER
001132      C    TWO ROOTS WHICH CROSS EACH OTHER AT OMEGA = 0 WITHOUT INTERRACTING.
001133           127 IF(AIMI.NE.RLI.AND.AIM2.LT.0..AND.RL2.LT.0.) GO TO 14
001134             IF(AIMI.NE.RLI.AND.AIMI.EQ.AIM2.AND.RLI.EQ.RL2.AND.P3.EQ.
001135            .5.E-04) GO TO 14
001136             PAIR = .TRUE.
001137             IF(IGV.GE.2) GO TO 36
001138            61 PAIR = .FALSE.
001139      C    IF WE REACH THE BEGINNING OF A  PAIR, CHECK TO SEE IF WE
001140      C    HAVE ARRIVED HERE PREVIOUSLY BY ANOTHER ROUTE.
001141             CALL CHECK
001142             IF(.NOT. DEJAVU) GO TO 23
001143             IF(CMPPR) PRINT 24, N
001144            24 FORMAT(29H WE HAVE REACHED GROWING ROOT, I4)
001145             IF(.NOT.CMPPR) PRINT 128, N
001146           128 FORMAT(29H WE HAVE REACHED  MOVING ROOT, I4)
001147             RETURN
001148            23 NCRIT = NCRIT + 1
001149             IF(NCRIT.GT.100) RETURN
001150      C    RECORDING THE BEGINNING OF A NEW GROWING (OR MOVING) ROOT.
001151             YINIT = Y2
001152             YCRIT(NCRIT) = REAL(Y2)
001153             IF(.NOT.CMPPR) YCRIT(NCRIT) = AIMAG(Y2)
001154             XCRIT(NCRIT) = KYAI
001155             PZCRIT(NCRIT) = PHZP
001156             CMCRIT(NCRIT) = CMPPR
001157      C    GET READY TO FOLLOW THE NEW GROWING (OR MOVING) ROOT.
001158             FKYAI = FKYAI1
001159             YO = Y2
001160             IF(CMPPR) DELT1 = .1*AIMAG(YO)**2
001161             IF(.NOT.CMPPR) DELT1 = .1*REAL(YO)**2
001162             DELT2 = DELT1
001163             P3 = YO
001164             P2 = YO + (0.,.1)*AIMAG(YO)
001165             P1 = YO - (0.,.1)*AIMAG(YO)
001166             XO = KYAI
001167             CALL TIMECHK(AOOT,FRAMES,ELAPSED)
001168             IF(CMPPR) PRINT 21, NCRIT,YCRIT(NCRIT),XCRIT(NCRIT),ELAPSED
001169            21 FORMAT(13H GROWING ROOT,I4,19H BEGINS AT OMEGA = ,E22.13,
001170            .8H KYAI = ,E22.13,5X,8H TIME IS,F8.3)
001171             IF(.NOT.CMPPR) PRINT 129,NCRIT,YCRIT(NCRIT),XCRIT(NCRIT),ELAPSED
001172           129 FORMAT(13H  MOVING ROOT,I4,19H BEGINS AT GAMMA = ,E22.13,
001173            .8H KYAI = ,E22.13,5X,8H TIME IS,F8.3)
001174             IF(AOOT) RETURN
001175             GO TO 16
001176            13 NSTART = NSTART + 1
001177      C    RECORD THE END OF A GROWING(OR MOVING) ROOT, AND ADD THE TWO NEW REAL
001178      C    (OR PURELY IMAGINARY) ROOTS TO THE LIST OF ROOTS TO BE FOLLOWED LATER.
001179      C    OR, IF THIS IS THE END OF A GROWING ROOT AND WE ARE USING AN IRREVERSIBLE
001180      C    DISPERSION RELATION, ADD THE DAMPED ROOT TO THE LIST.
001181             YSTART(NSTART) = Y2
```

```
001182            XSTART(NSTART) = KYAI
001183            ZEH(NSTART) = PHIZ(1)
001184            IF(.NOT.SW31.AND.AIMAG(Y1).GT.0..AND.CMPPR) GO TO 111
001185            NSTART = NSTART + 1
001186            YSTART(NSTART) = Y3
001187            XSTART(NSTART) = KYAI
001188            ZEH(NSTART) = PHIZ(2)
001189            IF(.NOT.SW31.AND.CMPPR) GO TO 115
001190         49 YEND(NCRIT) = REAL(Y2)
001191            IF(.NOT.CMPPR) YEND(NCRIT) = AIMAG(Y2)
001192            XEND(NCRIT) = KYAI
001193            IF(CMPPR) PRINT 22, NCRIT,YEND(NCRIT),XEND(NCRIT)
001194         22 FORMAT (13H GROWING ROOT,I4,17H ENDS AT OMEGA = ,E22.13,
001195           .8H KYAI = ,E22.13)
001196            IF(.NOT.CMPPR) PRINT 130,NCRIT,YEND(NCRIT),XEND(NCRIT)
001197        130 FORMAT (13H  MOVING ROOT,I4,17H ENDS AT GAMMA = ,E22.13,
001198           .8H KYAI = ,E22.13)
001199        111 RY2 = REAL(Y2)
001200            PRINT 22, NGROW, RY2, KYAI
001201        115 Y2 = Y1
001202            RETURN
001203      C     IF WE HAVE FOUND ONLY ONE ROOT, SEE IF HAS CHANGED TOO MUCH IN SOME WAY
001204      C     (E.G. FROM REAL TO COMPLEX OR VICE VERSA, OR IN ITS EIGENFUNCTION),
001205      C     AND IF SO, REDUCE DK.
001206         14 IF(AIM2.LT.0..AND.RL2.LT.0.) GO TO 132
001207            IF(.NOT.SW31) GO TO 131
001208            IF(RL2.NE.RL1) GO TO 94
001209        131 IF(.NOT.SW71) GO TO 132
001210            IF(AIM2.NE.AIM1) GO TO 133
001211        132 IF(REAL(Y2).LT.-1.E-06.AND.SW71) GO TO 95
001212            IF(AIMAG(Y2).LT.-1.E-06.AND.SW31) GO TO 96
001213            IF(PHZ1.NE.PHZP) GO TO 97
001214      C     FOR PHI(0) = 0 ROOTS, PHINEW IS NORMALIZED IN SUCH A WAY THAT PHI(-NP)
001215      C     (=PHINEW(1))  ALWAYS HAS POSITIVE REAL PART (SEE DEFINITION OF ARNM
001216      C     IN *MRAF*). IF PHI(-NP) HAS SWITCHED FROM -I TO +I, MULTIPLY PHIOLD
001217      C     BY -1 SO IT IS NORMALIZED IN THE SAME WAY AS PHINEW, AND WE CAN COMPARE
001218      C     THEM.
001219            IF(IGV.EQ.0) GO TO 11
001220            IF(.NOT.PHIZP) GO TO 36
001221            IF(AIMAG(PHINEW(1))*AIMAG(PHIOLD(1)).LT.0.) GO TO 35
001222            GO TO 36
001223         35 IF(ABS(AIMAG(PHINEW(1))).GT.ABS(REAL(PHINEW(1)))) GO TO 37
001224            GO TO 36
001225         37 DO 38 I = 1, NP
001226            PHIVLD(I) = -PHIVLD(I)
001227         38 PHIOLD(I) = -PHIOLD(I)
001228            FLIP = .TRUE.
001229         36 SUM2 = 0.
001230      C     SEE HOW MUCH THE EIGENFUNCTION HAS CHANGED, AND, IF DESIRED, PRINT
001231      C     OUT THE COMPONENTS WHICH HAVE CHANGED DRASTICALLY.
001232            DO 30 I = 1, NP
001233            DIFF2 = CABS2(PHINEW(I)-PHIOLD(I)-(PHIOLD(I)-PHIVLD(I))*(DELT2/
001234           .DELT1))
001235            SUM2 = SUM2 + DIFF2
001236            IF(DIFF2.LE..01) GO TO 30
001237         33 IF(SW(1)) PRINT 34, I, PHINEW(I), PHIOLD(I), PHIVLD(I)
001238         34 FORMAT(8H PHINEW(,I2,5H ) = ,2E14.6,10H PHIOLD = ,4E14.6)
001239         30 CONTINUE
001240            IF(SW(1)) PRINT 62, SUM2
001241         62 FORMAT(7H SUM2 = ,E14.6)
001242      C     IF THE EIGEMFUNCTION HAS CHANGED TOO MUCH(I.E. IF SUM2 IS TOO LARGE)
001243      C     REDUCE DK AND TRY AGAIN.
001244            IF(SUM2.GT.SUM2MX) GO TO 63
001245            IF(PAIR) GO TO 61
001246            FLIP = .FALSE.
001247            IF(.NOT.FLIPC) GO TO 11
001248            FLIPC = .FALSE.
001249            Y1 = -Y1
001250            YO = -YO
001251            GO TO 11
001252         63 IF(PAIR) GO TO 103
001253      C     SOMETIMES PHI(0) PASSES THROUGH ZERO EVEN FOR A PHI(0).NE.0 ROOT,
001254      C     IN WHICH CASE PHINEW AND PHIOLD WILL BE NORMALIZED DIFFERENTLY. SEE
001255      C     IF THIS IS THE CASE.
001256            IF(FLIPC) GO TO 72
001257            IF(.NOT.FLIP.AND..NOT.PHIZP) GO TO 37
001258            IF(FLIP) GO TO 75
001259      C     SOMETIMES TWO ROOTS PASS THROUGH EACH OTHER AT OMEGA = 0 WITHOUT
001260      C     INTERRACTING, IN WHICH CASE THE PHI(I) OF ONE ROOT IS THE COMPLEX
001261      C     CONJUGATE OF THE PHI(I) OF THE OTHER ROOT. IF THIS IS THE CASE,
001262      C     SWITCH TO FOLLOWING THE OTHER ROOT.
001263         74 IF( SW31.AND.SW71.AND.CABS2(Y2).LT.1.E-06)GO TO 70
001264            GO TO 98
001265      C     IF ROOT IS NO GOOD AND WE HAVE MULTIPLIED PHIOLD BY -1, MULTIPLY IT
001266      C     BACK AGAIN.
001267         75 DO 39 I = 1,NP
001268            PHIVLD(I) = -PHIVLD(I)
001269         39 PHIOLD(I) = -PHIOLD(I)
001270            FLIP = .FALSE.
001271            GO TO 74
001272         70 DO 71 I = 1,NP
001273            PHIVLD(I) = CONJG(PHIVLD(I))
001274         71 PHIOLD(I) = CONJG(PHIOLD(I))
001275            FLIPC = .TRUE.
001276            GO TO 36
```

```
001277     C    IF ROOT IS NO GOOD AND WE HAVE REPLACED PHIOLD BY ITS COMPLEX
001278     C    CONJUGATE, REPLACE IT BACK AGAIN.
001279          72 DO 73 I = 1,NP
001280             PHIVLD(I) = CONJG(PHIVLD(I))
001281          73 PHIOLD(I) = CONJG(PHIOLD(I))
001282             FLIPC = .FALSE.
001283             GO TO 136
001284          11 ERROR = CABS(P3 - Y2)
001285     C    IF THE ERROR IN ESTIMATING THE LAST ROOT(VIZ. P3 - Y2) IS VERY SMALL AND
001286     C    THE EIGENFUNCTION HAS NOT CHANGED TOO MUCH, INCREASE DK.
001287             IF(ERROR.GT.ERRMIN) GO TO 12
001288             IF(SUM2.GT..01) GO TO 12
001289             DK3 = DKMAX
001290             DK2 = DKMAX
001291             IF((ERROR/ERRMIN).GT.(DK/DKMAX)**3) DK3 = DK*(ERRMIN/ERROR)**(1./
001292            .3.)
001293             IF((SUM2/.01).GT.(DK/DKMAX)**2)DK2 = DK*SQRT(.01/SUM2)
001294             DK4= AMIN1(DK3,DK2)
001295     C    DON'T LET DK INCREASE TOO RAPIDLY, OR WE MAY GET ON THE WRONG ROOT.
001296             IF(DK4.GT.10.*DK) DK4= 10.*DK
001297             DK = DK4
001298             IF(SW(1)) PRINT 28, DK
001299          28 FORMAT(16H DK INCREASED TO , E12.6)
001300     C    IF  THE NEW ROOT Y2 IS OUTSIDE THE  PRESENT FREQUENCY RANGE, EXPAND THE
001301     C    RANGE TO INCLUDE IT(INCREASING YMAX OR DECREASING YMIN) AND CALCULATE
001302     C    VALUES OF QF, KMAX, AND KMIN (USED IN *DISP*).
001303          12 IF(UNMAG.AND..NOT.REALZ(Y2).AND.AIMAG(Y2).LT.0.) GO TO 105
001304             IF(ABS(REAL(Y2)).LE.YMAX) GO TO 104
001305             YMAX = ABS(REAL(Y2))
001306             GO TO 105
001307         104 IF(ABS(REAL(Y2)).GE.YMIN) GO TO 106
001308             YMIN = ABS(REAL(Y2))
001309         105 CONTINUE
001310         106 IF(IGV.EQ.0) GO TO 69
001311          64 DO 32 I = 1,NP
001312             PHIVLD(I) = PHIOLD(I)
001313          32 PHIOLD(I) = PHINEW(I)
001314     C    MAKE A NEW PARABOLIC FIT.
001315          69 A = Y2/(DELT2*SUM) - Y1/(DELT1*DELT2) + Y0/(DELT1*SUM)
001316             B = (Y1 - Y0)/DELT1 + A*DELT1
001317             C = Y0 - A*DELT1*DELT1 + B*DELT1
001318     C    LET *EIGPLT* KNOW THAT THIS IS A GOOD ROOT, SO IT WILL PLOT THE EIGE
001319             IF(EIGV.LE.2) GO TO 67
001320             IF(EIGV.EQ.4) GOOD(NRTSU) = .TRUE.
001321             IF(EIGV.EQ.3.AND.AIMAG(Y2).GE.1.E-05) GOOD(NRTSU) = .TRUE.
001322             IF(EIGV.GE.3.AND.GOOD(NRTSU)) GO TO 66
001323             GO TO 67
001324          66 BASE = KYAI
001325     C    KEEP TRACK OF THE NUMBER OF FRAMES TO BE PLOTTED, SINCE THIS IS USED
001326     C    BY *TIMECHK*.
001327             FRAMES = FRAMES + 2.
001328             IBRANCH(NRTSU) = NFOL
001329          67 IF(SW(1)) PRINT 60, EIGV,NRTSU,GOOD(NRTSU),BASE
001330          60 FORMAT(7H EIGV =,I4,8H NRTSU =,I4,I5,10X,8H BASE  =,E22.13)
001331     C    PRINT OUT INFORMATION ON ROOT, IF DESIRED.
001332             X1 = X0 + DELT1
001333             IF(SW(1)) PRINT 7, A, B, C, X1
001334           7 FORMAT(43H OMEGA = A*DELT**2 + B*DELT + C, WHERE A = ,2F10.5/38X,
001335            .5H B = ,2F10.5/38X,5H C = ,2F10.5,14H DELT = KYAI -,F15.10)
001336             IF(SW31) GO TO 107
001337     C    IF THE DISPERSION RELATION IS IRREVERSIBLE, AND THE GROWTH RATE HAS
001338     C    CHANGED SIGN, TAKE APPROPRIATE MEASURES, SINCE WE WANT  TO DISTINGUISH
001339     C    GROWING ROOTS FROM DAMPED ROOTS IN THE PLOT.
001340             IF(AIMAG(Y1    ).LE.0..AND.AIMAG(Y2).GT.0.) GO TO 108
001341             CMPPR = .TRUE.
001342             IF(AIMAG(Y1    ).GT.0..AND.AIMAG(Y2).LE.0.) GO TO 13
001343             IF(AIMAG(Y2).GT.0.) GO TO 17
001344             GO TO 109
001345         108 CALL TIMECHK(ADOT,FRAMES,ELAPSED)
001346             NGROW = NGROW + 1
001347             RY2 = REAL(Y2)
001348             PRINT 21, NGROW, RY2, KYAI, ELAPSED
001349             GO TO 17
001350         107 IF(.NOT.REALZ(Y2)) GO TO 17
001351         109 IF(PHIZ(1)) GO TO 18
001352             IF(NPT .EQ.0) GO TO 52
001353     C    DON'T BOTHER RECORDING ROOT FOR PLOTTING UNLESS IT IS A REASONABLE DISTANCE
001354     C    FROM PREVIOUS ROOT.
001355             DGRPH= ((KYAI-X(NPT))/FKYAI)**2 + ((REAL(Y2)-Y(NPT))/YMAX)**2
001356             IF(DGRPH.LT.3.E-05) GO TO 19
001357     C    RECORD REAL ROOT, PHIZ = .FALSE.
001358          52 NPT = NPT + 1
001359             Y(NPT) = REAL(Y2)
001360             X(NPT) = KYAI
001361             IF(NPT.GE.2000) RETURN
001362             GO TO 19
001363          18 IF(NPTZ.EQ.0) GO TO 51
001364             DGRPH=((KYAI-XZ(NPTZ))/FKYAI)**2+((REAL(Y2)-YZ(NPTZ))/YMAX)**2
001365             IF(DGRPH.LT.4.E-06) GO TO 19
001366     C    RECORD REAL ROOT, PHIZ = .TRUE.
001367          51 NPTZ = NPTZ + 1
001368             YZ(NPTZ) = REAL(Y2)
001369             XZ(NPTZ) = KYAI
001370             IF(NPTZ.GE.4000) RETURN
001371             GO TO 19
```

```
001372        17 IF(NPTC.EQ.0) GO TO 50
001373           IF(ABS(AIMAG(Y2) - YI(NPTC)).GT..01) GO TO 50
001374           DGRPH=((KYAI-XC(NPTC))/FKYAI)**2+((REAL(Y2)-YR(NPTC))/YMAX)**2
001375           IF(DGRPH.LT.4.E-06) GO TO 19
001376     C    RECORD COMPLEX ROOT
001377        50 NPTC = NPTC + 1
001378           YR(NPTC) = REAL(Y2)
001379           YI(NPTC) = AIMAG(Y2)
001380           XC(NPTC) = KYAI
001381           IF(NPTC.GE.1000) RETURN
001382        19 YO = Y1
001383           Y1 = Y2
001384           XO = XO + DELT1
001385           DELT1 = DELT2
001386           DELT2 = DK*KYAI
001387     C INCREASE KYAI AND PREDICT WHERE ROOT WILL BE (P3).
001388        27 KYAI = KYAI + DELT2
001389           IF(KYAI.GT.FKYAI) RETURN
001390           SUM = DELT1 + DELT2
001391           P3 = A*SUM*SUM + B*SUM + C
001392           IF(SW71.AND.SW31.AND.CABS2(P3).LT.1.E-06) P3 = .0005
001393     C     IF THE PREDICTED NEW ROOT IS TOO FAR FROM THE OLD ROOT, EITHER RELATIVELY
001394     C     OR ABSOLUTELY, THE PREDICTION IS UNRELIABLE, SO REDUCE DK AND TRY AGAIN.
001395           IF(CABS2(Y2-P3).GT.1.) GO TO 45
001396           IF(CABS2(Y2 - P3).LT.1.E-06) GO TO 26
001397           IF(REAL(P3).EQ.0..AND.AIMAG(P3).EQ.0.) GO TO 45
001398           IF(CABS2(Y2/P3 -(1.,0.)).LT..04)GO TO 26
001399        45 KYAI = KYAI - DELT2
001400           DELT2 = .5*DELT2
001401           DK = DELT2/KYAI
001402           IF(SW(1)) PRINT 42, DK
001403        42 FORMAT(15H DK REDUCED TO , E12.6,41H BECAUSE Y2 IS RELATIVELY TOO
001404          .FAR FROM P3    )
001405           IF(DK.LT.DKMIN) GO TO 46
001406           GO TO 27
001407        26 SPREAD = AMAX1(ERROR,.001)
001408           P2 = P3 - (0.,1.)*SPREAD
001409           P1 = P3 - (1.,0.)*SPREAD
001410           CYAI = KYAI
001411           KYSQ = KYAI*KYAI
001412           CHI = KYSQ + PMAX*PMAX*KOSQ
001413           MINQ = 2.*REAL(P3)
001414     C     SUBROUTINE *QINT* PRINTS OUT THE NEW KYAI IF DESIRED AND CALCULATES
001415     C THE PARAMETERS QMAX AND NMAX USED IN THE DISPERSION FUNCTION CALCULATION
001416           CALL QINT(CHI)
001417           IF(SW(1).AND..NOT.SW(2)) PRINT 65, P3
001418        65 FORMAT(5H P3 =,2E22.13)
001419           GO TO 8
001420     C     IF DK HAS BEEN REDUCED, PRINT THIS FACT(IF DESIRED) AND TELL WHY. THIS
001421     C     INFORMATION IS INVALUABLE FOR DEBUGGING.
001422        90 IWHY = 0 $ GO TO 9
001423        91 IWHY = 1 $ GO TO 9
001424        92 IWHY = 2 $ GO TO 9
001425        93 IWHY = 3 $ GO TO 9
001426        94 IWHY = 4 $ GO TO 9
001427        95 IWHY = 5 $ GO TO 9
001428        96 IWHY = 6 $ GO TO 9
001429        97 IWHY = 7 $ GO TO 9
001430        98 IWHY = 8 $ GO TO 9
001431        99 IWHY = 9 $ GO TO 9
001432       100 IWHY = 100$ GO TO 9
001433       122 IWHY = 122 $ GO TO 9
001434       124 IWHY = 124 $ GO TO 9
001435       126 IWHY = 126 $ GO TO 9
001436       133 IWHY = 133 $ GO TO 9
001437       136 IWHY = 136 $ GO TO 9
001438       103 PAIR = .FALSE.
001439           IWHY = 13 $ GO TO 9
001440         9 KYAI = KYAI - DELT2
001441           DK = .5*DK
001442           IF(DK.LT.DKMIN) GO TO 46
001443           DELT2 = DK*KYAI
001444           IF(SW(1)) PRINT 10, DK, IWHY
001445        10 FORMAT(15H DK REDUCED TO , E12.6,11H FOR REASON,I4)
001446           IF(.NOT.SW(1)) GO TO 27
001447           IF(IWHY.EQ.1.OR.IWHY.EQ.7)                PRINT 43,PHZP,PHIZP,PHZ1,
001448          .PHIZ(1),PHZ2,PHIZ(2)
001449        43 FORMAT(8H PHZP = ,E22.13,L5/8H PHZ1 = ,E22.13,L5/8H PHZ2 = ,E22.13
001450          .,L5)
001451           GO TO 27
001452        58 Y2 = Y1
001453     C     IF DK OR DELT1 IS LESS THAN DKMIN, GIVE UP ON THIS ROOT.
001454        46 PRINT 47, IWHY
001455        47 FORMAT(" DK IS LESS THAN DKMIN, FOR REASON ",I4/" EITHER
001456          .QMAX AND NMAX (IN SUBROUTINES QINT AND QCALC) ARE TOO SMALL,"
001457          ./" OR ZEE IS NOT ACCURATE ENOUGH, OR WE HAVE ENCOUNTERED A TOPO-"
001458          ./"LOGICAL SITUATION WHICH THE PROGRAM CANNOT HANDLE")
001459           IF(IWHY.NE.100) RETURN
001460     C     WE HAVE REACHED A CRITICAL POINT, IN THE IRREVERSIBLE CASE. CHECK IF
001461     C     WE'VE BEEN HERE BEFORE. IF NOT, GO JUST PAST IT AND USE THE TWO NEW
001462     C     ROOTS AS STARTING POINTS.
001463           CALL CHECK
001464           IF(.NOT.DEJAVU) GO TO 112
001465           PRINT 113, N
001466       113 FORMAT(31H WE HAVE REACHED CRITICAL POINT, I4)
```

