UNIVERSITY OF ALBERTA

AN INVESTIGATION OF RADIO-FREQUENCY HEATING PATTERNS ALONG A PARALLEL-PLATE WAVEGUIDE LOADED WITH LAYERED OILSAND MEDIA

BY

RICHARD MASLOWSKI

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

SPRING 1991

ISBN 0-315-66605-6

April 5, 1991

From : R.G. McPherson
P. O. Box 1624
Morinville, Alberta

To: Richard Maslowski
Dept. of Electrical Engineering
University of Alberta
Edmonton, Alberta
T6G 2M7

Please accept this letter as my permission to use the figure 2-1 of my thesis "High Frequency Electric Heating of Oilsands" in your thesis.

Yours truly

R. G. McPherson

Dr. R. G. McPherson

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: RICHARD MASLOWSKI

TITLE OF THESIS: AN INVESTIGATION OF RADIO-FREQUENCY HEATING PATTERNS ALONG A PARALLEL-PLATE WAVEGUIDE LOADED WITH LAYERED OILSAND MEDIA

DEGREE: MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED: 1991

Richard Maslowski
(student's signature)

11719 - 123 St.
(Permanent Address)
Edmonton, Alberta
Canada T5M 0G8

Date: April 15, 1991

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

THE UNDERSIGNED CERTIFY THAT THEY HAVE READ, AND RECOMMEND TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH FOR ACCEPTANCE, A THESIS ENTITLED

AN INVESTIGATION OF RADIO-FREQUENCY HEATING PATTERNS ALONG A PARALLEL-PLATE WAVEGUIDE LOADED WITH LAYERED OILSAND MEDIA

SUBMITTED BY RICHARD MASLOWSKI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

D. G. Hughes

C. G. Englefield

C. G. Englefield

F. S. Chute

F. E. Vermeulen

Date: March 14, 1991

Dedication

This work is dedicated to my mother and father, Zofia and Stanislaw.

Abstract

In order to enhance bitumen recovery from deeply bedded oilsand formations, electrically preheating the oilsand with radio-frequency guided electromagnetic energy to reduce its viscosity has been suggested. Prior investigations done in the AEL (Applied Electromagnetics Laboratory) in the Electrical Engineering Department at the University of Alberta have identified horizontal electrode placement as a preferred geometry. This geometry can be closely approximated by a parallel-plate waveguide. For waveguides longer than a fraction of the guided wavelength, standing wave effects will occur which lead to non-uniform ohmic heating patterns that are destructive to efficient bitumen recovery. If a region around each electrode (modelled as a layer in the waveguide) is depleted of moisture, then two important effects occur. Firstly, the typical TEM standing wave pattern associated with a homogeneously loaded waveguide disappears and an interleaving of different electric field components creates a more uniform electric field (and heating) distribution within the moisture-saturated region of the waveguide. Secondly, the depleted region acts to increase the guided wavelength and decrease the wave attenuation, resulting in deeper energy penetration, thereby allowing larger volumes of oilsand to be effectively heated. By proper termination of the parallel-plate waveguide and close examination of its analytic field solution, it has been found that the efficiency of the energy placement in the oilsand formations and thus, overall bitumen recovery, may be improved.

Research conducted at the University of Alberta has also shown that, under certain conditions, the familiar TEM transmission line equivalent circuit can in fact be used to describe the non-TEM wave behaviour of the parallel-plate waveguide under study. An investigation of this initial

discovery and its application to the aforementioned waveguide heating problem is also given.

Acknowledgement

I would sincerely like to thank my supervisors, F. E. Vermeulen and F. S. Chute, for their help and support during this project.

I would also like to thank the Alberta Oil Sands Technology and Research Authority for their generous support.

Table of Contents

List of Tables

List of Figures

List of Symbols

σ	electrical conductivity [S/m]
ε	electric permittivity [F/m]
μ	magnetic permeability [H/m]
$\mathbf{A}$	magnetic vector potential [A]
$\mathbf{F}$	electric vector potential [V]
k_z	complex axial propagation constant
α	axial attenuation constant [Np/m]
β	axial phase constant [rad/m]
k_{x1}	complex saturated region transverse propagation constant
k_{x2}	complex depleted region transverse propagation constant
$\mathbf{E}$	electric field intensity [V/m]
$\mathbf{H}$	magnetic field intensity [A/m]
Φ	electric scalar potential [V]
Ψ	scalar magnetic potential [A]
Y	admittance [S]
Z	impedance [Ω]
γ	complex axial propagation constant (γ=-jk_z)
a	plate separation[m]
d	saturated layer thickness[m]
f	frequency of operation [Hz]
ω	frequency of operation [rad/s]
R	resistance [Ω]
G	conductance [S]
C	capacitance [F]
L	inductance [H]
j	$\sqrt{-1}$

1. Background and Problem Identification

1.1 General Background

Of the 2100 billion barrels of oil contained in oilsand formations in the world today, 980 billion barrels are resident in Alberta.[1] The bulk (approximately 78%) of the comparatively moist Alberta oilsand lies below 150m of overburden and cannot be readily surface mined. It is generally sandwiched between layers of watersands and shale. At the average in-situ temperature of 10-15°C the oil is very viscous. The oilsand also has low thermal conductivity and permeability and, therefore, cannot readily transport heat. This renders conventional steam and fluid injection recovery methods relatively ineffective. Since the viscosity of the oil decreases rapidly with an increase in temperature (a rise in temperature from 15°C to 100°C reduces viscosity from 10^6 centipoise to several hundred centipoise), preheating the oilsand prior to steam or fluid injection recovery has been proposed. Converting electrical energy to heat the formation is an obvious possibility. Two approaches to electrical preheat have been considered: low frequency (60 Hz and below) and radio-frequency (generally greater than a few hundred kilohertz) electromagnetic methods.

1.2 Electromagnetic Heating

1.2.1 Heating Mechanisms

Given a volume of material subjected to a time-harmonic electromagnetic field, two kinds of current are generated. Current that is in time-phase with the applied electric field is classified as conduction current, while current that is in time-quadrature with the applied electric field is known as displacement current. If the conduction current is greater than the displacement current, the material is termed a conductor. If the displacement current dominates, the material is termed a dielectric. Because electrical heating of a material is solely dependent on conduction current, the present interest in heating oilsand requires study of the conduction current within the oilsand.

For any given material, the conduction current density is related to the applied electric field by

$$\mathbf{J}=\sigma\mathbf{E}$$

where σ is the conductivity of the material, **J** is the current density and **E** is the electric field strength. In general, two mechanisms contribute to the conductivity of a substance. If the substance contains free charge carriers, as in the form of ions in water, then the applied electric field will accelerate the ions whose collisions with other molecules will increase the random thermal motion, and thus increase the temperature of the material. If the material contains polar molecules, the application of an electric field will tend to turn and align the molecules with the field against rotational friction. Turning of the molecules dissipates energy as heat and, therefore, also results in an increase in temperature of the material.

The water-wet Athabasca oilsands have a conductivity which is dependent on both ionic and polarization contributing factors. At lower frequencies (<1MHz), the conductivity of the oilsands is mainly due to ionic conduction in the connate water. However, at frequencies above about 10 MHz, it has been found that losses due to polarization of the ions in the connate water are dominant.

Depending on the contribution to the conductivity of a material by each of these mechanisms, an optimal frequency of operation for heating may generally be chosen. Materials exhibiting low ionic conductivity but containing polar molecules, can generally be heated effectively with high frequency electromagnetic energy. On the other hand, materials with high ionic conductivity would generally be heated most effectively with low frequency electromagnetic energy.