```
001467          RETURN
001468        112 KYAI = KYAI +  10.*DELT2/((DELT1/DELT2)*CABS((Y1-Y2)/(Y0-Y1))-1.)
001469          KYSQ = KYAI*KYAI
001470          CYAI = KYAI
001471          CALL MRAF(DISP,YY,CV,2,P1,P2,P3,1.E-06,1.E-12,25)
001472          IF(.NOT.CV(1).OR..NOT.CV(2)) RETURN
001473          IF(SGNL(1.,PHIZ(1)).NE.PHZP.OR.SGNL(1.,PHIZ(2)).NE.PHZP) RETURN
001474          Y2 = YY(1)
001475          Y3 = YY(2)
001476          NCRIT = NCRIT + 1
001477          IF(NCRIT.GT.100) RETURN
001478          YCRIT(NCRIT) = REAL(Y2)
001479          XCRIT(NCRIT) = KYAI
001480          PZCRIT(NCRIT) = PHZP
001481          PRINT 114, NCRIT,Y2, KYAI
001482        114 FORMAT(42H WE HAVE REACHED A NEW CRITICAL POINT, NO.,I4,12H AT OME
001483         .GA = ,2E22.13,8H KYAI = ,E22.13)
001484          GO TO 13
001485          END
```

```
001486
001487            SUBROUTINE TIMECHK(AOOT,FRAMES,ELAPSD)
001488      C     CHECKS TO SEE IF WE ARE ALMOST OUT OF TIME, MAKES SURE THERE WILL BE
001489      C     ENOUGH CUS LEFT OVER AFTER EXECUTION TO MAKE CRT PLOTS. FRAMES IS
001490      C     NUMBER OF FRAMES TO BE PLOTTED(EMPIRICALLY, 12 FRAMES IN THIS PROGRA
001491      C     TAKE ABOUT 1 CU.),AOOT IS SET = .TRUE. IF WE ARE ALMOST OUT OF TIME,
001492      C     ELAPSED IS ELAPSED TIME IN SECONDS.
001493      C     ONE CU (AS USED AT LBL) IS ABOUT 0.25 SECONDS ON THE CDC7600.
001494            LOGICAL UNMAG
001495            COMMON/CMAG/ UNMAG
001496            COMMON/PMX/NPP
001497            LOGICAL AOOT
001498            DATA ITIM1/0/
001499            DATA ELAPSED/0./
001500            CALL OOTIM(ITIM1,ITIM2,ITIM3,ITIM4)
001501            ELAPSED = ELAPSED +FLOAT(ITIM1)*1.E-06
001502            LEFT = 4.E-06*FLOAT(ITIM2)
001503            NEED = 30. + 15.*FLOAT(NPP) + FRAMES/12.
001504            IF(UNMAG) NEED = NEED - 6.*FLOAT(NPP)
001505            IF(LEFT.LT.NEED) AOOT = .TRUE.
001506            ELAPSD = ELAPSED
001507            RETURN
001508            END
```

```
001509
001510            SUBROUTINE CHECK
001511      C     CHECKS TO SEE IF A COMPLEX ROOT HAS ALREADY BEEN REACHED BY ANOTHER ROUTE.
001512      C     IF IT HAS, WE SET DEJAVU = .TRUE.
001513            COMPLEX PHINEW(20)
001514            LOGICAL DEJAVU
001515            LOGICAL CMCRIT, CMPPR
001516            REAL KYAI
001517            COMPLEX Y2
001518            COMMON/CCHECK/XCRIT(100),YCRIT(100),NCRIT,DEJAVU,Y2,N,PZCRIT(100)
001519            COMMON/CPHCK/ PHINEW,PHZP
001520            COMMON/CCMP/ CMPPR, CMCRIT(100)
001521            COMMON/DIS/QF,KYSQ,KOSQ,EPSF(4),     MASS,KYAI,KOAI,MASQ,PMAX,OMPSQ
001522           .,AEAI,KMAX(4),AMASQ(4),EPS(4),KMIN(4)
001523            IF(NCRIT.LE.0) GO TO 3
001524            DO 1 I = 1, NCRIT
001525            IF(PZCRIT(I).NE.PHZP) GO TO 1
001526      C     THERE ARE TWO TYPES OF CRITICAL POINTS, CHARACTERIZED BY CMPPR = .TRUE. AND
001527      C     CMPPR = .FALSE.,.  THE FORMER CONSIST OF TWO REAL ROOTS TURNING INTO
001528      C   A CONJUGATE PAIR. THE LATTER CONSIST OF TWO PURELY IMAGINARY ROOTS
001529      C     TURNING INTO A NEGATIVE CONJUGATE PAIR.
001530            IF(SGNL(1.,CMCRIT(I)).NE.SGNL(1.,CMPPR)) GO TO 1
001531            IF(ABS(KYAI - XCRIT(I)).LT..01*KYAI) GO TO 2
001532          1 CONTINUE
001533          3 DEJAVU = .FALSE.
001534            RETURN
001535          2 Y = REAL(Y2)
001536            IF(.NOT.CMPPR) Y = AIMAG(Y2)
001537            IF(ABS(  Y       - YCRIT(I)).GT.:005) GO TO 3
001538            N = I
001539            DEJAVU = .TRUE.
001540            RETURN
001541            END
```

```
001542
001543            LOGICAL REALZ
001544            FUNCTION REALZ(Z)
001545            COMPLEX Z
001546            REALZ = .TRUE.
001547            PREC = 1.E-06
001548            IF(ABS(AIMAG(Z)).GT.PREC) REALZ = .FALSE.
001549            RETURN
001550            END
001551
001552
001553            SUBROUTINE QINT(CHI)
001554      C     INTERFACE BETWEEN *QCALC* AND *ROOTS* OR *FOLLOW*)
001555            INTEGER QMAX
001556            LOGICAL SW(6)
001557            REAL KYAI
001558            COMPLEX H, ROOT, FUNC, FROOT
001559            COMMON/CMRAF/SW,RTC,ITC,EP3,FRMAX,XRMAX,H,ROOT,FUNC,FROOT,EIGV,
001560           .SW7,START,RMX,NFL,KYAI
001561            COMMON/CRBESJ/ QMAX(4),NMAX(4)
001562            COMMON/CSPEC/ RLARM(4),AMASS(4),CHARG(4),JAY(4),DENSE(4),NSPEC
001563            DO 1 I = 1, NSPEC
001564            IF(RLARM(I).EQ.0.) GO TO 1
001565            CHI2 = CHI*RLARM(I)*RLARM(I)
001566            IF(JAY(I).EQ.0)  GO TO 2
001567            CALL QCALC(CHI2,QMAX(I),NMAX(I),1)
001568            GO TO 3
001569          2 CALL QCALC(CHI2,QMAX(I),NMAX(I),0)
001570            NMAX(I) = NMAX(I) + JAY(I)
001571          3 IF(SW(1)) PRINT 7,KYAI,QMAX(I),NMAX(I),I
001572          7 FORMAT(7H KYAI =,E22.13,8H QMAX = ,I4,8H NMAX = ,I4,12H FOR SPECIE
001573           .S,I3)
001574          1 CONTINUE
001575            RETURN
001576            END
001577
001578            SUBROUTINE PARAM(N1)
001579      C     THIS ROUTINE WRITES A LIST OF THE PARAMETERS ON PRINTER(N1=99) OR CRT(N1=98)
001580            INTEGER PMAX,RMAX,TYPE
001581            REAL LINE
001582            LOGICAL SINGLX, UNMAG
001583            COMMON/CIO/ ICR,IHSP,IGRAPH
001584            COMMON/CBETA/ BETA
001585            COMMON/CPARAM/ RMAX,AIL,EPSI,EPSE,NSPEC ,DFUI,DFUE,EP,SINGLX,XL(10
001586           .),NXL
001587            COMMON/CRGAM/R,GAM,RE,GAME,R1,RE1,AEAI1
001588            COMMON/CSPEC/ RLARM(4),AMASS(4),CHARG(4),JAY(4),DENSE(4),XXXX
001589            COMMON/CDFU/ DFU(4),DREL(4)
001590            COMMON/TVTUNE/LPENON,LPENOFF,ITALICS,IWINK,INTENSE,IRIGHT,IUP
001591            COMMON/DIS/QF,KYSQ,KOSQ,EPSF(4),     MASS,KYAI,KOAI,MASQ,PMAX,OMPSQ
001592           .,AEAI,KMAX(4),AMASQ(4),EPS(4),KMIN(4)
001593            COMMON/CMAG/UNMAG
001594            IF(N1.EQ.98) N = IGRAPH
001595            IF(N1.EQ.99) N = IHSP
001596            TYPE = 6H LOCAL
001597            IF(PMAX.GT.0) TYPE = 6H
001598            IF(UNMAG) TYPE = 6H UNMAG
001599            IF(NSPEC.GT.0) GO TO 36
001600            IF(N.EQ.IGRAPH) CALL SETCH(1.,6.,1,0,1,0,0)
001601      C     PMAX = 0 MEANS WE ARE USING THE LOCAL APPROXIMATION
001602            IF(PMAX.EQ.0) GO TO 4
001603            WRITE(N,3) TYPE,PMAX,AIL,OMPSQ,EPSI,EPSE,MASS,AEAI,DFUI,DFUE
001604          3 FORMAT(A6, 6H PMAX=, I2, 5H AIL=, F6.3, 7H OMPSQ=,F8.1, 6H EPSI=,
001605           .F6.3, 6H EPSE=, F6.3/ 6H MASS=, F9.2, 7H AE/AI=, F7.4, 6H DFUI=,
001606           .F6.4,6H DFUE=, F6.4)
001607            GO TO 5
001608          4 IF(.NOT.SINGLX) GO TO 7
001609            WRITE(N,6) OMPSQ,EP,MASS,AEAI,DFUI,DFUE,BETA
001610          6 FORMAT(14H LOCAL APPROX.,9H OMPSQ = ,F9.3,8H  EPS  = ,F7.3,
001611           . 8H  MASS = ,F9.3/9H AE/AI = ,F8.4,7H DFUI =,F6.4,7H DFUE =,F6.4,
001612           .6H BETA=,F5.2)
001613            GO TO 5
001614          7 WRITE(N,8) TYPE,AIL,OMPSQ,MASS,AEAI,DFUI,DFUE,(XL(I),I=1,NXL)
001615          8 FORMAT(5H SINE,A6,5H AIL=,F7.4,7H OMPSQ=,F7.1,6H MASS=,F7.1,7H AE/
001616           .AI=,F5.3,6H DFUI=,F6.4/6H DFUE=,F6.4,5H X/L=,10F4.2)
```

```
001617          5 CONTINUE
001618            IF(R.GT.0.) GO TO 19
001619            WRITE(N ,30)
001620         30 FORMAT(26H RING DISTRIBUTION OF IONS)
001621            GO TO 23
001622         19 IF(GAM.GT.0.) GO TO 31
001623            WRITE(N ,32)
001624         32 FORMAT(16H MAXWELLIAN IONS)
001625            GO TO 23
001626         31 WRITE(N ,24) R,GAM
001627         24 FORMAT(16H MIRROR RATIO = ,F7.2,8H GAMMA= ,F6.3,9H FOR IONS)
001628         23 CONTINUE
001629            IF(AEAI.GT.0.) GO TO 25
001630            WRITE(N ,26)
001631         26 FORMAT(15H COLD ELECTRONS)
001632            GO TO 27
001633         25 IF(RE.GT.0.) GO TO 28
001634            WRITE(N ,29)
001635         29 FORMAT(31H RING DISTRIBUTION OF ELECTRONS)
001636            GO TO 27
001637         28 IF(GAME.GT.0.) GO TO 33
001638            WRITE(N ,34)
001639         34 FORMAT(21H MAXWELLIAN ELECTRONS)
001640            GO TO 27
001641         33 WRITE(N ,35) RE,GAME
001642         35 FORMAT(16H MIRROR RATIO = ,F7.2,9H GAMMA = ,F6.3,14H FOR ELECTRONS
001643           .)
001644         27 CONTINUE
001645            RETURN
001646         36 IF(N.EQ.IGRAPH) CALL SETCH(30.,7.,1,0,1,0,0)
001647            IF(PMAX.EQ.0.AND.SINGLX) WRITE (N,37) BETA
001648         37 FORMAT(22H LOCAL APPROX., BETA =,F5.2)
001649            IF(N.EQ.IGRAPH) CALL SETCH(1.,6.,1,0,1,0,0)
001650            IF(PMAX.EQ.0.AND..NOT.SINGLX) WRITE(N,41) TYPE,AIL,EPSI,EPSE,
001651           .(XL(I),I=1,NXL)
001652         41 FORMAT(5H SINE,A6,5H AIL=,F7.4,6H EPSI=,F6.3,6H EPSE=,F6.3/5H X/L=
001653           .,10F4.2)
001654            IF(PMAX.GT.0) WRITE (N,40) PMAX,AIL,      EPSI,EPSE
001655         40 FORMAT(6H PMAX=,I3,5H AIL=,F7.4,6H EPSI=,F6.3,6H EPSE=,F6.3)
001656            WRITE(N,42)
001657         42 FORMAT("SPECIES GYRORADIUS MASS  DIFFUSION   J   DENSITY  DENS.GRA
001658           .D.  OMPSQ  GYROFREQ.")
001659            NSPEC1 = MINO(NSPEC,4)
001660            IF(N1.EQ.99) NSPEC1 = NSPEC
001661            DO 38 I = 1, NSPEC1
001662            WP2 = OMPSQ*DENSE(I)*CHARG(I)**2/AMASS(I)
001663            WC = CHARG(I)/AMASS(I)
001664            WRITE(N,39) I,RLARM(I),AMASS(I),DFU(I),  JAY(I),DENSE(I),EPS(I),
001665           .WP2,WC
001666         39 FORMAT(I5,3E10.2,I5,4E10.2)
001667         38 CONTINUE
001668            RETURN
001669            END
```

```
001670
001671              SUBROUTINE RHCALC(R,GAM,KOA,RHE)
001672        C     CALCULATES THE RATIO OF THE ACTUAL PARTICLE DENSITY TO THE GUIDING
001673        C     CENTER DENSITY, AND STORES IT IN RHE. R IS THE MIRROR RATIO (R.LT.0
001674        C     IS USED TO INDICATE A RING DISTRIBUTION), GAM IS THE DEGREE TO
001675        C     WHICH THE LOSS CONE IS EMPTY, AND KOA IS KO*LARMOR RADIUS.
001676              REAL KOA,KOSQ,KOSQR,J2(100)
001677              KOSQ = KOA*KOA
001678              IF(R.GT.0.) GO TO 1
001679        C     RING DISTRIBUTION
001680              CALL QCALC(KOSQ,N,N,1)
001681              CALL BESSJ(KOA,N,J2)
001682              RHE = J2(1)
001683              RETURN
001684        C     MAXWELLIAN OR LOSS-CONE DISTRIBUTION
001685            1 RHE = EXP(-.5*KOSQ)
001686              IF(GAM.GT.0.) GO TO 2
001687              RETURN
001688        C     LOSS-CONE DISTRIBUTION
001689            2 KOSQR = KOSQ/R
001690              RHE = (R*RHE - GAM*EXP(-.5*KOSQR))/(R - GAM)
001691              RETURN
001692              END
```