1.2.2 Low Frequency Methods

Initially, 60 Hz methods were considered for oilsand preheating, mainly due to the ready availability of large amounts of power at this frequency. These methods usually involve placing electrodes into or near the formation to be heated. This method however, may exhibit highly non-uniform current distributions, mainly near the electrodes, where the high current densities tend to drive off the connate water in the oilsand. Since the connate water present is responsible for the current passage in the oilsand, the depleted regions near the electrodes act to effectively decouple the formation and prevent further heating, often a very short time after power is first applied. Several methods have been devised to resolve this problem, such as pressurizing the electrode-oilsand interface to increase the water vapourization point or pumping fluid down to cool the electrode and replace the lost formation moisture.[2]

1.2.3 High Frequency Methods

Pioneered by the Illinois Institute of Technology Research Institute

(IITRI), radio-frequency methods have gained considerable attention recently. The IITRI group used electrode configurations that were excited at frequencies greater than 1 MHz to heat in-situ, the low moisture, low electrical conductivity oilsand formations found in Utah.[3]

The Utah oilsands are dry and exhibit very low electrical conductivity at frequencies below a few hundred kilohertz. This translates to low wave attenuation and deep, uniform energy deposition in the formation at low frequencies. However, the power dissipated within the formation per-unit-volume is given by

$$\sigma|E|^2$$

where σ is the formation conductivity and E is the rms electric field strength. Therefore, at low frequencies, extremely large fields must be excited in the formation in order to heat the formation in a reasonable length of time. The electrical conductivity of these low-moisture oilsands, however, increases with frequency.[4] By operating at relatively high frequencies, the electric field strength can be reduced without diminishing the power dissipated. The IITRI scheme operated at frequencies sufficiently high that the length of the guided wave excitor was comparable to the so-called electrical skin-depth[5] given by

$$\delta=\sqrt{\frac{1}{\pi f \mu \sigma}}$$

The Alberta oilsands are of high moisture content and as a result the electrical conductivity is relatively large and the skin depth is often less than a few metres at high frequencies. Overcoming this limited depth of penetration is a major hurdle to the successful application of radio-frequency preheating to the Alberta oilsands.

The Electrical Engineering Department at the University of Alberta has identified three possible schemes of heating high loss oilsand using horizontal transmission lines in the formation to be heated (see Figure 1). These schemes effectively increase the depth of penetration of the electromagnetic energy, thus allowing large channels of oilsand to be heated.

The first method, termed partial evaporation boring, uses an appropriate frequency and power level to dry out a small cylindrical shell around each transmission line conductor by applying radio-frequency power to the line. Each transmission line consists of two conductors, one placed vertically above the other in the formation as shown in Figure 1. The drying initially begins near the power feedpoint and progresses down the line. The result is an effective gradual decoupling of the two electrodes, thus permitting deeper and deeper penetration of wave energy into the formation. This scheme is sometimes referred to as electromagnetic flood heating or simply as EMF.[6]

The second approach, referred to as production controlled heating, operates in a manner similar to partial evaporation boring. Here however, the bottom electrodes are perforated hollow tubes designed to collect and produce bitumen that once having been heated, drains by gravity towards the lower wells. A reduced power level is used to heat the oilsand to temperatures below the steam point so as to promote a depletion region around the upper electrodes primarily due to bitumen drainage and not due to moisture evaporation. The draining bitumen may then be recovered from the lower wells by conventional techniques. An advantage of this method over the other two is that the oil is collected as the formation is being heated. No other recovery methods are necessary subsequent to the electrical heating process.

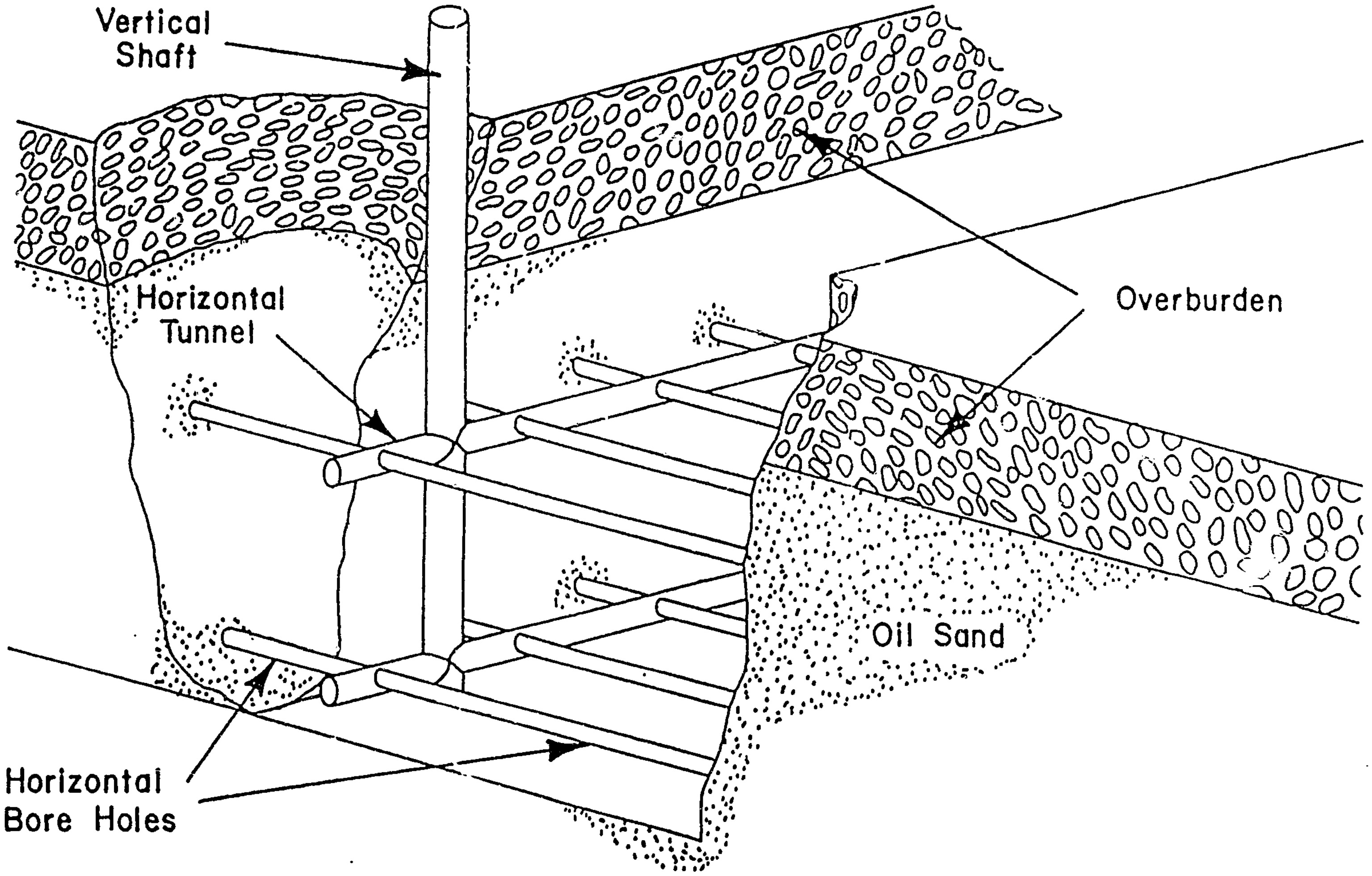

Figure 1. *A sketch showing a means of placing electrodes into the oilsand formation. The electrodes are placed in horizontal bore holes drilled into the formation payzone from mined cross shafts.Taken from McPherson, R.G., "High Frequency Electric Heating of the Athabasca Oilsands", Ph.D. Thesis, University of Alberta, 1985.*

The final method is again similar to the above ones, but instead of using electromagnetic energy to create a depletion region, one or both of the electrodes of each electrode pair are precoated with a dielectric material. This partially decouples the transmission line from the formation and allows immediate deep uniform energy penetration into the formation.