```
001693
001694          COMPLEX DISP
001695          FUNCTION DISP (OMEGA)
001696      C   THIS ROUTINE CALCULATES THE DISPERSION FUNCTION
001697          COMPLEX     OMEGA,OMSQ,DETC,DO,D1,A(20,20),AP,BP,APP,BPP,
001698         1  DEPOLE,SUM1,SUM2,DIV,DENOM,DN,PHI(18,22),OMEGA1,W, OMEGD
001699          COMPLEX MAXW2,RING2,TERM(4)
001700          INTEGER P,PMAX,QMAX
001701          REAL MASS,KOAI,KOSQ,KYAI,KYSQ,MASQ,KPSQ,KXSQ
001702          LOGICAL DTCALC
001703          LOGICAL UNMAG
001704          COMPLEX ARH(20,20,4)
001705          COMMON/CBETA/ BETA
001706          COMMON/DIS/QF,KYSQ,KOSQ,EPSF(4),     MASS,KYAI,KOAI,MASQ,PMAX,OMPSQ
001707         .,AEAI,KMAX(4),AMASQ(4),EP(4),KMIN(4)
001708          COMMON /PHIC/ A,      NRTSV,  PHI
001709          COMMON/PMX/NPP
001710          COMMON/APBP/N,M,P
001711          COMMON/CCHIP/CHIP(4),AP(4),BP(4),     OMEGA1,APP(4),BPP(4)
001712          COMMON/CDETAL/ DTCALC, ARH
001713          COMMON/CSPEC/ RLARM(4),AMASS(4),CHARG(4),JAY(4),DENSE(4),NSPEC
001714          COMMON/CRBESJ/QMAX(4),NMAX(4)
001715          COMMON/CDFU/ DFU(4),DREL(4)
001716          COMMON/CMAG/UNMAG
001717          COMMON/CKXSQ/ KXSQ
001718          DATA DTCALC/.TRUE./
001719          DISP = (0.,0.)
001720          OMEGA1 = OMEGA
001721          OMSQ = OMEGA*OMEGA
001722      C   IF YOU ARE RIGHT ON A SINGULARITY, MOVE OFF IT=
001723         4 DO 103 I = 1, NSPEC
001724          OMEGD = OMEGA + (0.,1.)*KYSQ*DFU(I)
001725          IF(CABS2(OMEGD).LT.1.E-26.AND.KOAI.NE.0.) GO TO 100
001726          IF(CABS2(OMEGD).LT.1.E-26.AND.DFU(I).NE.0.) GO TO 100
001727          W = OMEGD*AMASS(I)/CHARG(I)
001728          AW = ABS(REAL(W))
001729          DW = 1.E-13*AW
001730          IF(ABS(AIMAG(W)).GT.DW) GO TO 103
001731          IW = IFIX(AW)
001732          IF(AW - IW.GT.DW.AND.IW + 1. - AW.GT.DW) GO TO 103
001733        100 OMEGA = OMEGA + 3.E-13*AMAX1(1.,REAL(OMEGA))
001734          OMEGA1 = OMEGA
001735          OMSQ = OMEGA*OMEGA
001736          GO TO 102
001737        103 CONTINUE
001738      C   DEPOLE IS USED TO GET RID OF ZEROS IN THE DENOMINATORS
001739        102 DEPOLE = (1.,0.)
001740          DO 2 I = 1, NSPEC
001741          OMEGD = OMEGA + (0.,1.)*KYSQ*DFU(I)
001742          IF(KOAI.EQ.0..AND.PMAX.GT.0) GO TO 23
001743          IF(EP(I).EQ.0..AND.PMAX.EQ.0) GO TO 23
001744          IF(I.EQ.1) GO TO 22
001745          IF(PMAX.EQ.0.AND.EP(I-1).EQ.0.) GO TO 22
001746          IF(DFU(I).EQ.DFU(I-1)) GO TO 23
001747         22 DEPOLE = OMEGD*DEPOLE
001748         23 IF(KMAX(I).EQ.0) GO TO 2
001749          KMX = KMAX(I)
001750          JI = KMIN(I)
001751          IF(PMAX.GT.0) GO TO 11
001752          DO 1 J = JI,KMX
001753          1 DEPOLE = DEPOLE*(OMEGD - J/AMASS(I))
001754          GO TO 2
001755         11 DO 12 J = JI, KMX
001756         12 DEPOLE = DEPOLE*(OMSQ - J*J/AMASQ(I))
001757          2 CONTINUE
001758      C   THE FACTOR QF IS PUT IN TO MAKE DEPOLE ON THE ORDER OF 1
001759          DEPOLE = DEPOLE*QF
001760          IF(PMAX.GT.0) DEPOLE = DEPOLE*QF
001761
001762          IF(PMAX.EQ.0) GO TO 10
001763      C   CALCULATE DISP USING  SINUSOIDAL DENSITY PROFILE IF PMAX.GT.0
001764          P=-PMAX-1
001765          M=1
001766          NP =   PMAX+1
001767      C   THE LOOP OVER ROWS IN THE DETERMINANT STARTS HERE
001768      C   N IS THE INDEX OF THE ROWS, RUNNING FROM 1 TO 2PMAX +1
001769      C   M IS THE COLUMN INDEX, WITH VALUES N-1,N,N+1
001770          DO 9 N=1,NP
001771          P=P+1
001772      C   THIS SKIPS THE 1,0 ELEMENT WHICH IS NOT NEEDED
001773          IF(N .EQ. 1) GO TO 40
001774      C   THIS IS THE CALCULATION OF THE M=N-1 OR LEFT ELEMENTS
001775          A(N,M) = 0.
001776          DO 3 I = 1, NSPEC
001777          ARH(N,M,I) = EPSF(I)*(APP(I) + BPP(I))
001778         3 A(N,M) = A(N,M) + ARH(N,M,I)
001779          A(N,M) = A(N,M)*DEPOLE
001780          M=M+1
001781      C   THIS IS THE CALCULATION OF THE DIAGONAL ELEMENTS
001782         40 CONTINUE
001783          CALL ABCALC
001784          A(N,M) = KYSQ + P*P*KOSQ
001785          DO 8 I = 1, NSPEC
001786          ARH(N,M,I) = OMPSQ*AP(I)*CHARG(I)**2*DENSE(I)/AMASS(I)
001787         8 A(N,M) = A(N,M) + ARH(N,M,I)
001788          A(N,M) = A(N,M)*DEPOLE
001789          M=M+1
001790      C   THIS IS THE CALCULATION OF THE M=N+1 OR RIGHT ELEMENTS
001791          IF(N .EQ. NP) GO TO 60
001792          CALL ABCALC
```

```
001793          A(N,M) = 0.
001794          DO 5 I = 1, NSPEC
001795          ARH(N,M,I) = EPSF(I)*(AP(I) - BP(I))
001796        5 A(N,M) = A(N,M) + ARH(N,M,I)
001797          A(N,M) = A(N,M)*DEPOLE
001798          M=M-1
001799       60 CONTINUE
001800        9 CONTINUE
001801    C     WE MAKE USE OF THE FACT THAT A(P,P') = A(-P',-P)
001802          N1 = NP
001803          NN1 = N1
001804          NP = 2*PMAX
001805          IF(NP.LT.NN1) GO TO 104
001806          DO 20 N = NN1,NP
001807          A(N,N+1) = A(N1-1,N1)
001808          A(N+1,N) = A(N1,N1-1)
001809          A(N+1,N+1) = A(N1-1,N1-1)
001810          DO 21 I = 1, NSPEC
001811          ARH(N,N+1,I) = ARH(N1-1,N1,I)
001812          ARH(N+1,N,I) = ARH(N1,N1-1,I)
001813       21 ARH(N+1,N+1,I) = ARH(N1-1,N1-1,I)
001814       20 N1 = N1-1
001815    C     THE DETERMINANT OF THE COEFFICIENTS, DETC, IS DONE HERE
001816      104 IF(.NOT.DTCALC) RETURN
001817          DETC=A(1,1)
001818          DN = 1.
001819          IF(PMAX .EQ. 0) GO TO 70
001820          DO 7 N=2,NPP
001821          DO=DN
001822          DN=DETC
001823        7 DETC=A(N,N)*DN-A(N,N-1)*A(N-1,N)*DO
001824       70 DISP=DETC/OMEGA
001825          RETURN
001826    C
001827    C     CALCULATE DISP USING THE LOCAL APPROXIMATION IF PMAX.EQ.0
001828       10 KPSQ = KYSQ + KXSQ
001829          DO 15 I = 1, NSPEC
001830          OMEGD = OMEGA + (0.,1.)*KYSQ*DFU(I)
001831          IF(DENSE(I).EQ.0.) GO TO 110
001832          EPS = EP(I)
001833          IF(RLARM(I).GT.0.) GO TO 16
001834    C     SPECIES I IS COLD
001835          TERM(I) = -KPSQ - (1.+.5*BETA)*EPS*KYAI*CHARG(I)/(AMASS(I)*OMEGD)
001836          GO TO 13
001837       16 IF(JAY(I).GE.0) GO TO 17
001838          TERM(I) = RING2(OMEGA,KYAI,EPS,RLARM(I),AMASS(I),CHARG(I),NMAX(I)
001839         .,QMAX(I),OMEGD)
001840          GO TO 18
001841       17 EPSI = 2.*EPSF(I)*AMASS(I)/(OMPSQ*CHARG(I)**2*DENSE(I)*DREL(I))
001842    C     EPSI IS THE EPSI OR EPSE (DEPENDING ON WHETHER SPECIES I IS IONS OR
001843    C     ELECTRONS) USED IN *ROOTS*, DIVIDED BY DREL (=N(X)/NO).  IT IS NEEDED
001844    C     IN *MAXW2* WHEN WE ARE USING THE HIGH GROWTH RATE SINUSOIDAL DISP. REL.
001845          TERM(I) = MAXW2( OMEGA,KYAI,EPS,RLARM(I),AMASS(I),CHARG(I),JAY(I),
001846         .NMAX(I),QMAX(I),OMEGD,KOAI,EPSI,KPSQ,DREL(I))
001847       18 TERM(I) = TERM(I)/RLARM(I)**2
001848       13 TERM(I) = TERM(I)*AMASS(I)*DENSE(I)*DREL(I)
001849          GO TO 15
001850      110 TERM(I) = 0.
001851       15 CONTINUE
001852          DISP = KPSQ/OMPSQ + .5*BETA
001853          DO 19 I = 1, NSPEC
001854       19 DISP = DISP - TERM(I)
001855          DISP = DISP*DEPOLE
001856          RETURN
001857          END
```

```
001858
001859          COMPLEX RING2
001860          FUNCTION RING2(W,KY,E,RLARM,AMASS,CHARG,NMAX,QMAX,WD)
001861          COMMON/BSTORE/ BJ(2000)
001862          COMPLEX OMEGA,W,SUM1,SUM2,WD,OMEGD
001863          INTEGER QMAX
001864          REAL KYAI,KY,KO,KOAI,KOSQ
001865          OMEGA = W*AMASS/CHARG
001866          OMEGD = WD*AMASS/CHARG
001867          KYAI = KY*RLARM
001868          EPS = E*RLARM
001869          CALL BESSJ(KYAI,NMAX,BJ)
001870          SUM1 = 0.
001871          IF(EPS.EQ.0.) GO TO 3
001872          SUM1 = BJ(1)*BJ(1)/OMEGD
001873        3 SUM2 = 0.
001874          DO 1 I = 1, QMAX
001875          AI = I
001876          SUM1 = SUM1 + BJ(I+1)*BJ(I+1)*(1./(OMEGD-AI) + 1./(OMEGD+AI))
001877        1 SUM2 = SUM2 + BJ(I+1)*(BJ(I) - BJ(I+2))*AI*(1./(OMEGD-AI) - 1./
001878         .(OMEGD+AI))
001879          SUM2 = SUM2*(1. + OMEGA*EPS/KYAI)
001880          SUM1 =-SUM1*EPS*KYAI
001881          RING2 = SUM1 + SUM2
001882          RETURN
001883          END
```

```
001884
001885          COMPLEX MAXW2
001886          FUNCTION MAXW2(W,KY,E,RLARM,AMASS,CHARG,JAY,NMAX,QMAX,WD,
001887         .KO,EPSI,KP,DREL)
001888          COMPLEX OMEGA,W,SUM1,SUM2,TP,TN,AI,WD,OMEGD
001889          COMPLEX IXKO,VP,VPPVD,VPMVD,ZEE
001890          COMPLEX CMIX,CPIX,ZVP,ZVPP,ZVPM
001891          INTEGER QMAX
001892          REAL KYAI,KY,KYSQ,KO,KOAI,KOSQ,KP,KPSQ,KPAI
001893          LOGICAL UNMAG,SNGLX
001894          COMMON/CXK/XKO
001895          COMMON/BSTORE/ BI(2000)
001896          COMMON/CMAG/UNMAG
001897          COMMON/CSNGLX/ SNGLX
001898          OMEGA = W*AMASS/CHARG
001899          OMEGD = WD*AMASS/CHARG
001900          KYAI = KY*RLARM
001901          KYSQ = KYAI*KYAI
001902          EPS = E*RLARM
001903
001904          IF(UNMAG) GO TO 4
001905          CALL BESSI(KYSQ,NMAX,BI)
001906          SUM1 = 0.
001907          IF(EPS.EQ.0.) GO TO 3
001908          SUM1 = BI(1)/OMEGD
001909        3 SUM2 = 0.
001910          DO 2 I = 1, QMAX
001911          AI = CMPLX(FLOAT(I),0.)
001912          TP = BI(I+1)/(OMEGD - AI)
001913          TN = BI(I+1)/(OMEGD + AI)
001914          SUM2 = SUM2 + AI*(TP - TN)
001915        2 SUM1 = SUM1 + TP + TN
001916          SUM2 = SUM2*(1. + OMEGA*EPS/KYAI)
001917          SUM1 =-SUM1*EPS*KYAI
001918          MAXW2 = SUM1 + SUM2
001919          RETURN
001920        4 KOAI = KO*RLARM
001921          KOSQ = KOAI*KOAI
001922          KPSQ = KP*RLARM*RLARM
001923          KPAI = SQRT(KPSQ)
001924          VP =  SQRT(.5)*OMEGD/KPAI
001925          IF(.NOT.SNGLX) GO TO 5
001926          MAXW2 = -1. - OMEGA*EPS/KYAI - SQRT(.5)*(-EPS+ OMEGD*(1.+OMEGA*EPS
001927         ./KYAI)/KYAI)*ZEE(VP)
001928          RETURN
001929        5 IXKO = CMPLX(0.,XKO)
001930          VDDOTK = KOAI*KYAI/KPAI
001931          VPPVD = SQRT(.5)*(OMEGD/KPAI + CMPLX(0.,VDDOTK))
001932          VPMVD = SQRT(.5)*(OMEGD/KPAI - CMPLX(0.,VDDOTK))
001933          CPIX= CEXP(IXKO)
001934          CMIX = CEXP(-IXKO)
001935          ZVP = ZEE(VP)
001936          ZVPP = ZEE(VPPVD)
001937          ZVPM = ZEE(VPMVD)
001938          EXPL = EXP(-.5*KOSQ)
001939          MAXW2 = EXPL*(    .5*CMIX*VPPVD*ZVPP + .5*CPIX*VPMVD*ZVPM )
001940          MAXW2 = -1. - MAXW2*EPSI - VP*ZVP/DREL
001941          RETURN
001942          END
```

```
001943
001944          SUBROUTINE QFKCAL(QF,KMAX,KMIN,RMX,RMIN)
001945          LOGICAL UNMAG
001946          DIMENSION KMAX(4), KMIN(4)
001947          COMMON/CSPEC/ RLARM(4),AMASS(4),CHARG(4),JAY(4),DENSE(4),NSPEC
001948          COMMON/CMAG/UNMAG
001949          Q = 1.
001950          DO 11 I = 1, NSPEC
001951          IF(UNMAG) GO TO 87
001952          IF(DENSE(I).EQ.0.) GO TO 87
001953          IF(I.EQ.1) GO TO 91
001954          DO 85 J = 2, I
001955       85 IF(AMASS(I).EQ.AMASS(J - 1).AND.DENSE(J-1).NE.0.) GO TO 87
001956       91 KMAX(I) = RMX*AMASS(I)
001957          KMIN(I) = (RMIN - 2.)*AMASS(I)
001958          IF(KMIN(I).LT.1) KMIN(I) = 1
001959          GO TO 88
001960       87 KMAX(I) = 0
001961       88 IF(KMAX(I).EQ.0) GO TO 11
001962          KMX = (KMAX(I) - KMIN(I))/2 + 1
001963          DO 86 K = 1, KMX
001964       86 Q = FLOAT(K)*Q/AMASS(I)
001965       11 CONTINUE
001966          QF = 1./(Q*Q)
001967          RETURN
001968          END
```

```
001969
001970            SUBROUTINE QCALC(CHI,QMAX,NMAX,N)
001971            INTEGER QMAX
001972            COMMON/CQN/MINQ
001973            IF(N.GT.0) GO TO 1
001974      C   THIS IS USED FOR A MAXWELLIAN DISTRIBUTION OF IONS.
001975      C   QMAX IS MADE LARGE ENOUGH SO THAT MODIFIED BESSEL FUNCTIONS OF INDEX
001976      C   GREATER THAN QMAX ARE LESS THAN 1.E-03
001977      C   THE PARAMETER NMAX USED IN THE SUBROUTINE BESSI IS CHOSEN SO THAT
001978            IF(CHI.LE.64.) AMAX = 28.
001979            IF(CHI.GT.64.) AMAX = 3.5*SQRT(CHI)
001980            NMAX = AMAX
001981      C   QMAX AND NMAX MUST BE SIGNIFICANTLY MORE THAN THE HIGHEST CYCLOTRON
001982      C   HARMONIC SOUGHT.
001983            IF(NMAX.LT.MINQ) NMAX = MINQ
001984      C   IF NMAX IS TOO GREAT, BESSI WILL OVERFLOW.
001985            IF(NMAX.GT.1988) NMAX = 1988
001986            QMAX = NMAX
001987            RETURN
001988      C   THIS IS USED FOR RING DISTRIBUTIONS OF IONS.
001989      C   QMAX IS MADE LARGE ENOUGH SO THAT BESSEL FUNCTIONS OF INDEX
001990      C   GREATER THAN QMAX ARE LESS THAN 1.E-05.
001991          1 QMAX = 10
001992            IF(CHI.GT.4.) QMAX = SQRT(CHI) + 6.5*CHI**.166667
001993      C   THE PARAMETER NMAX USED IN THE SUBROUTINE BESSJ IS CHOSEN SO THAT
001994      C   THE BESSEL FUNCTIONS ARE GOOD TO 10 DECIMAL PLACES.
001995            NMAX = 15
001996            IF(CHI.GT.9.) NMAX = SQRT(CHI) + 11.*CHI**.166667
001997            IF(NMAX.GT. 398) NMAX =  398
001998            IF(QMAX.GT.NMAX - 2) QMAX = NMAX - 2
001999            RETURN
002000            END
```

```
002001
002002            SUBROUTINE BESSI(X,N,BI)
002003            DIMENSION BI(2000)
002004            N1 = N+1
002005            IF (X.LT.1.E-14) GO TO 4
002006            BI(N+2) = 0.
002007            BI(N+1) = 1.
002008            DO 1 I = 1, N
002009            M = N - I + 1
002010            AM1 = M
002011          1 BI(M) = (2.*AM1/X)*BI(M+1) + BI(M+2)
002012            SUM = BI(1)/2.
002013            DO 2 M = 2, N1
002014          2 SUM = SUM + BI(M)
002015            SUM = SUM + SUM
002016            DO 3 I = 1, N1
002017          3 BI(I) = BI(I)/SUM
002018            GO TO 6
002019          4 BI(1) = 1.
002020            DO 5 I = 2, N1
002021          5 BI(I) = 0.
002022          6 RETURN
002023            END
```

```
002024
002025            SUBROUTINE BESSJ (X,N,BJ)
002026            DIMENSION  BJ(400)
002027            N1 = N+1
002028            IF (X.LT.1.E-14) GO TO 4
002029            BJ(N+2) = 0.
002030            BJ(N+1) = 1.
002031            DO 1 I = 1, N
002032            M = N - I + 1
002033            AM1 = M
002034          1 BJ(M) = (2.*AM1/X)*BJ(M+1) - BJ(M+2)
002035            SUM = BJ(1)/2.
002036            N2 = N/2
002037            DO 2 M = 1, N2
002038          2 SUM = SUM + BJ(M+M+1)
002039            SUM = SUM + SUM
002040            DO 3 I = 1, N1
002041          3 BJ(I) = BJ(I)/SUM
002042            GO TO 6
002043          4 BJ(1) = 1.
002044            DO 5 I = 2, N1
002045          5 BJ(I) = 0.
002046          6 RETURN
002047            END
```

```
002048
002049          FUNCTION CABS2(Z)
002050          COMPLEX Z
002051          CABS2 = Z*CONJG(Z)
002052          RETURN
002053          END
```

```
002054
002055          COMPLEX ZEE
002056          FUNCTION ZEE(X)
002057          COMPLEX X,TERM
002058          DOUBLE PRECISION RZED,IZED,RTERMD,ITERMD,RTNEW,RX2,IX2
002059          DATA PI/3.14159265359/
002060          ABX2 = CABS2(X)
002061    C     IF ABS(X) IS GREATER THAN 5.7, USE THE ASYMPTOTIC FORM.
002062          IF(ABX2.GT.33.) GO TO 4
002063          ZEE = (0.,1.)*SQRT(PI)*CEXP(-X*X)
002064          IF(ABX2.GT.1.) NMAX = 2.8*ABX2 + 19.
002065          IF(ABX2.LE.1.) NMAX = 11.*ABX2 + 10.
002066          TERM = 2.*X
002067    C     IF ABS(X) IS GREATER THAN 2.6, WE MUST USE DOUBLE PRECISION IF ZEE IS
002068    C     TO BE ACCURATE TO 1.E-12.
002069          IF(ABX2.GT.7.) GO TO 2
002070          DO 1 N = 1,NMAX
002071          ZEE = ZEE - TERM
002072        1 TERM = -TERM*2.*X*X/(2*N + 1)
002073          RETURN
002074        2 RX2 = REAL(X*X)
002075          IX2 = AIMAG(X*X)
002076          RZED = REAL(ZEE)
002077          IZED = AIMAG(ZEE)
002078          RTERMD = REAL(TERM)
002079          ITERMD = AIMAG(TERM)
002080          DO 3 N = 1, NMAX
002081          RZED = RZED - RTERMD
002082          IZED = IZED - ITERMD
002083          RTNEW = -2.*(RX2*RTERMD - IX2*ITERMD)/(2*N +1)
002084          ITERMD = -2.*(RX2*ITERMD + IX2*RTERMD)/(2*N + 1)
002085        3 RTERMD = RTNEW
002086          RZEE = RZED
002087          AZEE = IZED
002088          ZEE = CMPLX(RZEE,AZEE)
002089          RETURN
002090        4 NMAX = 20
002091          TERM = 1./X
002092          ZEE = 0.
002093          DO 5 N = 1, NMAX
002094          ZEE = ZEE - TERM
002095        5 TERM = FLOAT(2*N-1)*TERM/(2.*X*X)
002096          RETURN
002097          END
```

```
002098
002099          SUBROUTINE ABCALC
002100    C     THIS IS AN INTERFACE BETWEEN DISP AND THE ROUTINES MAXW, RING, AND
002101    C     COLD WHICH CALCULATE THE ALPHAS AND BETAS.
002102          DIMENSION CHIP1(4)
002103          COMPLEX AP,BP,OMEGA,W,APP,BPP
002104          REAL J1(400,4), J2(400,4)
002105          COMMON/CCHIP/CHIP(4),AP(4),BP(4),     OMEGA,APP(4),BPP(4)
002106          COMMON/CSPEC/ RLARM(4),AMASS(4),CHARG(4),JAY(4),DENSE(4),NSPEC
002107          DO 5 I = 1, NSPEC
002108          IF(DENSE(I).EQ.0.) GO TO 6
002109          W = OMEGA*AMASS(I)/CHARG(I)
002110          IF(RLARM(I).EQ.0.) GO TO 7
002111          IF(JAY(I).GE.0) CALL MAXW(W,RLARM(I),AP(I),BP(I),APP(I),BPP(I),
002112         .CHIP(I),CHIP1(I),I)
002113          IF(JAY(I).LT.0) CALL RING(W,RLARM(I),AP(I),BP(I),APP(I),BPP(I),
002114         .CHIP(I),CHIP1(I),J1(1,I),J2(1,I),I)
002115          FACTOR =                (AMASS(I)/(CHARG(I)*RLARM(I)))**2
002116          AP(I) = AP(I)*FACTOR
002117          BP(I) = BP(I)*FACTOR
002118          APP(I) = APP(I)*FACTOR
002119          BPP(I) = BPP(I)*FACTOR
002120          GO TO 5
002121        6 AP(I) = 0.
002122          BP(I) = 0.
002123          APP(I) = 0.
002124          BPP(I) = 0.
002125          GO TO 5
002126        7 CALL COLD(W,AP(I),BP(I),APP(I),BPP(I),CHIP(I),CHIP1(I))
002127          FACTOR =              (AMASS(I)/CHARG(I))**2
002128          AP(I) = AP(I)*FACTOR
002129          BP(I) = BP(I)*FACTOR
002130          APP(I) = APP(I)*FACTOR
002131          BPP(I) = BPP(I)*FACTOR
002132        5 CONTINUE
002133          RETURN
002134          END
```

```
002135
002136          SUBROUTINE MAXW(OMEGA,RLARM,AP,BP,APP,BPP,LAMDP,LAMDP1,K)
002137      C   THIS ROUTINE CALCULATES ALPHAS AND BETAS FOR A MAXWELLIAN DISTRIBUTI
002138          DIMENSION IPP(2000)
002139          COMPLEX OMEGA,UMEGA,OMSQ,AP,BP,DIV,DENOM,SUM1,SUM2,VP,ZEE
002140          COMPLEX APP,BPP,EIAP,EIQAP,SUM3,SUM4,VPM,VPP,ZVPP,ZVPM
002141          REAL MASS,KOAI,KOSQ,KYAI,KYSQ,MASQ,LAMDP,LAMDP1,LAM,IPP
002142          INTEGER PMAX, P,QMAX
002143          LOGICAL UNMAG
002144          COMMON/DIS/QF,CYSQ,COSQ,EPSF(4),    MASS,CYAI,COAI,MASQ,PMAX,OMPSQ
002145         .,AEAI,KMAX(4),AMASQ(4),EPS(4),KMIN(4)
002146          COMMON/APBP/N,M,P
002147          COMMON/CRBESJ/QMAX(4), NMAX1(4)
002148          COMMON/CMAG/ UNMAG
002149          NQ = QMAX(K) + 1
002150          NMAX = NMAX1(K)
002151          KYAI = CYAI*RLARM
002152          KYSQ = KYAI*KYAI
002153          KOAI = COAI*RLARM
002154          KOSQ = KOAI*KOAI
002155          OMSQ = OMEGA*OMEGA
002156          IF(M.NE.N) GO TO 2
002157          IF(N.EQ.1) LAMDP1= SQRT(KYSQ + PMAX*PMAX*KOSQ)
002158          LAMDP = LAMDP1
002159          LAMDP1 = SQRT(KYSQ + (P+1)*(P+1)*KOSQ)
002160          LAM = LAMDP*LAMDP
002161          IF(UNMAG) GO TO 7
002162          CALL BESSI(LAM,NMAX,IPP)
002163          SUM1 = 0.
002164          DO 5 K1=2,NQ
002165          Q=K1-1
002166          DENOM = (1.,0.)/(OMSQ - Q*Q)
002167        5 SUM1= SUM1+ IPP(K1)*DENOM*Q*Q
002168          AP  = -2.0*SUM1
002169          BP= 0.
002170          APP = 0.
002171          BPP = 0.
002172          RETURN
002173        7 VP = OMEGA/SQRT(2.*LAM)
002174          AP = 1. + VP*ZEE(VP)
002175          BP = 0.
002176          APP = 0.
002177          BPP = 0.
002178          RETURN
002179        2 IF(M.NE.N+1) GO TO 3
002180          LAM = LAMDP1 * LAMDP
002181          EXPL = EXP(-.5*(LAMDP1 - LAMDP)**2)
002182          IF(UNMAG) GO TO 8
002183          EIAP = (KYSQ + P*(P+1)*KOSQ -(0.,1.)*KOAI*KYAI)/LAM
002184          CALL BESSI(LAM,NMAX,IPP)
002185          SUM1 = 0.
002186          SUM2 = IPP(1)/OMEGA
002187          SUM3 = 0.
002188          SUM4 = SUM2
002189          EIQAP = 1.
002190          DO 6 K1=2,NQ
002191          Q=K1-1
002192          EIQAP = EIQAP*EIAP
002193          SUM1 = SUM1 + Q*IPP(K1)*(EIQAP/(OMEGA-Q) - 1./(EIQAP*(OMEGA+Q)))
002194          SUM2 = SUM2 +   IPP(K1)*(EIQAP/(OMEGA-Q) + 1./(EIQAP*(OMEGA+Q)))
002195          SUM3 = SUM3 + Q*IPP(K1)*(1./(EIQAP*(OMEGA-Q)) - EIQAP/(OMEGA+Q))
002196        6 SUM4 = SUM4 +   IPP(K1)*(1./(EIQAP*(OMEGA-Q)) + EIQAP/(OMEGA+Q))
002197          AP = -EXPL*SUM1
002198          BP = (0.,1.)*KOAI*KYAI*EXPL*SUM2
002199          APP = -EXPL*SUM3
002200          BPP = (0.,1.)*KOAI*KYAI*EXPL*SUM4
002201          RETURN
002202        8 SQ2LM = SQRT(2.*LAM)
002203          VP = OMEGA/SQ2LM
002204          VPM = (OMEGA - (0.,1.)*KYAI*KOAI)/SQ2LM
002205          VPP = (OMEGA + (0.,1.)*KYAI*KOAI)/SQ2LM
002206          ZVPP = ZEE(VPP)
002207          ZVPM = ZEE(VPM)
002208          EXPL = EXP(-.5*KOSQ)
002209          AP = EXPL*(1. + VP*ZVPP)
002210          BP = -(0.,1.)*KOAI*KYAI*EXPL*ZVPP /SQ2LM
002211          APP= EXPL*(1. + VP*ZVPM)
002212          BPP= -(0.,1.)*KOAI*KYAI*EXPL*ZVPM /SQ2LM
002213          RETURN
002214        3 AP = 0.
002215          BP = 0.
002216          APP = 0.
002217          BPP = 0.
002218          RETURN
002219          END
```

```
002220
002221          SUBROUTINE RING (OMEGA,RLARM,AP,BP,APP,BPP,CHIP,CHIP1,J1,J2,K)
002222      C   CALCULATES THE ALPHAS AND BETAS FOR A RING DISTRIBUTION.
002223          DIMENSION J1(1), J2(1)
002224          COMPLEX OMEGA,UMEGA,OMSQ,AP,BP,DIV,DENOM,SUM1,SUM2
002225          COMPLEX SUM3,SUM4,EIAP,EIQAP,APP,BPP
002226          REAL MASS,KOAI,KOSQ,KYAI,KYSQ,MASQ,J1,J2,JP1,JP2
002227          INTEGER PMAX, P,QMAX
002228          COMMON/DIS/QF,CYSQ,COSQ,EPSF(4),     MASS,CYAI,COAI,MASQ,PMAX,OMPSQ
002229         .,AEAI,KMAX(4),AMASQ(4),EPS(4),KMIN(4)
002230          COMMON/APBP/N,M,P
002231          COMMON/CRBESJ/QMAX(4),NMAX1(4)
002232          NMAX = NMAX1(K)
002233          NQ = QMAX(K) + 1
002234          NQ1 = NQ + 1
002235          KYAI = CYAI*RLARM
002236          KYSQ = KYAI*KYAI
002237          KOAI = COAI*RLARM
002238          KOSQ = KOAI*KOAI
002239          OMSQ = OMEGA*OMEGA
002240          IF(M.NE.N) GO TO 2
002241          IF(N.EQ.1) CHIP1 = SQRT(KYSQ + PMAX*PMAX*KOSQ)
002242          IF(N.EQ.1) CALL BESSJ(CHIP1,NMAX,J2)
002243          CHIP = CHIP1
002244          DO 7 K1 = 1, NQ1
002245        7 J1(K1) = J2(K1)
002246          CHIP1 = SQRT(KYSQ + (P+1)*(P+1)*KOSQ)
002247          CALL BESSJ(CHIP1,NMAX,J2)
002248          SUM1 = 0.
002249          DO 5 K1=2,NQ
002250          Q=K1-1
002251          DENOM = (1.,0.)/(OMSQ-Q*Q)
002252          JP1 = .5*(J1(K1-1) - J1(K1+1))
002253        5 SUM1 = SUM1 + Q*Q*JP1*J1(K1)*DENOM
002254          AP = -4.0*SUM1*CHIP
002255          BP = 0.
002256          APP = 0.
002257          BPP = 0.
002258          RETURN
002259        2 IF(M.NE.N+1) GO TO 3
002260          EIAP = (KYSQ + P*(P+1)*KOSQ - (0.,1.)*KOAI*KYAI)/(CHIP*CHIP1)
002261          SUM1 = 0.
002262          SUM2 = J1(1)*J2(1)/OMEGA
002263          SUM3 = 0.
002264          SUM4 = SUM2
002265          EIQAP = 1.
002266          DO 6 K1=2,NQ
002267          Q=K1-1
002268          EIQAP = EIQAP*EIAP
002269          JP1 = .5*(J1(K1-1) - J1(K1+1))
002270          JP2 = .5*(J2(K1-1) - J2(K1+1))
002271          SUM1 = SUM1 +   Q*(CHIP*JP1*J2(K1) + CHIP1*JP2*J1(K1))*(EIQAP/
002272         .(OMEGA-Q) - 1./(EIQAP*(OMEGA+Q)))
002273          SUM2 = SUM2 + J1(K1)*J2(K1)*(EIQAP/(OMEGA-Q) + 1./(EIQAP*(OMEGA+Q)
002274         .))
002275          SUM3 = SUM3 +   Q*(CHIP*JP1*J2(K1) + CHIP1*JP2*J1(K1))*(1./(EIQAP*
002276         .(OMEGA-Q)) - EIQAP/(OMEGA+Q))
002277        6 SUM4 = SUM4 + J1(K1)*J2(K1)*(1./(EIQAP*(OMEGA-Q))+ EIQAP/(OMEGA+Q)
002278         .)
002279          AP = -SUM1
002280          BP = (0.,1.)*KOAI*KYAI*SUM2
002281          APP = -SUM3
002282          BPP= (0.,1.)*KOAI*KYAI*SUM4
002283          RETURN
002284        3 AP = 0.
002285          BP = 0.
002286          APP = 0.
002287          BPP = 0.
002288          RETURN
002289          END
```

```
002290
002291            SUBROUTINE COLD(OMEGA, AP,BP,APP,BPP,CHIP,CHIP1)
002292      C   CALCULATES ALPHAS AND BETAS FOR COLD ELECTRONS
002293            COMPLEX OMEGA,OMSQ,DIV,AP,BP,APP,BPP
002294            REAL MASS,KOAI,KOSQ,KYAI,KYSQ,MASQ
002295            INTEGER PMAX, P
002296            COMMON/DIS/QF,KYSQ,KOSQ,EPSF(4),     MASS,KYAI,KOAI,MASQ,PMAX,OMPSQ
002297           .,AEAI,KMAX(4),AMASQ(4),EPS(4),KMIN(4)
002298            COMMON/APBP/N,M,P
002299            OMSQ = OMEGA*OMEGA
002300            IF(M.NE.N) GO TO 2
002301            IF(N.EQ.1) CHIP1 = SQRT(KYSQ + PMAX*PMAX*KOSQ)
002302            CHIP = CHIP1
002303            CHIP1 = SQRT(KYSQ + (P+1)*(P+1)*KOSQ)
002304            AP = -CHIP*CHIP/(OMSQ - 1.)
002305            BP = 0.
002306            APP = 0.
002307            BPP = 0.
002308            RETURN
002309          2 COSAP = (KYSQ + P*(P+1)*KOSQ)/(CHIP*CHIP1)
002310            SINAP = -KOAI*KYAI/(CHIP*CHIP1)
002311            AP = -CHIP*CHIP1*(COSAP + (0.,1.)*OMEGA*SINAP)/(OMSQ - 1.)
002312            APP= -CHIP*CHIP1*(COSAP - (0.,1.)*OMEGA*SINAP)/(OMSQ - 1.)
002313            IF(KOAI.NE.0.) GO TO 1
002314            BP = 0.
002315            BPP = 0.
002316            RETURN
002317          1 BP = KOAI*KYAI/OMEGA
002318            BP =CMPLX(-AIMAG(BP ),REAL(BP ))
002319            BPP = BP
002320            RETURN
002321            END
```

```
002322
002323            SUBROUTINE SAVPHI
002324      C   FINDS PHI(P)
002325            COMPLEX PHI(18,22), A(20,20), PHI1(19), PHI2(19)
002326            COMMON /PHIC/ A,     NRTSV, PHI
002327            COMMON/PMX/NP
002328            DATA NRTSV/1/
002329            IF(NRTSV.LT.19) GO TO 1
002330            CALL EIGPLT
002331            NRTSV = 1
002332          1 PHI1(1) = (1.0,0.0)
002333            IF((NP-1)/2 .EQ. 0) GO TO 5300
002334            PHI1(2) = -A(1,1)/A(1,2)
002335            DO 5301 K2 = 3,NP
002336            K1 = K2-1
002337            PHI1(K2) = -(A(K1,K1-1)*PHI1(K1-1)+A(K1,K1)*PHI1(K1))/A(K1,K2)
002338       5301 CONTINUE
002339            PHI2(NP) = PHI1(NP)
002340            PHI2(NP-1) = -A(NP,NP)*PHI2(NP)/A(NP,NP-1)
002341            DO 5302 K2 = 3,NP
002342            K1 = NP-K2+1
002343            PHI2(K1) = -(A(K1+1,K1+1)*PHI2(K1+1)+A(K1+1,K1+2)*PHI2(K1+2))
002344           .     /A(K1+1,K1)
002345       5302 CONTINUE
002346            DO 5303 K1 = 1,NP
002347            PHI(NRTSV,K1) = 0.5*(PHI1(K1) + PHI2(K1))
002348       5303 CONTINUE
002349       5300 CONTINUE
002350            RETURN
002351            END
```

```
002352            SUBROUTINE PAGE
002353            COMMON/CIO/ICR,IHSP,IGRAPH
002354            WRITE(IHSP,1)
002355          1 FORMAT(1H1)
002356            RETURN
002357            END
```

```
002358
002359            SUBROUTINE MRAF(F,RT,CONV,NRTS,PP1,PP2,PP3,EP1,EP2,MXM)
002360            DIMENSION ARPHI (80), ABPHI(80)
002361            LOGICAL PHIZ(200), WARN
002362            LOGICAL CONV(200)
002363            LOGICAL NOCONJ, NONEG, NOPHNU
002364            LOGICAL SW1,SW2,SW3,SW4,SW5,SWR,SW7,SWR1,NREDH,SW31,SW71
002365            COMPLEX PHI(18,22), A(20,20), PHJ
002366            COMPLEX F,RT(200),PP1,PP2,PP3,P1,P2,P3,CHANGE
002367            COMPLEX H,ROOT,FUNC,FROOT
002368            COMPLEX DNR,FX1,FX2,FX3,FX21,FX32,FX321,RHO,T,X1,X2,X3,W
002369            COMPLEX PHINEW(20)
002370            REAL EP3,FRMAX,XRMAX
002371            REAL TEST,XR
002372            REAL KYAI
002373            REAL EP1,EP2
002374            INTEGER NRTS,MXM
002375            INTEGER RTC,ITC
002376            INTEGER PMAX, EIGV, FAILC
002377            INTEGER I,NCR,LNCR,NRET
002378            COMMON/CIO/ ICR,IHSP,IGRAPH
002379            COMMON/CMRAF2/ ERRMAX,NRTSU,SW31,SW71
002380            COMMON /PHIC/ A,      NRTSV, PHI
002381            COMMON/PMX/NPP
002382            COMMON/CPHZ/ PHIZ
002383            COMMON/CMRAF/SW1,SW2,SW3,SW4,SW5,SWR,RTC,ITC,EP3,FRMAX,XRMAX
002384           .   ,H,ROOT,FUNC,FROOT,EIGV, SW7,START,RMX,NFL,KYAI
002385            COMMON/CPHCK/ PHINEW,PHZP
002386            DATA SW1,SW2,SW3,SW4,SW5,SWR/6*(.FALSE.)/
002387            DATA SW7/.FALSE./
002388            DATA RTC,EP3,FRMAX,XRMAX/0,0.,10.,10./
002389            DATA ERRMAX/ 1.E+20/
002390      C  'MULLER'S (METHOD FOR) ROOTS (OF A LOCALLY) ANALYTIC FUNCTION'.
002391      C
002392      C  NRTS ROOTS OF F(Z) WILL BE FOUND (HOPEFULLY) AND PLACED IN ARRAY
002393      C   RT(NRTS).
002394      C  P1, P2, P3 ARE STARTING VALUES FOR THE SEARCH (EXCEPT SEE SW5 BELOW).
002395      C   ROOTS NEAREST THE STARTING VALUES WILL BE FOUND FIRST, ROUGHLY.
002396      C  EP1, EP2 ARE CONVERGENCE TOLERANCE PARAMETERS.
002397      C   A ROOT IS CONSIDERED TO HAVE BEEN FOUND IF..
002398      C     (1) THE MODULUS OF THE RELATIVE DIFFERENCE OF TWO ITERANTS .LT. EP1
002399      C     (2) MODULII OF THE FUNCTION AND MODIFIED FUNCTION VALUES BOTH .LT.
002400      C       EP2.
002401      C    'MODIFIED FUNCTION' MEANS F(Z)/DNR, WHERE DNR=PRODUCT (Z-RT(I)),
002402      C    THE PRODUCT BEING OVER ALL PREVIOUSLY FOUND ROOTS.
002403      C  IF MXM ITERATIONS ON ROOT I DOES NOT PRODUCE CONVERGENCE, ITERATION
002404      C   STOPS. WHAT HAPPENS NEXT DEPENDS ON SW4 - DEFAULT PROCEDURE IS TO
002405      C   SET THE REST OF THE ARRAY CONV=.FALSE. AND RETURN.
002406      C
002407      C  IF SW1 INFORMATION ABOUT EACH ROOT FOUND IS PRINTED.
002408      C  IF SW2, THEN INFORMATION ABOUT EACH ITERANT OF EACH ROOT IS PRINTED.
002409      C  IF SW3 , THEN THE CONJUGATE OF EACH NON-REAL ROOT FOUND IS CONSIDERED
002410      C   TO BE A ROOT ALSO.
002411      C  IF SW4, SOME ATTEMPT IS MADE TO CONTINUE AFTER FAILURE TO CONVERGE
002412      C   ON A ROOT. THE LATEST ITERANT IS ASSIGNED TO RT(I) AND CONV(I)=
002413      C   .FALSE. (ELSE CONV(I)=.TRUE.), AND WE CONTINUE NORMALLY. THE
002414      C   MODIFIED FUNCTION NOW HAS A POLE AT THIS FALSE ROOT. BUT
002415      C   GENERALLY WE SEEM NOW TO CONVERGE TO A ROOT, SO IT CAN BE
002416      C   WORTHWHILE CONTINUING, IF THERE ARE MORE ROOTS ASKED FOR. THIS
002417      C   IS ESPECIALLY TRUE IN CASES OF SLOW CONVERGENCE TO MULTIPLE ROOTS.
002418      C   WHEN NRTS (GOOD OR BAD) ROOTS HAVE BEEN FOUND, MRAF WILL GO BACK
002419      C   AND TRY TO RE-DO THE NONCONVERGENT ROOT ESTIMATES. ON RETURN, CONV
002420      C   WILL INDICATE WHICH ROOTS ARE GOOD.
002421      C  IF SW5, THE STARTING VALUES IN THE SEARCH FOR ROOT I WILL BE PP1 ETC.
002422      C   PLUS WHATEVER WAS IN RT(I) ON ENTRY.
002423      C  IF SW7, THE NEGATIVE CONJUGATE OF EACH ROOT FOUND IS CONSIDERED TO BE
002424      C   A ROOT.
002425      C  IF SWR, ONLY REAL ROOTS ARE FOUND, AND ONLY REAL ARGUMENTS ARE GIVEN
002426      C   TO F, ALTHOUGH F ITSELF IS STILL A COMPLEX FUNCTION (UNLESS YOUR
002427      C   COMPILER RETURNS REAL FUNCTION VALUES IN THE SAME PLACE AS THE
002428      C   REAL PART OF COMPLEX FUNCTION VALUES).
002429      C     IN GENERAL IT IS EASIER TO SOLVE FOR ALL ROOTS INSTEAD OF ONLY REAL
002430      C     ROOTS, ESPECIALLY IF THERE ARE ROOTS NEAR REAL AXIS IN REGION OF
002431      C     INTEREST.
002432      C   (IN REVISED VERSION OF PROGRAM, SWR IS TEMPORARILY MADE FALSE IF THERE
002433      C   IS  OBVIOUSLY A COMPLEX ROOT NEARBY. IF SWR IS TRUE THEN REAL ROOTS ARE
002434      C   FOUND MORE QUICKLY, SO IT IS BETTER TO SET SWR TRUE INITIALLY. - MJG
002435      C  IF RTC.LT.0 ON ENTRY, WE CONTINUE WITHOUT REINITIALIZING JUST AS
002436      C     THOUGH WE WERE ABOUT TO LOOK FOR ROOT ABS(RTC)+1.
002437      C  THE MODIFIED FUNCTION IS NOT WELL DEFINED CLOSE TO ROOTS ALREADY
002438      C   FOUND. THE RELATIVE SIZE OF THE REGION OF INDETERMINACY IS TAKEN
002439      C   TO BE EP3. A STARTING VALUE IN SUCH A REGION WILL BE MOVED OUT
002440      C   BEFORE CONTINUING. AN ITERANT THERE WILL BE ASSUMED TO BE ANOTHER
002441      C   ROOT.
002442      C  THE NORMAL ITERATION IS ALTERED IN SOME CASES WHERE THE METHOD
002443      C   OTHERWISE TENDS TO GET LOST..
002444      C     H IS THE DIFFERENCE BETWEEN 2 ITERANTS. THE RATIO OF TWO SUCCESSIVE
002445      C   H'S NEVER EXCEEDS XRMAX. IF A NEWLY FOUND H IS TOO LARGE, ITS
002446      C   MAGNITUDE IS DECREASED BUT ITS DIRECTION IS PRESERVED.
002447      C     IF THE RATIO OF THE MODIFIED FUNCTION AT TWO SUCCESSIVE ITERANTS
002448      C   EXCEEDS FRMAX, THE LATEST ITERATION IS DISCARDED AND THE H FOR
002449      C   THAT ITERATION IS NOW TAKEN TO BE THE SAME AS IN THE PRECEEDING
002450      C   ITERATION.
002451      C  IF YOU DON'T  ASSIGN ANYTHING TO A CMRAF VARIABLE YOU GET THE DEFAULT
002452      C   SET BY THE DATA STATEMENTS. A COROLLARY IS THAT YOU SHOULDN'T TRY
002453      C   SETTING THEM BY DATA STATEMENTS IN YOUR PROGRAM.
```