All of the above methods, when viewed in cross-section, result in a region surrounding each electrode that is of lower electrical conductivity than the bulk of the formation, as suggested by Figure 2. For arrays of transmission lines situated side by side, the overall configuration may be approximated by a parallel-plate waveguide with a two-layer medium between the plates, the upper layer being the moisture-free depleted region with low conductivity, and the lower layer being the high conductivity water-wet oilsand, as illustrated in Figure 3. In actuality, depending on which of the three previously discussed heating schemes is used, the region between the plates may be divided into three layers: a dried-out layer near each electrode and the moisture-saturated region in-between. A solution for this configuration can be obtained from the solution to the two-layer model by noting the line of symmetry which runs along the waveguide at the midpoint between the plates. The problem may be divided into two identical parts which may be solved by the two-layer model. The two-layer model of Figure 3 may be used as a model to study RF heating to determine if it is a possible and viable means of economically preparing deeply bedded oilsand payzones for further oil recovery. The great depth of penetration effected by the introduction of a depleted layer into the parallel-plate geometry allows much larger volumes of oilsand to be heated much more uniformly than would otherwise be possible.

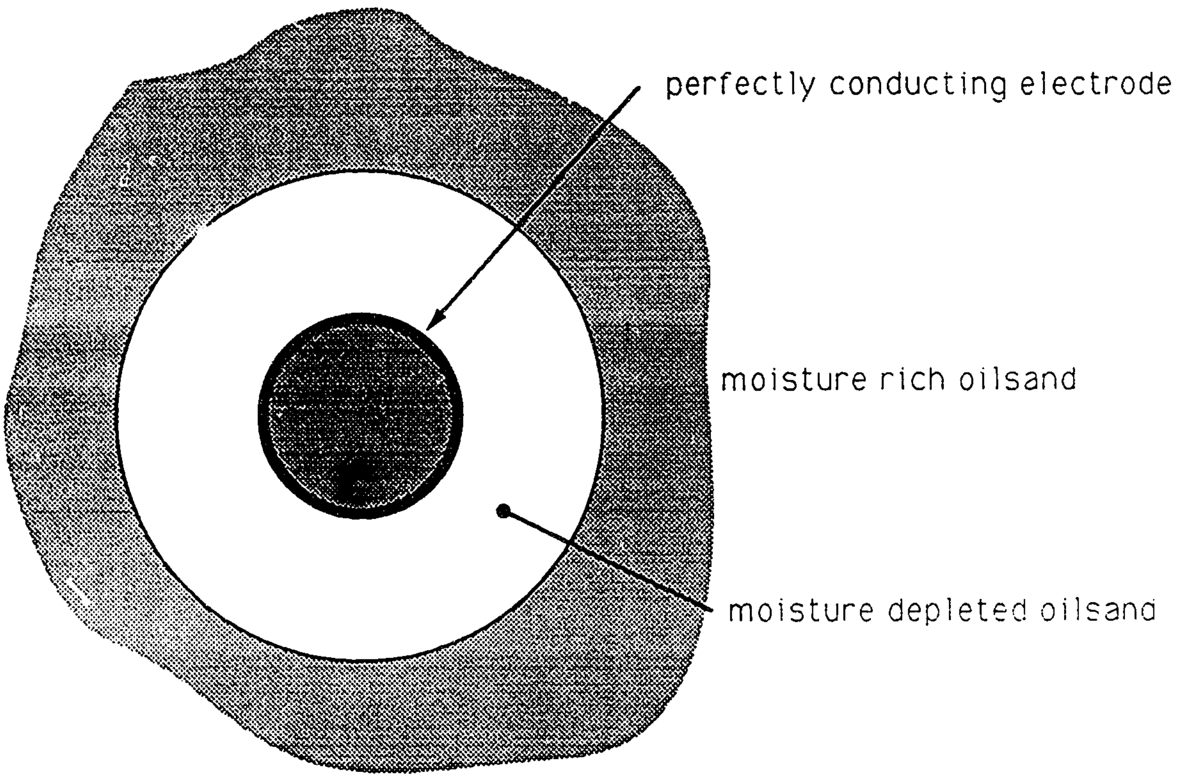

Figure 2. *Cross-sectional view of a single electrode surrounded by an insular region of depleted oilsand.*