```
002454      C
002455      C   THE PROCEDURE MAY BE THOUGHT OF AS A GENERATION OF A SEQUENCE Z(I),
002456      C    WHOSE LIMIT IS A ROOT OF F AND WHOSE FIRST 3 ELEMENTS ARE Z(-2)=P1,
002457      C    Z(-1)=P2, Z(0)=P3. A QUADRATIC IS FITTED TO THE FUNCTION AT THREE
002458      C    ELEMENTS Z(I-2),Z(I-1),Z(I), USING NEWTON'S DIVIDED DIFFERENCES.
002459      C    Z(I+1) IS NORMALLY THAT ROOT OF THE QUADRATIC CLOSEST TO Z(I).
002460      C   SINCE THE FITTED FUNCTION, A QUADRATIC, IS ANALYTIC WE SHOULD NOT
002461      C    BE SURPRISED WHEN THE METHOD FAILS ON NON-ANALYTIC FUNCTIONS.
002462      C   THE USER CAN OFTEN HELP GREATLY BY MAKING HIS FUNCTION WELL-
002463      C    BEHAVED IN THE REGION OF INTEREST, BY MULTIPLYING HIS ORIGINAL
002464      C    FUNCTION BY ANOTHER WHICH CANCELS SINGULARITIES IN THE ORIGINAL
002465      C    WITHOUT ADDING ANYTHING MORE.
002466      C   CLOSE TO A SINGLE ROOT, CONVERGENCE IS LITTLE FASTER THAN FOR THE
002467      C    SECANT METHOD, BECAUSE THE ONLY ADDITIONAL INFORMATION USED IS FROM
002468      C    X(I-2), WHICH IS A LONG WAY OFF.
002469      C   HOWEVER CONVERGENCE TO MULTIPLE ROOTS IS IMPROVED OVER THE SECANT OR
002470      C    NEWTON-RAPHSON METHODS. IN PARTICULAR CONVERGENCE TO DOUBLE ROOTS
002471      C    IS ABOUT AS FAST AS TO SIMPLE ROOTS.
002472      C   THIS METHOD IS CAPABLE OF FINDING COMPLEX ZEROES OF A FUNCTION WHICH
002473      C    IS REAL FOR REAL ARGUMENT, USING REAL STARTING VALUES. THIS IS NOT
002474      C    TRUE OF, FOR INSTANCE, NEWTON-RAPHSON OR THE SECANT METHOD.
002475      C
002476      C   THIS PROGRAM BEGAN AS A FORTRAN TRANSLATION OF CACM ALGORITHM 196,
002477      C    INCORPORATING CHANGES SUGGESTED BY J. TRAUB AND THE CERTIFICATION
002478      C    APPEARING IN THE CACM, JAN. 1968, P12. THEN SOME REMAINING GLITCHES
002479      C    WERE REMOVED AND NEW FEATURES ADDED BY
002480      C   A.B.LANGDON, EECS, BERKELEY, FEB. 1968.
002481      C
002482      C   CABS2(Z)=CABS(Z)**2=REAL(Z)**2+AIMAG(Z)**2
002483      C     (AVOIDS TIME WASTED IN SQRT IN CABS)
002484      C
002485            ANP = NPP
002486            IZ = (ANP + 1.)/2.
002487            P1 = CMPLX(REAL(PP1),AIMAG(PP1))
002488            P2 = CMPLX(REAL(PP2),AIMAG(PP2))
002489            P3 = CMPLX(REAL(PP3),AIMAG(PP3))
002490            LNCR=0
002491            NOPHNU = .FALSE.
002492            FAILC = 0
002493            EP3 = 1.E-13
002494            SWR1 = SWR
002495            IF (SW7 .AND. SW1) PRINT 512
002496        512 FORMAT (61H NEGATIVE CONJUGATE OF A CONVERGED ROOT IS CONSIDERED A
002497           . ROOT.  )
002498            IF( RTC.LT.0 ) GO TO 100
002499            RTC=0
002500            DO 99 I=1,NRTS
002501            PHIZ(I) = .TRUE.
002502         99 CONV(I)=.TRUE.
002503        100 RTC=IABS(RTC)
002504      C
002505      C   INITIALIZE SEARCH FOR NEXT ROOT.
002506         10 ITC=0
002507      C    IF EIGV = 0, PHI(P) IS NOT CALCULATED.
002508      C    IF EIGV = 2, PHI(P) IS CALCULATED FOR ALL ROOTS
002509      C    IF EIGV = 3, PHI(P) IS CALCULATED FOR ALL ROOTS, AND PHI(X) IS
002510      C       PLOTTED FOR UNSTABLE ROOTS.
002511      C    IF EIGV = 4, PHI(X) PLOTTED FOR ALL ROOTS.
002512            NOCONJ=.FALSE.
002513            NONEG = .FALSE.
002514            SWR = SWR1
002515      C   IF SW5, START SEARCH NEAR RT(RTC+1)
002516            IF( .NOT.SW5 ) GO TO 9
002517            P1=PP1+START
002518            P2=PP2+START
002519            P3=PP3+START
002520         9 CONTINUE
002521      C   SET FX1=FROOT(P1) ETC.
002522            I=RTC+1
002523            IF(SW2) PRINT    498, I
002524       498 FORMAT(5H ROOT,I3,23H SEARCH STARTING POINTS)
002525            ROOT=P1
002526            NRET=1
002527            GO TO 200
002528          1 FX1=FROOT
002529            X1=P1
002530            IF( SW2 ) PRINT    499, NRET, ROOT, FROOT
002531       499 FORMAT(3H  Z,I1,1H=,2E22.13,19H MODIFIED FUNCTION=,2E22.13)
002532            ROOT=P2
002533            NRET=2
002534            GO TO 200
002535          2 FX2=FROOT
002536            X2=P2
002537            IF( SW2 ) PRINT    499, NRET, ROOT, FROOT
002538            ROOT=P3
002539            NRET=3
002540            GO TO 200
002541          3 FX3=FROOT
002542            X3=P3
002543      C    THE FUNCTION WILL BE CONSIDERED SMALL (FOR PURPOSES OF CONVERGENCE TEST)
002544      C    WHEN IT IS SMALL COMPARED TO FI2, EVEN IF THIS IS NOT OF ORDER UNITY
002545            FI2 = AMIN1(CABS2(FX3),CABS2(FX2))
002546            FI2 = AMIN1(FI2,CABS2(FX1))
002547            IF( SW2 ) PRINT    499, NRET, ROOT, FROOT
002548      C   PREPARE FIRST MULLER-TRAUB ITERATION.
002549            H=X3-X2
002550            FX21=(FX2-FX1)/(X2-X1)
002551      C   IF P3= A ROOT WE GET FX3/FX2=0/0 LATER, SO CHECK THAT NOW.
002552            GO TO 20
```

```
002553     C  DO MULLER-TRAUB ITERATION. SEE P212 OF TRAUB'S BOOK.
002554        11 CONTINUE
002555           NREDH = .FALSE.
002556           FX32=(FX3-FX2)/H
002557           FX321=(FX32-FX21)/(X3-X1)
002558           W=FX32+H*FX321
002559     C     IF W IS SO BIG THAT T WILL OVERFLOW, GIVE UP ON THIS ROOT.
002560           IF(ABS(REAL(W)).GT.1.E+150.OR.ABS(AIMAG(W)).GT.1.E+150) GO TO 33
002561           T=CSQRT(W*W-(4.,0.)*FX3*FX321)
002562           RHO=W+T
002563           T   =W-T
002564           IF( CABS2(T).GT.CABS2(RHO) ) RHO=T
002565           H=-2.*FX3/RHO
002566     C     EVEN IF SWR IS ORIGINALLY TRUE, IF WE SMELL AN OBVIOUS COMPLEX
002567     C     ROOT, WE TEMPORARILY MAKE SWR FALSE AND FIND IT.
002568           IF(.NOT.SWR) GO TO 12
002569           RH = ABS(REAL(H))
002570           AIH = ABS(AIMAG(H))/RH
002571           IF(RH.LE.1.E-02.AND.AIH.GE.1.E+02) SWR  = .FALSE.
002572           IF(.NOT.SWR) NREDH = .TRUE.
002573       115 IF( SWR ) H=REAL(H)
002574        12 ITC=ITC+1
002575           X1=X2
002576           X2=X3
002577           FX1=FX2
002578           FX2=FX3
002579           FX21=FX32
002580     C  IF NEW H IS TOO MUCH LARGER THAN LAST H REDUCE ITS MAGNITUDE TO THE
002581     C  MAXIMUM ALLOWED KEEPING ITS DIRECTION THE SAME.
002582           IF(.NOT.NREDH) GO TO 35
002583           IF(CABS2(H+X3).LT.RMX*RMX) GO TO 13
002584           XR = CABS2(H)/CABS2(X2-X1)
002585           GO TO 36
002586        35 XR = CABS2(H)/CABS2(X2-X1)
002587           IF(XR.LT.XRMAX*XRMAX) GO TO 13
002588        36 IF( SW2 ) PRINT    496, H
002589       496 FORMAT(30H (X3-X2)/(X2-X1) TOO LARGE. H=,2E22.13,17H WILL BE REDUC
002590          .ED.)
002591           H=H*(XRMAX/SQRT(XR))
002592        13 X3=X2+H
002593           IF(CABS2(X3 - P3).LT.ERRMAX*ERRMAX) GO TO 40
002594           IF(SW2) PRINT 41
002595        41 FORMAT(19H X3 TOO FAR FROM P3)
002596           I = RTC + 1
002597           GO TO 33
002598     C  CHECK FOR CONVERGENCE.
002599     C     .. DUE TO CLOSE RELATIVE SPACING OF ITERATES.
002600        40 IF(REAL(X2).EQ.0..AND.AIMAG(X2).EQ.0.) GO TO 45
002601           IF(CABS2(X3/X2 - 1.).LT.EP2*EP2) GO TO 22
002602        45 IF(CABS2(X3 - X2).LT.EP2*EP2) GO TO 22
002603           IF(REAL(X1).EQ.0..AND.AIMAG(X1).EQ.0.) GO TO 46
002604           IF(CABS2(X3/X1 - 1.).LT.EP2*EP2) GO TO 22
002605        46 IF(CABS2(X3 - X1).LT.EP2*EP2) GO TO 22
002606           ROOT=X3
002607           NRET=4
002608           GO TO 200
002609         4 FX3=FROOT
002610     C  IF NEW ITERATE GIVES MUCH LARGER FUNCTION MODULUS,
002611     C  HALVE THE STEP AND TRY AGAIN.
002612           IF(ABS(REAL(FX3)).GT.1.E+150.OR.ABS(AIMAG(FX3)).GT.1.E+150)GO TO
002613          . 44
002614           IF( CABS (FX3)/CABS (FX2).LT.FRMAX        ) GO TO 103
002615        44 IF( SW2 ) PRINT    497, X3,FUNC,FROOT
002616       497 FORMAT(36H FZ3/FZ2 TOO LARGE. Z3 ADJUSTED FROM,2E22.13/
002617          . 11H  FUNCTION=,2E22.13,20H, MODIFIED FUNCTION=,2E22.13)
002618           H=.5*H
002619           GO TO 13
002620     C  ITERATION COMPLETE.
002621       103 CONTINUE
002622     C  ITERATION OUTPUT
002623           I=RTC+1
002624           IF( SW2 ) PRINT    500, I,ITC,X3,FUNC,FROOT
002625       500 FORMAT(6H ROOT ,I3,12H, ITERATION ,I3,16H, ROOT ESTIMATE=,2E22.13/
002626          . 11H  FUNCTION=,2E22.13,20H, MODIFIED FUNCTION=,2E22.13)
002627     C    CHECK FOR CONVERGENCE
002628     C     .. DUE TO FUNCTION BEING VERY SMALL.
002629        20 IF( CABS2(FX3)+CABS2(FUNC).LE.EP2*EP2 ) GO TO 22
002630     C  NOT CONVERGED YET. SHOULD WE ITERATE AGAIN..
002631           IF(CABS2(X3).GT.RMX*RMX.AND.CABS2(H).LT..25.AND.ITC.GE.5)GO TO 33
002632     C    IF ROOT-SEARCH SEEMS TO BE GOING OFF TO INFINITY, STOP IT.
002633           IF(CABS2(X3).GT.RMX*RMX.AND.ITC.GE.5.AND.CABS2(X2-X1).GT..25.AND.
002634          .CABS2(H).GT.CABS2(X2-X1)) GO TO 33
002635           IF( ITC.LT.MXM ) GO TO 11
002636     C    IF WE SEEM TO BE GETTING CLOSE TO A ROOT, ALLOW 10 MORE ITERATIONS
002637     C    BEFORE GIVING UP.
002638           IF(ITC.LT.MXM+10 .AND. CABS2(FUNC).LT.1.E-06*FI2) GO TO 11
002639           IF(ITC.LT.MXM+20 .AND. CABS2(FUNC).LT.1.E-12*FI2) GO TO 11
002640     C    IF FUNCTION IS PRETTY SMALL, BUT MORE THAN EP2,, SEE IF EIGENMODE
002641     C    IS GOOD, AND IF IT IS, CONSIDER ROOT CONVERGED.
```

```
002642           IF(EIGV.EQ.0) GO TO 33
002643           IF (CABS2(FUNC).GT.1.E-14) GO TO 33
002644           CALL SAVPHI
002645           IF(CABS2(PHI(NRTSV,1) - (1.,0.)).LT..0001) GO TO 22
002646     C  NO, WE HAVE TRIED LONG ENOUGH.
002647        33 IF( SW1 ) PRINT   502, I, ITC
002648       502 FORMAT(28H FAILURE TO CONVERGE ON ROOT,I3,I10,11H ITERATIONS )
002649           CONV(I)=.FALSE.
002650           FAILC = FAILC + 1
002651     C  UNLESS SW4, FAILURE TO CONVERGE MEANS WE MUST STOP AFTER SETTING
002652     C  THE REST OF THE ARRAY CONV =.FALSE..
002653           IF( SW4 ) GO TO 22
002654           RTC=RTC+1
002655        43 RT(RTC)=X3
002656           DO 21 I=RTC,NRTS
002657        21 CONV(I)=.FALSE.
002658           RETURN
002659     C  RECORD THE FINDING OF A ROOT.
002660        22 CONTINUE
002661           IF(.NOT.SW7 .OR. .NOT.SW3 .OR..NOT. CONV(I)) GO TO 30
002662           L = 0
002663           IF (REAL(X3).LT.0..OR. AIMAG(X3).LT.0.) L = 1
002664           RX3 = ABS(REAL(X3))
002665           AX3 = ABS(AIMAG(X3))
002666           X3 = CMPLX(RX3, AX3)
002667           IF (L.EQ. 1) FUNC = F(X3)
002668        30 RTC=RTC+1
002669           RT(RTC)=X3
002670           IF (SW1 .AND..NOT. NONEG .AND..NOT. NOCONJ) PRINT 501,RTC,X3,ITC
002671       501 FORMAT(6H ROOT ,I3,4H IS=,2E22.13,1H,,I5,11H ITERATIONS)
002672           IF (.NOT. CONV(RTC)) GO TO 5304
002673           IF(NONEG.OR.NOCONJ) GO TO 5305
002674           IF(EIGV.EQ.0) GO TO 5304
002675        24 CALL SAVPHI
002676           NRTSU = NRTSV
002677     C    IF PHI(1)  DOES NOT EQUAL 1,  THEN SOMETHING IS WRONG WITH OUR
002678     C    CRITERION FOR HAVING FOUND A ROOT
002679           PHJ = PHI(NRTSU, 1)
002680           IF(ABS(AIMAG(PHJ)).LE..01.AND.ABS(REAL(PHJ)-1.).LE..01) GO TO 38
002681           IF(SW1) PRINT 513, PHJ
002682       513 FORMAT (37H ROOT NOT GOOD, UNNORMALIZED PHI(1) = , 2E12.4)
002683           CONV(I) = .FALSE.
002684           FAILC = FAILC + 1
002685           GO TO 5304
002686        38 IF(EIGV.GT.3) GO TO 39
002687           IF(AIMAG(RT(RTC)).LT.1.E-05.OR.EIGV.LT.3) GO TO 34
002688     C    PHI(X) FOR THIS ROOT WILL BE PLOTTED AT END OF RUN.
002689        39 PHI(NRTSV, NPP+1)= RT(RTC)
002690           PHI(NRTSV, NPP+2) = CMPLX(KYAI,0.)
002691           NRTSV = NRTSV + 1
002692        34 PHJ = PHI(NRTSU,IZ)
002693     C    CALCULATE AND PRINT OUT ABS.VAL. AND ARG. OF PHI(P) S.
002694           SUM = 0.
002695           DO 26 I = 1, NPP
002696           SUM = SUM + CABS2(PHI(NRTSU,I))
002697           ABPHI(I) = CABS(PHI(NRTSU,I))
002698           IF (ABPHI(I).LE.1.E-20) GO TO 32
002699           ARPHI (I) = 57.296*AIMAG(CLOG(PHI(NRTSU, I)))
002700           GO TO 26
002701        32 ARPHI(I) = 0.
002702        26 CONTINUE
002703     C    PHIZ = .TRUE. IFF PHI(0) = 0, THE PURPOSE OF PHIZ IS EXPLAINED IN *ROOTS*
002704     C    FOLLOWING STATEMENT 76
002705           IF(CABS2(PHJ)/SUM.GT.1.E-11) PHIZ(RTC) = .FALSE.
002706           PHPLUS = CABS2(PHI(NRTSV,IZ+1))
002707           PHMIN = CABS2(PHI(NRTSV,IZ-1))
002708           IF(CABS2(PHJ).GT.1.E-04*PHPLUS) PHIZ(RTC) = .FALSE.
002709           IF(CABS2(PHJ).GT.1.E-04*PHMIN) PHIZ(RTC) = .FALSE.
002710           ABNM = SQRT(SUM)
002711           I = IZ
002712           ARNM = ARPHI(I)
002713           IF (ABPHI(I) .LE. .1E-03) ARNM = .5*ARPHI(NPP)
002714           DO 31 I = 1, NPP
002715           ABPHI(I) = ABPHI(I)/ABNM
002716           ARPHI(I) = ARPHI(I) - ARNM
002717           IF (ARPHI(I) .LE.-180.) ARPHI(I) = ARPHI(I) + 360.
002718           IF (ARPHI(I) .GT. 180.) ARPHI(I) = ARPHI(I) - 360.
002719           IF(NRTS.GT.2) GO TO 31
002720     C    DON'T CALCULATE PHINEW UNLESS THIS IS THE FIRST GOOD ROOT WE HAVE FOUND.
002721           IF(NOPHNU) GO TO 31
002722           IF(PHZP.NE.SGNL(1.,PHIZ(RTC))) GO TO 31
002723           IF(AIMAG(RT(RTC)).LT.-1.E-06.AND.SW31) GO TO 31
002724           IF( REAL(RT(RTC)).LT.-1.E-06.AND.SW71) GO TO 31
002725     C    PHINEW IS USED BY *FOLLOW* IN COMPARING EIGENFUNCTIONS OF OLD AND NEW KYAI.
002726        42 PHINEW(I) = ABPHI(I)*CEXP(ARPHI(I)*(0.,1.)/57.296)
002727           IF(I.EQ.NPP) NOPHNU = .TRUE.
002728        31 CONTINUE
002729           IF(.NOT.SW1) GO TO 5304
002730           PRINT 507, (ABPHI(I), I = 1, NPP)
002731       507 FORMAT (10E12.4/10E12.4)
002732           PRINT 506, (ARPHI(I), I = 1, NPP)
002733       506 FORMAT (10F12.2/10F12.2)
002734      5304 IF(.NOT.SW5) GO TO 47
```

```
002735          START = ABS(REAL(X3))
002736          IF(START.GE.RMX) START = 1.5
002737       47 IF(ITC.GT.0) GO TO 5305
002738     C    IF ONE OF THE STARTING POINTS WAS A ROOT (I.E. ITC = 0), THEN CHANGE
002739     C    THE STARTING POINTS, OTHERWISE WE WILL GET THE SAME ROOT AGAIN NEXT
002740          CHANGE = .15*(P2 - P1)
002741          P1 = P1 + CHANGE
002742          P2 = P2 + CHANGE
002743          P3 = P3 + CHANGE
002744     5305 CONTINUE
002745     C  QUIT IF ENOUGH ROOTS ARE FOUND, OR IF WE ARE ALMOST OUT OF TIME.
002746          IF(WARN(0)) RETURN
002747          IF( RTC.GE.NRTS ) GO TO 300
002748          IF(FAILC.GT.NFL) RETURN
002749     C  IF SW7, THE NEGATIVE CONJUGATE OF A CONVERGENT ROOT FOUND IS TO BE
002750     C   REGARDED AS A ROOT.
002751       28 IF (.NOT. SW7 .OR. NONEG .OR. ABS(REAL(X3)).LT.EP1) GO TO 29
002752          IF(SW1) PRINT 514
002753      514 FORMAT(62H NEGATIVE CONJUGATE OF LAST ROOT WILL BE CONSIDERED NEXT
002754         , ROOT.    )
002755          ITC = 0
002756          NONEG = .TRUE.
002757          X3 = -1.*CONJG(X3)
002758          ROOT = X3
002759          IF(.NOT.CONV(RTC)) CONV(RTC+1) = .FALSE.
002760          PHIZ(RTC+1) = PHIZ(RTC)
002761          GO TO 30
002762       29 NONEG = .FALSE.
002763     C  IF SW3, THE CONJUGATE OF A CONVERGENT NON-REAL ROOT IS TO BE
002764     C  REGARDED AS A ROOT ALSO, IF NOT ALREADY USED.
002765       23 IF( .NOT.SW3 .OR. ABS(AIMAG(X3)).LT.EP1 .OR. NOCONJ ) GO TO 10
002766          IF( SW1 ) PRINT   503
002767      503 FORMAT(53H CONJUGATE OF LAST ROOT WILL BE CONSIDERED NEXT ROOT.)
002768          ITC=0
002769          NOCONJ=.TRUE.
002770          X3=CONJG(X3)
002771          ROOT=X3
002772          IF(.NOT.CONV(RTC)) CONV(RTC+1) = .FALSE.
002773          PHIZ(RTC+1) = PHIZ(RTC)
002774          NRET=5
002775     C      GO TO 200
002776          5 GO TO 30
002777     C
002778     C   EVALUATES FROOT
002779     C   WHERE (ROOT-RT(1))*...*(ROOT-RT(RTC))*FROOT=F(ROOT)
002780     C   RETURNS TO STATEMENT NUMBER NRET.
002781     C   SPECIAL TREATMENT WHEN ROOT IS CLOSE TO AN EARLIER FOUND ROOT IS
002782     C   DESCRIBED AT HEAD OF PROGRAM.
002783       200 CONTINUE
002784          IF(WARN(0)) RETURN
002785          FUNC=F(ROOT)
002786          IF( SWR ) FUNC=REAL(FUNC)
002787          DNR=(1.,0.)
002788          IF(RTC.EQ.0) GO TO 202
002789          TEST=EP3*EP3*CABS2(ROOT)
002790          DO 203 I=1,RTC
002791          IF( CABS2(ROOT-RT(I)).GE.TEST ) GO TO 201
002792          T=.3333333333*(P1+P2+P3)+EP1
002793          IF( NRET.NE.1 ) GO TO 210
002794          P1=2.*P1-T
002795          ROOT=P1
002796          GO TO 200
002797       210 IF( NRET.NE.2 ) GO TO 211
002798          P2=2.*P2-T
002799          ROOT=P2
002800          GO TO 200
002801       211 IF( NRET.NE.3 ) GO TO 212
002802          P3=2.*P3-T
002803          ROOT=P3
002804          GO TO 200
002805       201 DNR=DNR*(ROOT-RT(I))
002806       203 CONTINUE
002807       202 FROOT=FUNC/DNR
002808          GO TO (1,2,3,4,5), NRET
002809       212 IF( SW1 ) PRINT   504
002810       504 FORMAT(71H ITERANT CLOSE TO ROOT FOUND BEFORE. ACCEPT IT AS A ROOT
002811          , WITHOUT TEST.)
002812          FROOT=0.
002813          GO TO 22
002814     C
002815     C   TERMINATION PROCEDURE.
002816     C   INCLUDES CAPABILITY OF REDOING 'BAD ROOTS' -ONES WHICH HADN'T
002817     C   CONVERGED.
002818     C
002819     C   HOW MANY GOOD ROOTS ARE THERE..
002820       300 NCR=0
002821          DO 301 I=1,NRTS
002822          IF( CONV(I) ) NCR=NCR+1
002823       301 CONTINUE
002824     C   IF ALL ROOTS ARE GOOD, OR IF WE HAVEN'T BEEN ASKED TO REDO BAD ONES,
002825     C   OR IF WE HAVE HAD NO LUCK THE LAST TIME THROUGH THE ROOTFINDING CODE,
002826     C   WE QUIT.
002827          IF( NCR.EQ.NRTS .OR. .NOT.SW4 .OR. NCR.LE.LNCR ) RETURN
```

```
002828      C  FIND FIRST BAD ROOT.
002829            DO 302 I=1,NRTS
002830            IF( .NOT.CONV(I) ) GO TO 303
002831          302 CONTINUE
002832      C   IF IT IS THE LAST ROOT WE WOULD ONLY BE REPEATING THE SAME NON-
002833      C   CONVERGENT ITERATIONS IF WE TRIED AGAIN.
002834          303 IF( I.EQ.NRTS ) RETURN
002835      C   MOVE LAST ROOT INTO 1ST BAD ROOTS PLACE.
002836      C   GO BACK AND TRY TO FIND A NEW LAST ROOT. GOOD LUCK.
002837            IF( SW1 ) PRINT   505,  I
002838          505 FORMAT(48H WILL ATTEMPT TO RE-DO UNCONVERGED ROOT ESTIMATE,I3)
002839            X3=RT(NRTS)
002840            RT(NRTS)=RT(I)
002841            RT(I)=X3
002842            CONV(I)=CONV(NRTS)
002843            RTC=NRTS-1
002844            LNCR=NCR
002845            CONV(NRTS)=.TRUE.
002846            GO TO 28
002847      C
002848            END
```

```
002849
002850            SUBROUTINE EIGPLT
002851      C     PLOTS EIGENFUNCTIONS
002852            COMPLEX OMEGA, PHI(18,22), A(20,20), F(100),IPXKO
002853            INTEGER PMAX
002854            REAL KYAI
002855            LOGICAL GOOD(18)
002856            DIMENSION X(100), Y(100)
002857            DATA KYAI/0./
002858            DATA NDY /10/
002859            COMMON/CIO/ICR,IHSP,IGRAPH
002860            COMMON/PHIC/ A,NRTSV,PHI
002861            COMMON/PMX/NP
002862            COMMON/FOLEIG/ GOOD, IBRANCH(18)
002863            COMMON/TVPOOL/XMIN,XMAX,YMIN,YM   ,TVXMIN,TVXMAX,TVYMIN,TVYMAX
002864            COMMON/TVTUNE/LPENON,LPENOFF,ITALICS,IWINK,INTENSE,IRIGHT,IUP
002865            NDY1 = NDY - 1
002866            CALL PAGE
002867            PRINT 10, NRTSV
002868         10 FORMAT(8H NRTSV =,I4)
002869            PRINT 11, GOOD
002870         11 FORMAT(2015)
002871            IF(NRTSV.EQ.1) RETURN
002872            NRTSU = NRTSV - 1
002873            PMAX = (NP-1)/2
002874            PI = 3.141592653589
002875            DO 3 N = 1, NRTSU
002876            IF(.NOT.GOOD(N)) GO TO 3
002877            OMEGA = PHI(N     ,NP+1)
002878            KYAI = REAL(PHI(N,NP+2))
002879            SUM = 0.
002880            DO 6 J = 1,NP
002881          6 SUM = SUM + CABS2(PHI(N,J))
002882            SUM = SQRT(SUM)
002883            DO 7 J = 1, NP
002884          7 PHI(N,J) = PHI(N,J)/SUM
002885            DO 1 I = 1, 100
002886            AI = I
002887            X(I) = AI/100.
002888            XKO = 2.*PI*X(I)
002889            F(I) = (0.,0.)
002890            DO 2 J = 1, NP
002891            P = J - PMAX - 1
002892            IPXKO = CMPLX(0.,XKO*P)
002893          2 F(I) = F(I) + PHI(N,J)*CEXP(IPXKO)
002894          1 Y(I) = F(I)*CONJG(F(I))
002895            YM = Y(1)
002896            DO 5 I = 2,100
002897          5 YM = AMAX1(YM,Y(I))
002898            XMAX = 1.
002899            YMIN = 0.
002900            XMIN = 0.
002901            CALL MAPS(XMIN,XMAX,YMIN,YM,0.11328,1.0,0.15,1.0)
002902            CALL TRACE(X,Y,100)
002903            CALL SETCH(1.,2.,1,0,1,0,0)
002904            WRITE(IGRAPH,4) OMEGA
002905          4 FORMAT(34H PLOT OF PHI**2 VS. X FOR OMEGA = ,2E22.13)
002906            WRITE(IGRAPH,8) KYAI, IBRANCH(N)
002907          8 FORMAT(8H KYAI = ,E12.6,10X,7H BRANCH,I4)
002908            CALL FRAME(1)
002909      C     DO PLOT OF PHI(X,Y)
002910            FM = SQRT(YM)
002911            YM = NDY
002912            CALL MAPS(XMIN,XMAX,YMIN,YM,0.11328,1.0,0.15,1.0)
002913            DO 12 K = 1, NDY1
002914            FLK = K
```

```
002915           PHASE = 2.*3.1415926535*FLK/YM
002916           DO 13 I = 1,100
002917        13 Y(I) = FLK + REAL(F(I)*CEXP((0.,1.)*PHASE))/FM
002918        12 CALL TRACE(X,Y,100)
002919           CALL SETCH(1.,1.,1,0,1,0,0)
002920           WRITE(IGRAPH,14)
002921        14 FORMAT(51H PLOT OF PHI(X,Y) FOR ONE WAVELENGTH IN Y DIRECTION)
002922           CALL FRAME(1)
002923           PRINT 9, OMEGA, KYAI, IBRANCH(N)
002924         9 FORMAT(35H EIGENFUNCTION PLOTTED FOR OMEGA = ,2E10.2,8H KYAI = ,E1
002925          .0.3,10X,7H BRANCH,I4)
002926         3 CONTINUE
002927           RETURN
002928           END
```

```
002929
002930           SUBROUTINE PRNPLT(X,Y,XMAX,XINCR,YMAX,YINCR,ISX,ISY,NPTS)
002931           COMMON/CIO/ ICR,IHSP,IGRAPH
002932           DIMENSION X(NPTS),Y(NPTS),IGRID(105),XAXIS(11)
002933           INTEGER BLANK,DOT,STAR,IGRID,PLUS
002934           DATA BLANK,DOT,STAR,PLUS / 1H ,1H.,1H*,1H+ /
002935     901   FORMAT(14X,105A1)
002936     902   FORMAT(1XE10.2,2X,1H+,105A1,1H+)
002937     903   FORMAT(15X,103(1H.))
002938     904   FORMAT(7X,11(F10.0),2H (,I4,5H PTS)  )
002939     905   FORMAT(16X,11(1H+,9X))
002940    9800   FORMAT(46H1SCALING ERROR IN PRNPLT, EXECUTION TERMINATED )
002941           IF(XINCR.EQ.0..OR.YINCR.EQ.0.) GO TO 800
002942           YAXMIN=0.01*YINCR
002943           XAXMIN=0.01*XINCR
002944           IZERO=YMAX/YINCR+1.5
002945           JZERO=103.5-XMAX/XINCR
002946           IF(JZERO.GT.103.OR.JZERO.LT.4) JZERO=2
002947           PRINT 905
002948           PRINT 903
002949           DO 10 I=1,51
002950           IF ( I.NE.IZERO) GO TO 16
002951           DO 14 J=1,105
002952     14    IGRID(J)=PLUS
002953           GO TO 15
002954     16    DO 11 J=1,105
002955     11    IGRID(J)=BLANK
002956     15    IGRID(JZERO)=PLUS
002957           IGRID(104)=DOT
002958           IGRID(2)=DOT
002959           DO 12 K=1,NPTS
002960           ITEST =(YMAX-Y(K))/YINCR+1.5
002961           IF(ITEST .NE.I) GO TO 12
002962           J=103.5-(XMAX-X(K))/XINCR
002963           IF(J.GT.103) J=105
002964           IF(J.LT.3) J=1
002965           IGRID(J)=STAR
002966     12    CONTINUE
002967           IF(MOD(I,10).EQ.1) GO TO 13
002968           PRINT 901,IGRID
002969           GO TO 10
002970     13    YAXIS=YMAX-(I-1)*YINCR
002971           IF(ABS(YAXIS).LT.YAXMIN) YAXIS=0.
002972           PRINT 902,YAXIS,(IGRID(J),J=1,105)
002973     10    CONTINUE
002974           PRINT 903
002975           PRINT 905
002976           DO 20 M=1,11
002977           XAXIS(M)=XMAX-XINCR*(FLOAT(11-M))*10.0
002978           IF(ABS(XAXIS(M)).LT.XAXMIN)XAXIS(M)=0.
002979     20    CONTINUE
002980           PRINT 904,XAXIS,NPTS
002981           RETURN
002982     800   PRINT 9800
002983           CALL EXIT
002984           END
```

```
002985
002986           SUBROUTINE TRACET(TEXTURE,X,Y,NPT)
002987           REAL TEXTURE,X(1),Y(1)
002988           INTEGER NPT
002989           IF(TEXTURE.EQ.0.0) CALL TRACE(X,Y,NPT)
002990           IF(TEXTURE.NE.0.0) CALL TRACEP(X,Y,NPT,3)
002991           RETURN
002992           END
```

```
002993
002994          FUNCTION SGNL(A,L)
002995          LOGICAL L
002996          IF(L) B = -1.
002997          IF(.NOT.L) B = 1.
002998          SGNL = SIGN(A,B)
002999          RETURN
003000          END


003001
003002          LOGICAL WARN
003003          FUNCTION WARN(N)
003004          WARN = .FALSE.
003005          RETURN
003006          END
```

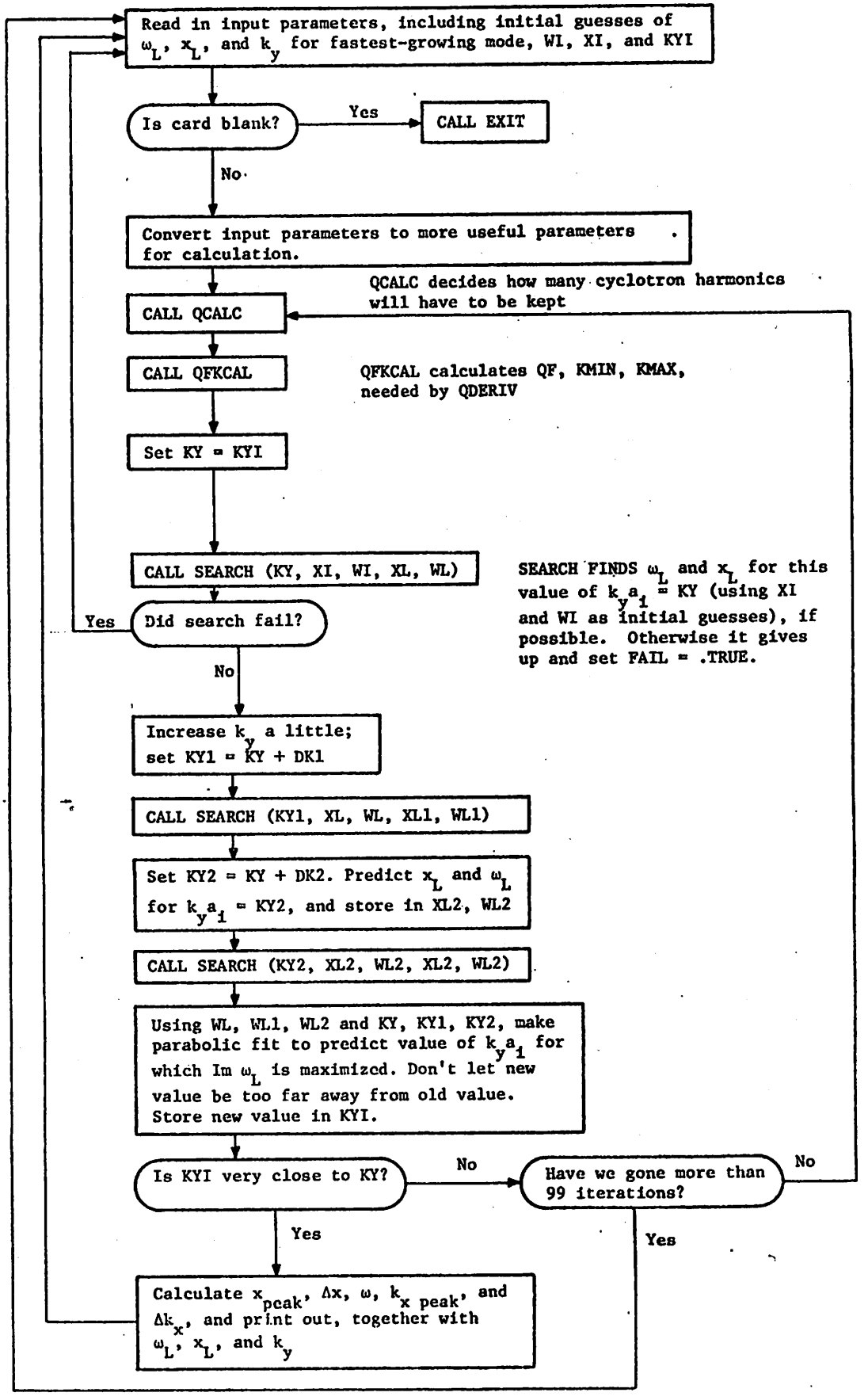
An outline of the program LOCAL and its subroutines is shown in the flow chart. LOCAL was used to find the normal modes and frequencies using the local method. In contrast to ROOTS, which was used for the nonlocal method, all branches $\omega(k_y)$ were not calculated, and "smart" root-finders and branch-followers were not used in LOCAL. Instead, a Newton-Raphson iteration scheme was used to find the solutions $x = x_L$ and $\omega = \omega_L$ to the simultaneous equations $D(x,\omega,k_x = 0) = 0$ and $\partial D/\partial x = 0$. Only the mode with the highest growth rate was sought. The Newton-Raphson iteration method worked well only if initial guesses of x_L and ω_L were reasonably close to the correct solutions. The mode found as the fastest-growing mode was really only the local maximum of Im $\omega(k_y)$ closest to the initial value of k_y chosen. The limitations in LOCAL were tolerable because ROOTS could be used to make good estimates of x_L, ω_L and k_y for initial guesses in LOCAL.

The routine QDERIV shown in the flow chart was similar to the subroutine DISP used with ROOTS, but instead of calculating det $A_{p,p'}$ it calculated

$$D(x,\omega) = G_0(\omega) + G_{-1}(\omega)\exp(-ik_0x) + G_{+1}(\omega)\exp(ik_0x)$$

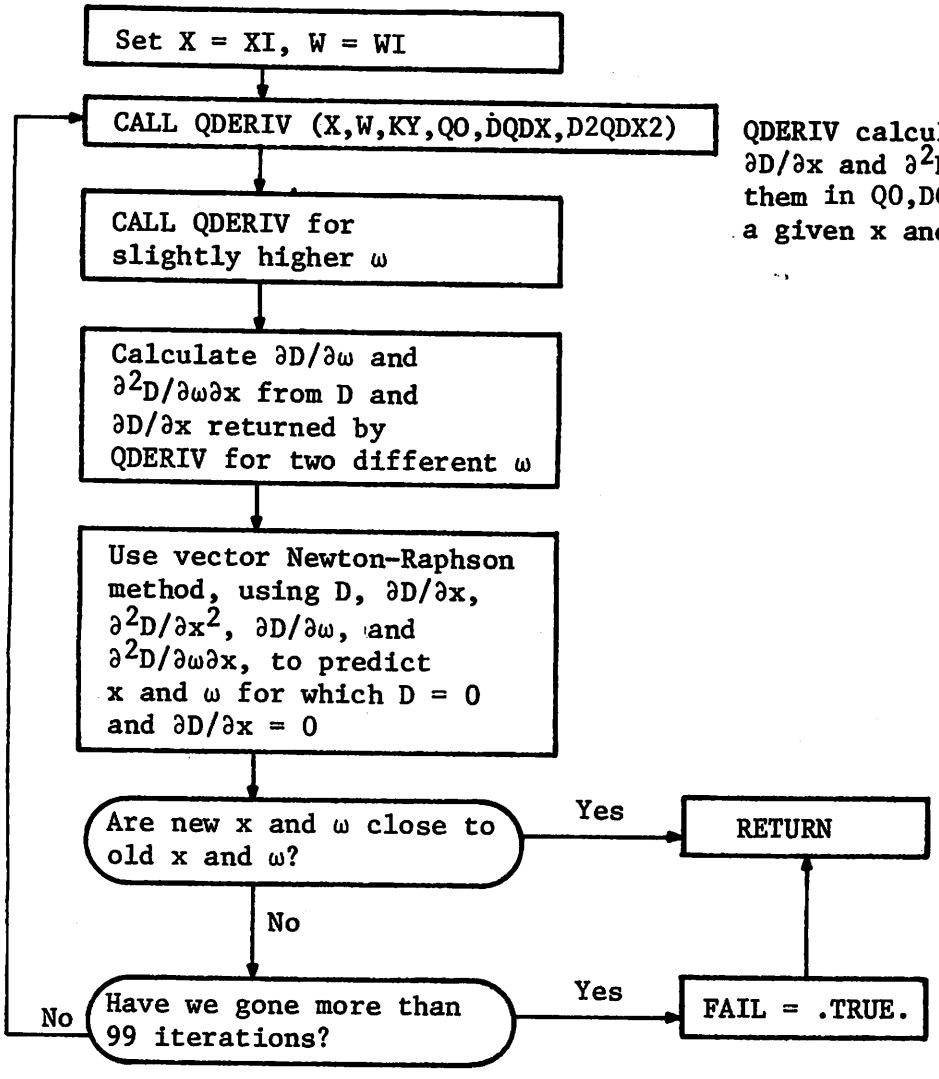
(with $k_x = 0$ assumed). Since the G's do not depend on x, the derivatives $\partial D/\partial x$ and $\partial^2 D/\partial x^2$, needed for the Newton-Raphson method, could be easily found analytically, and were also returned by QDERIV. On the other hand, the derivatives $\partial D/\partial\omega$ and $\partial^2 D/\partial x\partial\omega$ were found by calling QDERIV twice using slightly different values of ω.

FLOW CHART OF LOCAL



FLOW CHART OF SEARCH (KY, XI, WI, XL, WL)

KY is $k_y a_i$. XI and WI are the initial guesses for x_L AND ω_L (normalized to ω_{ci} and $2\pi/k_0$). XL and WL are final values of x_L and ω_L returned by SEARCH



QDERIV calculates $D(x,\omega,k_x = 0)$ $\partial D/\partial x$ and $\partial^2 D/\partial x$ and stores them in QO,DQDX, and D2QDX2, for a given x and ω (stored in X and W).

Listing of LOCAL and Subroutines

Like ROOTS, LOCAL runs on the A-machine of the National Magnetic Fusion Energy Computer Center. However, it is not believed to contain any non-standard Fortran, except for the PROGRAM, READ, FORMAT, and FUNCTION statements.

The subroutines QFKCAL, QCALC, BESSI, BESSJ, CABS2, ZEE, ABCALC, MAXW, RING, and COLD, used by LOCAL, have been already listed with ROOTS, so they are not shown here.

```
C-170A     BOX BOS      18.55.58     09/16A        1976
000001           PROGRAM LOCAL (INPUT,HSP,OUTPUT,TAPE10=OUTPUT)
000002           REAL KY,KOAI,KOSQ,KY1,KYNEW,KY1,KY2
000003           COMPLEX XI,WI,XL1,WL1,XL2,WL2,WIDTH   ,XLNEW,WLNEW
000004           COMPLEX SLPX1,SLPX2,SLPW1,SLPW2,D2XDK2,D2WDK2
000005           COMPLEX DQDX,D2QDX2,DQDW,XL,WL,QO,DELW,DQDKXS
000006           INTEGER QMAX
000007           LOGICAL FAIL
000008           LOGICAL UNMAG
000009           COMMON/CMAG/UNMAG
000010           COMMON/DIS/QF,KYSQ,      KOSQ,EPSF(4),XXX,KYAI,KOAI,XXXX,PMAX,
000011          .OMPSQ,AEAI,KMAX(4),AMASQ(4),XXXXX(4),KMIN(4)
000012           COMMON/CDFU/ DFU(4),  YYY(4)
000013           COMMON/CSPEC/ RLARM(4),AMASS(4),CHARG(4),JAY(4),DENSE(4),NSPEC
000014           COMMON/CFAIL/ FAIL
000015           COMMON/CQDER/ QO,DQDX,D2QDX2,DQDW,DQDKXS
000016           COMMON/CRBESJ/ QMAX(4),NMAX(4)
000017           DATA AEAI/0./,DFU/4*0./,JAY/4*0/
000018           DATA RLARM/4*1./,AMASS/4*1./,CHARG/4*1./,DENSE/4*1./,AMASQ
000019          ./4*1./
000020           DATA ICR/2/
000021           PRINT 17
000022        17 FORMAT(1H1)
000023           GO TO 18
000024        15 CONTINUE
000025        18 READ (ICR,16) AIL, R, OMPSQ,XI,WI,KYI,UNMAG
000026        16 FORMAT(8F5.0,11)
000027           IF(AIL.EQ.0.) CALL EXIT
000028           IF(XI.NE.0..OR.WI.NE.0..OR.KYI.NE.0.) GO TO 20
000029           IF(FAIL) GO TO 15
000030           WI = WL
000031           XI = XL
000032           KYI = KY
000033        20 IONS = 5H MAG.
000034           IF(UNMAG) IONS = 6H UNMAG
000035           GAM = 1.
000036           IF(R.LE.1.) GAM = 0.
000037           RLARM(2) = 1./SQRT(R)
000038           RLARM(3) = 0.
000039           AMASS(3) = 1./3700.
000040           CHARG(3) = -1.
000041           DENSE(1) = R/(R - GAM)
000042           DENSE(2) = -GAM/(R-GAM)
000043           NSPEC = 3
000044           AMASQ(3) = AMASS(3)**2
000045        19 KOAI = 2.*3.1415926535*AIL
000046           KOSQ = KOAI**2
000047           EPSI = .999
000048           EPSI = .99
000049           EPSE = EPSI*(DENSE(1)*EXP(-.5*KOSQ) + DENSE(2)*EXP(-.5*KOSQ*RLARM(
000050          .2)**2))
000051           DO 1 I = 1,2
000052         1 EPSF(I) = .5*EPSI*OMPSQ*DENSE(I)
000053           EPSF(3) = .5*EPSE*OMPSQ/AMASS(3)
000054           ITER = 0
000055         5 KY = KYI
000056           RHIGH = 2.*REAL(WI)
000057           MINQ = RHIGH
000058           CHI = KY*KY
000059           DO 9 I = 1, NSPEC
000060           IF(RLARM(I).EQ.0.) GO TO 9
000061           CHI2 = CHI*RLARM(I)**2
000062           IF(JAY(I).EQ.0.) GO TO 10
000063           CALL QCALC(CHI2,QMAX(I),NMAX(I),1)
000064           GO TO 9
000065        10 CALL QCALC(CHI2,QMAX(I),NMAX(I),0)
000066           NMAX(I) = NMAX(I) + JAY(I)
000067         9 CONTINUE
000068           CALL QFKCAL(QF,KMAX,KMIN,RHIGH,0.)
000069           CALL SEARCH(KY,XI,WI,XL,WL)
000070           IF(FAIL) GO TO 13
000071           DK1 = .001*KY
000072           DK2 = .01*KY
000073           KY1 = KY + DK1
000074           KY2 = KY1+ DK2
000075           WIDTH = CSQRT(2.*DQDKXS*AIL*AIL/D2QDX2)
000076           DELW = -.5*WIDTH*D2QDX2/DQDW
000077           WIDTH = CSQRT(WIDTH)
000078           CALL SEARCH(KY1,XL,WL,XL1,WL1)
000079           IF(FAIL) GO TO 13
000080           XL2 = 11.*XL1 - 10.*XL
000081           WL2 = 11.*WL1 - 10.*WL
000082           CALL SEARCH (KY2,XL2,WL2,XL2,WL2)
000083           IF(FAIL) GO TO 13
000084           SLOPE1 =(AIMAG(WL1) - AIMAG(WL))/DK1
000085           SLOPE2 =(AIMAG(WL2) - AIMAG(WL1))/DK2
000086           IF(SLOPE2.GE.SLOPE1) GO TO 3
000087           D2QDK2 = 2.*(SLOPE2-SLOPE1)/(DK1+DK2)
000088           KYNEW = KY + .5*DK1 - SLOPE1/D2QDK2
000089           IF(ABS(KYNEW - KY).GT..1*KY) GO TO 3
000090           IF(ABS(KYNEW-KY).LT..005*KY) GO TO 4
000091         7 ITER = ITER + 1
```

```
000092          IF(ITER.GT.99) GO TO 13
000093          SLPX1 = (XL1-XL)/DK1
000094          SLPX2 = (XL2-XL1)/DK2
000095          D2XDK2 = 2.*(SLPX2 - SLPX1)/(DK1+DK2)
000096          SLPW1 = (WL1 - WL)/DK1
000097          SLPW2 = (WL2 - WL1)/DK2
000098          D2WDK2 = 2.*(SLPW2-SLPW1)/(DK1+DK2)
000099          XLNEW = .5*(XL+XL1) + SLPX1*(KYNEW - .5*(KY+KY1)) + .5*D2XDK2*
000100         .(KYNEW - .5*(KY+KY1))**2
000101          IF(CABS(XLNEW-XL).GT..1) GO TO 6
000102          WLNEW = .5*(WL+WL1) + SLPW1*(KYNEW - .5*(KY+KY1)) + .5*D2WDK2*
000103         .(KYNEW - .5*(KY+KY1))**2
000104           IF(CABS(WLNEW/WL - 1.).GT..2) GO TO 6
000105          KYI = KYNEW
000106          XI = XLNEW
000107          WI = WLNEW
000108          GO TO 5
000109        3 IF(SLOPE2.LE.0.) KYNEW = .9*KY
000110          IF(SLOPE2.GT.0.) KYNEW = 1.1*KY
000111          GO TO 7
000112        4 WL1 = WL + .5*DELW
000113          WL2 = WL1 + DELW
000114          XCORR = CABS(WIDTH)
000115          XPEAK = REAL(XL) + AIMAG(XL)*AIMAG(WIDTH**2)/REAL(WIDTH**2)
000116          PEAKK = AIMAG(XL)/REAL( WIDTH**2)/(2.*3.1415926535)
000117          F = SQRT(ALOG(2.))*2.
000118          HWX = F/SQRT(REAL(1./WIDTH**2))
000119          HWK = F/SQRT(REAL(WIDTH**2))/(2.*3.1415926535)
000120          PRINT 14, AIL, R, OMPSQ, IONS
000121       14 FORMAT(//"AIL = ",E14.5," R = ",E14.5,", OMPSQ = ",E14.5,", MASS
000122         .RATIO = 3700. ",A6," IONS")
000123          PRINT 8, WL,KY,XL,XCORR,WIDTH,WL1,WL2
000124        8 FORMAT(" ACCORDING TO LOCAL APPROX., FASTEST-GROWING MODE HAS OMEG
000125         .A = ",2E14.5," KYAI = ",E14.5/"LOCATED AT X/L = ",2E14.5," WITH
000126         .CORRELATION LENGTH ",E14.5,"(",E14.5,"+",E14.5," I)"/" OMEGA CORRE
000127         .CTED FOR FINITE WIDTH (WKB APPROX.) IS ",2E14.5/" AND THE NEXT WKB
000128         . MODE HAS OMEGA = ",2E14.5)
000129          PRINT 12, XPEAK,HWX,PEAKK,HWK
000130       12 FORMAT(" ABS.VAL.PHI(X) SQUARED HAS PEAK AT X/L = ",E14.5,", AND H
000131         .ALF-WIDTH ",E14.5/"ABS.VAL.PHI(KX) SQUARED HAS PEAK AT KX/KO = ",
000132         .E14.5," AND HALF-WIDTH ",E14.5)
000133          GO TO 15
000134        6 KYNEW = .5*(KYNEW + KY)
000135          GO TO 7
000136       13 PRINT 14,AIL,R,OMPSQ,IONS
000137          GO TO 15
000138          END

000139
000140          SUBROUTINE SEARCH (KY,XI,WI,XL,WL)
000141          LOGICAL FAIL,NDPOLE
000142          COMPLEX XI,WI,XL,WL,X,W,QO,Q1,Q2,DQDX,DQDW,D2QDX2,D2QDXW,DQDX1,DET
000143          COMPLEX DQDKXS
000144          COMPLEX XDIFF,WDIFF
000145          REAL KY,KXSQ
000146          COMMON/CFAIL/ FAIL
000147          COMMON/CQDER/ QO,DQDX,D2QDX2,DQDW,DQDKXS
000148          COMMON/CNDP/ NDPOLE
000149          COMMON/CKX/ KXSQ
000150          NDPOLE = .FALSE.
000151          KXSQ = 0.
000152          FAIL = .FALSE.
000153          ITER = 0
000154          X = XI
000155          W = WI
000156        2 CALL QDERIV(X,W,KY,QO,DQDX,D2QDX2)
000157          CALL QDERIV(X,W+.001,KY,Q1,DQDX1,Q2)
000158          DQDW = 1000.*(Q1-QO)
000159          D2QDXW = 1000.*(DQDX1 - DQDX)
000160          DET = DQDX*D2QDXW - D2QDX2*DQDW
000161          XDIFF = (-D2QDXW*QO + DQDW*DQDX)/DET
000162          XL = X + XDIFF
000163          WDIFF = (D2QDX2*QO - DQDX*DQDX)/DET
000164          WL = W + WDIFF
000165          IF(CABS(XL - X).GT..005) GO TO 1
000166          IF(CABS(WL - W).GT..005) GO TO 1
000167          NDPOLE = .TRUE.
000168          CALL QDERIV(X,W,KY,QO,DQDX,D2QDX2)
000169          CALL QDERIV(X,W+.001,KY,Q1,DQDX1,DQDX1)
000170          KXSQ = .0001
000171          CALL QDERIV(X,W,KY,Q2,DQDX1,DQDX1)
000172          DQDW = 1000.*(Q1-QO)
000173          DQDKXS = 10000.*(Q2-QO)
000174          KXSQ = 0.
000175          NDPOLE = .FALSE.
000176          RETURN
000177        1 ITER = ITER + 1
000178          IF(ITER.GT.99) GO TO 4
000179          IF(CABS(XL - X).GT..1) GO TO 3
000180          IF(CABS(WL/W - 1.).GT..2) GO TO 3
000181          X = XL
000182          W = WL
000183          GO TO 2
000184        3 XL = .5*(XL + X)
000185          WL = .5*(WL + W)
000186          GO TO 1
000187        4 FAIL = .TRUE.
000188          PRINT 5, KY,XL,WL,QO,DQDX
000189        5 FORMAT("SEARCH FAILED", 9E12.4)
000190          RETURN
000191          END
```

```
000192
000193          SUBROUTINE QDERIV(X,W,KY,Q,DQDX,D2QDX2)
000194          COMPLEX X,W,OMEGA,GO,GPLUS,GMINUS, APP,BPP,AP,BP,IX,DEPOLE,OMEGD,U
000195          COMPLEX Q,DQDX,D2QDX2,EIKOX
000196          INTEGER PMAX,P
000197          REAL KY,KYSQ,KYAI,KOAI,KOSQ,KXSQ
000198          LOGICAL UNMAG,NDPOLE
000199          COMMON/CMAG/UNMAG
000200          COMMON/DIS/QF,KYSQ,      KOSQ,EPSF(4),XXX,KYAI,KOAI,XXXX,PMAX,
000201         .OMPSQ,AEAI,KMAX(4),AMASQ(4),XXXXX(4),KMIN(4)
000202          COMMON/CDFU/ DFU(4), YYY(4)
000203          COMMON/CSPEC/ RLARM(4),AMASS(4),CHARG(4),JAY(4),DENSE(4),NSPEC
000204          COMMON/CCHIP/CHIP(4),AP(4),BP(4),OMEGA,APP(4),BPP(4)
000205          COMMON/APBP/N,M,P
000206          COMMON/CNDP/ NDPOLE
000207          COMMON/CKX/KXSQ
000208          DATA PI /3.1415926535/
000209          KYAI = KY
000210          KYSQ = KY*KY
000211          OMEGA = W
000212          DO 103 I = 1, NSPEC
000213          OMEGD = OMEGA + (0.,1.)*KYSQ*DFU(I)
000214          IF(CABS2(OMEGD).EQ.0..AND.KOAI.NE.0.) GO TO 100
000215          IF(CABS2(OMEGD).EQ.0..AND.DFU(I).NE.0.) GO TO 100
000216          U = OMEGD*AMASS(I)/CHARG(I)
000217          AW = ABS(REAL(U))
000218          DW = 1.E-13*AW
000219          IF(ABS(AIMAG(U)).GT.DW) GO TO 103
000220          IW = IFIX(AW)
000221      100 OMEGA = OMEGA + 3.E-13*OMEGA
000222          IF(AW - IW.GT.DW.AND.IW + 1. - AW.GT.DW) GO TO 103
000223          GO TO 102
000224      103 CONTINUE
000225      C       DEPOLE IS USED TO GET RID OF ZEROS IN THE DENOMINATORS
000226      102 DEPOLE = (1.,0.)
000227          IF(NDPOLE) GO TO 5
000228          DO 2 I = 1, NSPEC
000229          OMEGD = OMEGA + (0.,1.)*KYSQ*DFU(I)
000230          IF(KOAI.EQ.0.) GO TO 23
000231          IF(I.EQ.1) GO TO 22
000232          IF(DFU(I).EQ.DFU(I-1)) GO TO 23
000233       22 DEPOLE = OMEGD*DEPOLE
000234       23 IF(KMAX(I).EQ.0) GO TO 2
000235          KMX = KMAX(I)
000236          JI = KMIN(I)
000237          DO 1 J = JI,KMX
000238        1 DEPOLE = DEPOLE*(OMEGD - J/AMASS(I))
000239        2 CONTINUE
000240      C       THE FACTOR QF IS PUT IN TO MAKE DEPOLE ON THE ORDER OF 1
000241          DEPOLE = DEPOLE*QF
000242        5 PMAX = 0
000243          P = 0
000244          N = 1
000245          M = 1
000246          CALL ABCALC
000247          GO = KYSQ + KXSQ
000248          DO 3 I = 1, NSPEC
000249        3 GO = GO + AP(I)*OMPSQ*CHARG(I)**2*DENSE(I)/AMASS(I)
000250          M = 2
000251          CALL ABCALC
000252          GPLUS = 0.
000253          GMINUS = 0.
000254          DO 4 I = 1, NSPEC
000255          GPLUS = GPLUS + EPSF(I)*(APP(I) + BPP(I))
000256        4 GMINUS = GMINUS + EPSF(I)*(AP(I) - BP(I))
000257          IX = (0.,2.)*3.1415926535*X
000258          EIKOX = CEXP(IX)
000259          Q = GO + EIKOX*GPLUS + GMINUS/EIKOX
000260          DQDX = 2.*PI*(EIKOX*GPLUS - GMINUS/EIKOX)*(0.,1.)
000261          D2QDX2 = -4.*PI*PI*(Q - GO)
000262          Q = Q*DEPOLE
000263          DQDX = DQDX*DEPOLE
000264          D2QDX2 = D2QDX2*DEPOLE
000265          RETURN
000266          END
```

APPENDIX J

The characteristics of the most unstable mode are shown for various values of density $\omega_{pi}^2/\omega_{ci}^2$ and density gradient $k_0 a_i$, using the local method (shown in Table I) and the nonlocal method (shown in Table II). In all cases the mirror ratio R = 3 and the mass ratio $m_i/m_e = 3700$. In the table headings, x_{peak} is the value of x at which $|\phi(x)|$ is greatest; Δx is the half-width of $|\phi(x)|^2$; $k_{x\ peak}$ is the value of k_x at which $|\tilde{\phi}(k_x)|^2$ is greatest; and "p_{max} needed" is the minimum value of p_{max} which is needed in order for the nonlocal method to be valid [defined by requiring $|\tilde{\phi}(\pm p_{max}\ k_0)| < |\tilde{\phi}(k_{x\ peak})|/4$]. The results in the two tables are seen to be in good agreement when inequality (21) is well-satisfied. The one case in which the agreement is not very good is for $\omega_{pi}^2/\omega_{ci}^2 = 1000$ and $k_0 a_i/2\pi = 0.40$. This set of parameters is near the border between regions D and E in Figure 6. Using the local method, the "region E instability" (at higher Re ω and k_y) has a slightly higher growth rate than the "region D instability," while the reverse is true using the nonlocal method; hence the huge discrepancy between the two tables for Re ω and $k_y a_i$ with this set of parameters.

TABLE I

$\frac{\omega_{pi}^2}{\omega_{ci}^2}$	$k_0 \frac{a_i}{2\pi}$	$\frac{Im\omega_L}{\omega_{ci}}$	$\frac{Re\omega_L}{\omega_{ci}}$	$k_y a_i$	$k_0 \frac{x_{peak}}{2\pi}$	$k_0 \frac{\Delta x}{2\pi}$	$\frac{k_{x\ peak}}{k_0}$	p_{max} needed
10000	0.015	3.55	6.75	19.4	0.56	0.010	29.1	74
	0.03	5.94	10.65	24.5	0.57	0.015	-3.2	36
	0.06	9.39	14.88	31.4	0.57	0.023	-15.9	39
	0.12	12.26	14.67	36.3	0.57	0.044	-13.8	27
	0.18	11.62	7.83	34.0	0.56	0.070	-7.0	14
	0.21	13.65	0.	22.1	0.50	0.085	-0.7	6
	0.24	12.50	0.	24.6	0.50	0.086	4.1	9
	0.27	6.75	0.	27.7	0.50	0.092	9.1	14
	0.30	3.10	0.	0.	0.	0.188	±12.2	14
	0.35	3.18	0.	0.	0.	0.230	±10.7	13
	0.40	6.10	30.70	27.3	0.17	0.117	5.5	9
	0.50	10.23	24.75	29.5	0.17	0.138	3.4	7
	0.60	9.35	17.79	29.0	0.18	0.169	2.6	5
	0.70	7.13	12.33	27.1	0.20	0.207	2.2	4
1000	0.015	1.35	2.65	9.8	0.61	0.019	25.8	49
	0.3	2.27	4.31	11.9	0.62	0.026	7.2	25
	0.06	3.73	6.45	14.8	0.63	0.036	-2.1	15
	0.12	5.47	7.18	17.7	0.62	0.063	-5.2	14
	0.18	5.70	4.20	17.4	0.58	0.101	-3.3	8
	0.21	6.85	0.	11.1	0.50	0.126	-0.3	4
	0.24	6.47	0.	12.8	0.50	0.124	-2.0	6
	0.27	3.57	0.	14.8	0.50	0.131	4.6	8
	0.30	1.66	0.	0.	0.	0.258	±6.5	8
	0.35	1.63	0.	0.	0.	0.327	±5.5	7
	0.40	3.50	16.32	15.2	0.18	0.158	3.0	6
	0.50	5.56	13.16	16.1	0.18	0.190	1.8	4
	0.60	5.06	9.53	15.7	0.19	0.231	1.6	4
100	0.015	0.49	0.86	3.4	0.63	0.035	10.4	23
	0.03	0.72	1.49	4.7	0.65	0.045	3.9	14
	0.06	1.23	2.18	5.2	0.64	0.063	-0.2	8
	0.12	1.83	2.47	6.2	0.63	0.107	-1.6	7
	0.18	1.95	1.48	6.0	0.58	0.174	-1.1	4
	0.21	1.78	0.52	5.8	0.54	0.210	-0.7	3
	0.24	2.24	0.	4.4	0.50	0.214	0.7	3
	0.27	1.25	0.	5.2	0.50	0.224	1.6	4
10	0.03	0.15	0.65	2.19	0.70	0.077	0.7	6
	0.06	0.42	0.79	1.99	0.67	0.109	-0.2	4
	0.12	0.62	0.84	1.88	0.65	0.149	-0.7	4

TABLE II

$\frac{\omega_{pi}^2}{\omega_{ci}^2}$	$\frac{k_0 a_i}{2\pi}$	$\frac{\text{Im } \omega}{\omega_{ci}}$	$\frac{\text{Re } \omega}{\omega_{ci}}$	$k_y a_i$	$\frac{k_0 x_{peak}}{2\pi}$	$\frac{k_0 \Delta x}{2\pi}$	$\frac{k_{x\ peak}}{k_0}$	p_{max} needed
10000	0.21	12.48	0.	22.0	0.50	0.10	-1	6
	0.24	11.63	0.	24.7	0.50	0.11	3	7
	0.40	3.48	29.19	29.3	0.22	0.15	5	8
	0.50	7.75	23.21	30.0	0.21	0.19	3	5
	0.60	6.75	16.25	29.7	0.22	0.24	2	4
	0.70	4.41	10.72	28.4	0.26	0.36	2	3
	0.85	2.31	0.	26.9	0.	0.27	4	5
	1.00	1.16	0.	10.6	0.	0.42	1	2
1000	0.18	5.43	4.46	17.8	0.59	0.08	-3	9
	0.21	5.63	0.	10.7	0.50	0.12	-1	5
	0.24	5.63	0.	13.1	0.50	0.14	2	4
	0.27	3.18	0.	15.2	0.50	0.18	3	6
	0.30	1.95	0.	0.	0.	0.05	±5	8
	0.35	1.81	0.	0.	0.	0.09	±5	8
	0.40	1.76	2.13	7.5	0.13	0.14	-6	9
	0.50	3.19	11.91	17.6	0.24	0.24	2	4
	0.60	2.50	7.99	16.9	0.25	0.33	1	3
	0.70	1.16	0.	4.2	0.	0.22	0	3
	0.85	1.25	0.	6.6	0.	0.31	1	2
	1.00	0.82	0.31	7.9	0.95	0.42	1	2
100	0.06	1.19	2.17	5.4	0.65	0.06	0	7
	0.12	1.71	2.52	6.6	0.66	0.09	-1	6
	0.18	1.71	1.69	6.4	0.63	0.13	-1	5
	0.21	1.50	1.04	6.2	0.60	0.16	-1	4
	0.24	1.32	0.	4.5	0.50	0.22	0	2
	0.27	0.84	0.	5.3	0.50	0.27	1	2
	0.30	0.36	1.53	3.8	0.40	0.34	1	3
	0.35	0.39	1.68	3.1	0.33	0.38	1	2
	0.40	0.42	0.83	4.6	0.07	0.25	2,-3	5
	0.50	0.38	0.78	2.3	0.22	0.18	-1	3
	0.60	0.44	0.63	3.6	0.16	0.23	-1	3
	0.70	0.54	0.	3.6	0.	0.32	0	1
	0.85	0.42	0.35	4.6	0.95	0.39	1	1
	1.00	0.35	0.45	6.3	0.93	0.38	1	1
10	0.06	0.38	0.75	2.13	0.68	0.15	0	3
	0.12	0.52	0.80	2.17	0.68	0.16	0	3
	0.18	0.47	0.64	2.20	0.68	0.20	0	3
	0.24	0.15	0.41	2.08	0.66	0.30	0	2
	0.30	stable						
	0.50	stable						
3	0.15	stable						

APPENDIX K

Derivation of Eq. (38):

$G_{\pm 1,s}(k_x,\omega)$ is given by Eq. (35) for ions in the straight-line orbit approximation and by Eq. (36) for cold electrons. We note that $G_{+1,s} = G_{-1,s}$ for the ions if $k_y = 0$, and this is also true for the electrons if $k_y = 0$ and $k_0 << k_x$. Then, using Eqs. (23) and (25), we have

$$D(x,k_x,\omega) = -k_x^2 + \sum_s G_{0,s}(k_x,\omega) + 2\cos k_0 x \sum_s G_{+1,s}(k_x,\omega) \tag{K1}$$

The condition $\partial D/\partial x = 0$ from Eq. (16) can be satisfied if and only if $k_0 x = 0$ or π in Eq. (K1). We take $k_0 x = 0$. Then Eq. (16) becomes

$$-k_x^2 + \sum_s [G_{0,s}(k_x,\omega) + 2\, G_{+1,s}(k_x,\omega)] = 0 \tag{K2}$$

$$-2k_x + (\partial/\partial k_x) \sum_s [G_{0,s} + 2G_{+1,s}] = 0 \tag{K3}$$

evaluated at $k_x = k_L$, $\omega = \omega_L$.

For a loss cone ion distribution we have, from Eq. (11), two ion "species", with

$$\begin{aligned} n_{0,1} &= R(R-1)^{-1}\, n_{0,i} \\ n_{0,2} &= -(R-1)^{-1}\, n_{0,i} \\ v_1 &= v_i \\ v_2 &= v_i R^{-1/2} \end{aligned} \tag{K4}$$

For $\omega \ll k_x v_i$ (justified a posteriori), we can take ζ_0, $\zeta_{\pm 1} \ll 1$ in Eq. (35), which becomes

$$G_{0,s} \approx -(\omega_{ps}^2/v_s)\ [1 + i(\pi/2)^{1/2}\ \omega/kv_s]$$

$$G_{\pm 1,s} \approx -(\omega_{ps}^2 \Delta/2v_s^2)\ \exp(-k_0^2 a_s^2/2)$$

$$[1 + i(\pi/2)\ \omega/kv_s] \qquad \text{(K5)}$$

Putting Eq. (K4) into Eq. (K5), and assuming $\Delta_i = 0$,

$$G_{0,i} = -(\omega_{pi}^2/v_i^2)R(R-1)^{-1}(1-R^{1/2})i(\pi/2)^{1/2}\ \omega/k_x v_i$$

$$G_{\pm 1,i} = -(\omega_{pi}^2/v_i^2)R(R-1)^{-1}\ \{[\exp(-k_0^2 a_i^2/2)$$

$$-\exp(-k_0^2 a_i^2/2R)] + i(\pi/2)^{1/2}(\omega/k_x v_i)$$

$$[\exp(-k_0^2 a_i^2/2) - R^{1/2}\exp(-k_0^2 a_i^2/2R)]\} \qquad \text{(K6)}$$

For the electrons, assuming $\omega \ll \omega_{ce}$, $k_0 \ll k_x$, and $k_y = 0$, Eq. (36) gives

$$G_{0,e} = -\omega_{pe}^2 k_x^2/\omega_{ce}^2$$

$$G_{\pm 1,e} = -\omega_{pe}^2 k_x^2 \Delta_e/2\omega_{ce}^2 \qquad \text{(K7)}$$

Using Eqs. (K6) and (K7), Eqs. (K2) and (K3) become

$$-k_L^2[1 + \omega_{pe}^2(1 + \Delta_e)/\omega_{ce}^2] + (\omega_{pi}^2/v_i^2)R(R-1)^{-1}$$

$$\{e_2 - e_1 + i(\pi/2)^{1/2}(\omega_L/k_L v_i)[R^{1/2}(e_2 + 1) - (e_1 + 1)]\} = 0$$

(K8)

$$-2k_L[1 + \omega_{pe}^2(1 + \Delta_e)/\omega_{ce}^2] - (\omega_{pi}^2/v_i^2)R(R-1)^{-1}$$

$$i(\pi/2)^{1/2}(\omega_L/k_L^2 v_i)[R^{1/2}(e_2 + 1) - (e_1 + 1)] = 0 \qquad \text{(K9)}$$

where $e_1 \equiv \exp(-k_0^2 a_i^2/2)$, $e_2 \equiv \exp(-k_0^2 a_i^2/2R)$, and Eqs. (14a) and (K4) can be used to find Δ_e

$$\Delta_e = (Re_1 - e_2)(R - 1)^{-1} \qquad \text{(K10)}$$

Simultaneously solving Eqs. (K8) and (K9) for ω_L and k_L yields Eq. (38).

We now justify the assumption that $\omega << k_x v_i$ by using Eq. (38) to find

$$\omega_L/k_L v_i = 0.532i(e_2 - e_1)\ [R^{1/2}(e_2 + 1) - (e_1 + 1)]^{-1} \qquad \text{(K11)}$$

For $R = 3$ and $k_0 a_i = 2$ (typical parameters for region C in Figure 6), Eq. (K11) yields $\omega_L/k_L v_i = 0.135$. The next term in the expansion of the Z function is[20]

$$Z(\zeta) = i\pi^{1/2} - 2\zeta + \ldots \qquad \text{(K12)}$$

For $k_y = 0$, $\zeta_0 = \zeta_{\pm 1} = \omega/\sqrt{2}\ k_x v_i$ in Eq. (35), so the relative error in $Z(\zeta)$ due to finite $\omega_L/k_L v_i$ is

$$\text{Relative error} = 2\pi^{-1/2}\ \zeta = (2/\pi)^{1/2}\ \omega_L/k_L v_i = 0.108$$

Since the approximation $Z(\zeta) \approx i\pi^{1/2}$ is good to within about 10%, Eq. (38) probably good to within about 10% as well. For $m_i/m_e \leq 3700$, this error is comparable to or less than the error due to nonlocal effects (i.e. due to finite k_0/k_x), so there is no point in using Eq. (K12) to approximate the Z function.

APPENDIX L

We consider two different configurations of plasma, a slab with ion guiding center density

$$n_i(x,y) = n_0 \exp(-x^2/L^2) \tag{L1}$$

and a column with

$$n_i(x,y) = n_0 \exp(-x^2/L^2 - y^2/L^2) \tag{L2}$$

The ion distribution function is

$$f_{0i}(x,y,v_x,v_y,v_z) = g_i(v_\perp,v_z,v_{gc},y_{gc})$$

$$= g_i(v_\perp,v_z,x + v_y/\omega_{ci},\ y - v_x/\omega_{ci}) \tag{L3}$$

To apply the Penrose criterion with $k_y = k_z = 0$, at $x = 0$, we will need

$$F(v_x) = \int dv_y dv_z\ f_{0i}(0,0,v_x,v_y,v_z)$$

$$= \int dv_y\ g_{\perp i}(v_\perp,v_y/\omega_{ci},\ -v_x/\omega_{ci}) \tag{L4}$$

where $g_{\perp i}(v_\perp,x,y) \equiv \int dv_z\ g_i(v_\perp,v_z,x,y)$.

We take $g_{\perp i}$ to be a loss cone distribution of the form given by Eq. (11). Then Eq. (L4) becomes

$$F(v_x) = [2\pi v_i^2(1 - R^{-1})]^{-1} \int dv_y\ n_i(v_y/\omega_{ci},\ -v_x/\omega_{ci})$$

$$[\exp(-v_\perp^2/2v_i^2) - \exp(-Rv_\perp^2/2v_i^2)] \tag{L5}$$

For a slab, $n_i(x,y)$ is given by Eq. (L1), and Eq. (L5) becomes

$$F_{slab}(v_x) = (2\pi)^{-1/2}[v_i(1 - R^{-1})]^{-1} n_0$$

$$[(1 + 2v_i^2/L^2\omega_{ci}^2)^{-1/2} \exp(-v_x^2/2v_i^2)$$

$$-(R + 2v_i^2/L^2\omega_{ci}^2)^{-1/2} \exp(-Rv_x^2/2v_i^2)] \qquad (L6)$$

For a column, $n_i(x,y)$ in Eq. (L5) is given by Eq. (L2), and

$$F_{col}(v_x) = (2\pi)^{-1/2} [v_i(1 - R^{-1})]^{-1} n_0$$

$$[(1 + 2v_i^2/L^2\omega_{ci}^2)^{-1/2} \exp(-v_x^2/2v_i^2 - v_x^2/L^2\omega_{ci}^2)$$

$$-(R + 2v_i^2/L^2\omega_{ci}^2)^{-1/2} \exp(-Rv_x^2/2v_i^2 - v_x^2/L^2\omega_{ci}^2)] \qquad (L7)$$

According to the Penrose criterion, an instability occurs when

$$\int_{-\infty}^{\infty} \frac{F(0) - F(v_x)}{v_x^2} dv_x < 0 \qquad (L8)$$

(The electrons are ignored since they are unimportant for this instability.)

Both Eqs. (L6) and (L7) are of the form

$$F(v_x) = A \exp(-av_x^2) - B \exp(-bv_x^2) \qquad (L9)$$

Putting Eq. (L9) into inequality (L8) yields

$$A^2a < B^2b \qquad (L10)$$

for instability. Thus the column is marginally stable, and the slab is unstable, to electrostatic perturbations with $\underset{\sim}{k}$ in the x-direction, at $\underset{\sim}{x} = 0$, according to the Penrose criterion.

This analysis is cruder than the analysis in Appendix K since it does not find the normal modes but assumes there is a normal mode with

$\phi(\underset{\sim}{x}) \approx \exp(-i\underset{\sim}{k}\cdot\underset{\sim}{x})$ at $\underset{\sim}{x} = 0$. However, it makes the physical mechanism of the instability more clear; it is due to a relative deepening of the loss cone in $F(v_x)$ as a result of $d^2n_i/dx^2 < 0$, and $d^2n_i/dx^2 < d^2n_i/dy^2$.

References

1. N. A. Krall and A. W. Trivelpiece, Principles of Plasma Physics, McGraw-Hill, New York, 1973, p. 424.

2. A. B. Mikhailovskii, Theory of Plasma Instabilities, Consultants Bureau, New York, 1974, Vol. 2.

3. F. H. Coensgen, W. F. Cummins, V. A. Finlayson, W. C. Nexsen, T. C. Simonen, in Plasma Physics and Controlled Nuclear Fusion Research, International Atomic Energy Agency, Vienna, 1971, Vol. II, p. 721.

4. C. K. Birdsall, D. Fuss, N. Lindgren, Bull, Am. Phys. Soc. 14, 1024 (1969).

5. C. K. Birdsall and D. Fuss, Bull. Am. Phys. Soc. 15, 1437 (1970).

6. N. A. Krall and A. W. Trivelpiece, op. cit., p. 402.

7. Ibid, p. 404

8. Ibid, p. 429.

9. W. M. Tang, L. D. Pearlstein, H. L. Berk, Phys. Fluids 15, 1153 (1972).

10. N. A. Krall, "Drift Waves," in A. Simon and W. Thompson (eds.), Advances in Plasma Physics, Wiley, New York, 1968, Vol. 1, p. 153.

11. L. D. Pearlstein, Phys. Fluids 8, 1743 (1965).

12. F. C. Hoh, Phys. Fluids 8, 1741 (1965).

References (cont.)

13. H. L. Berk. T. K. Fowler, L. D. Pearlstein, R. F. Post, J. D. Callen, W. C. Horton, M. N. Rosenbluth, in Plasma Physics and Controlled Nuclear Fusion Research, International Atomic Energy Agency, Vienna, 1969, Vol. II, p.151.

14. R. F. Post and M. N. Rosenbluth, Phys. Fluids 9, 730 (1966).

15. A. B. Mikhailovskii and A. V. Timofeev, Zh. Eksp. Teor. Fiz. 44, 919 (1963) [Sov. Phys.-JETP 17, 626 (1963)].

16. R. D. Rodman, Assoc. Computing Machinery Communications 6, 442 (1963).

17. V. W. Whitley, Assoc. Computing Machinery Communications 11, 12 (1968).

18. J. F. Traub, Iterative Methods for the Solution of Equations, Prentice-Hall, Englewood Cliffs, N. J., 1964, p. 210-214.

19. M. Abramowitz and I. A. Stegun (eds.), Handbook of Mathematical Functions, National Bureau of Standards, 1964, formula 9.7.7 with $z \gg 1$.

20. B. Fried and S. Conte, The Plasma Dispersion Function, Academic Press, New York, 1961.

21. J. F. Traub, op. cit., p. 220.

22. O. Penrose, Phys. Fluids 3, 258 (1960).

23. I. S. Gradshteyn and I.M. Ryzhik, Table of Integrals, Series and Products, Academic Press, New York, 1965.

24. G. Floquet, Annales Scientifiques Ecole Normale Superieure 12, 47 (1883).