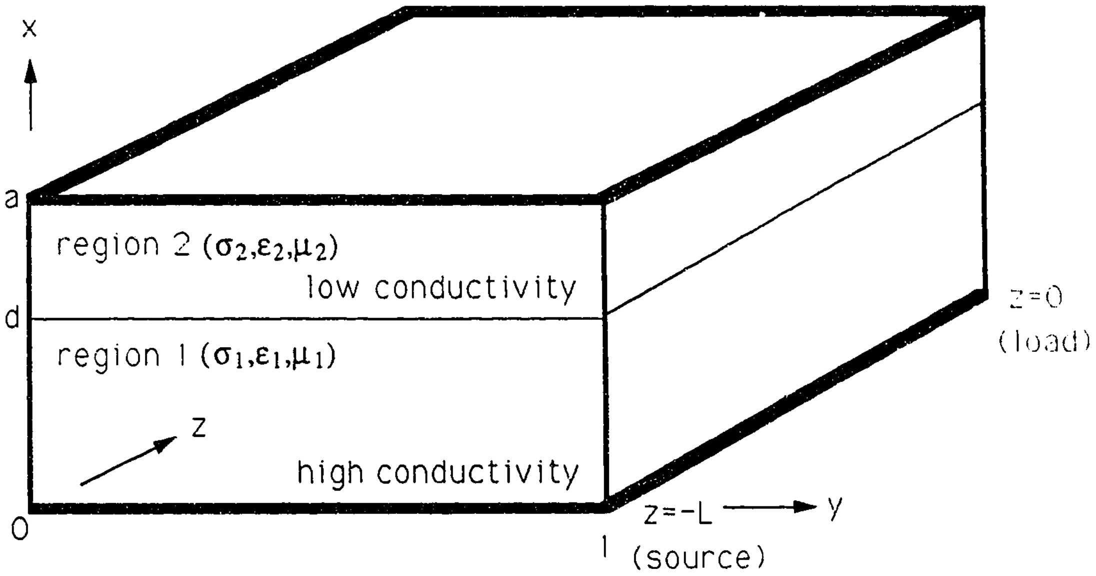
x
a
region 2 ($\sigma_2,\varepsilon_2,\mu_2$)
low conductivity
d
region 1 ($\sigma_1,\varepsilon_1,\mu_1$)
z
high conductivity
0
1
z=-L
(source)
y
z=0
(load)

Figure 3. *The parallel-plate waveguide system configuration.*

1.2.4 Heating Uniformity of Standing vs. Travelling Waves

Research done in the Applied Electromagnetics Laboratory has investigated the use of forward travelling waves as the energy source for radio-frequency heating of oilsand formations. For horizontal electrode pairs whose lengths are comparable to or less than the 'effective' skin depth of the guided wave, highly uniform axial power deposition can be achieved by utilizing a matched load to terminate the electrode pair and thereby, eliminate or minimize wave reflections. Such an approach is not, however, necessarily cost effective. For example, consider the case of a matched load connected to a transmission line of length comparable to the skin depth of the guided electromagnetic wave. As the heating proceeds, system properties, such as the saturated oilsand conductivity change with temperature and affect other parameters such as the transmission line characteristic impedance. Consequently, the matched load would require retuning. This would likely require physical access to the load through an additional mine shaft, a costly further expense.

On the other hand, open or short-circuit waveguide terminations, which would result in a standing wave field distribution along the electrodes, could be implemented for a relatively low or no additional cost. At first appearance this would seem to be self-defeating, for a typical standing wave pattern would normally create undesirable non-uniform heating. However, the presence of an axial field due to the layered medium, creates an additional axial electric field standing wave pattern which interleaves with the transverse electric field standing wave pattern and is shifted one-quarter wavelength spatially with respect to the latter, as

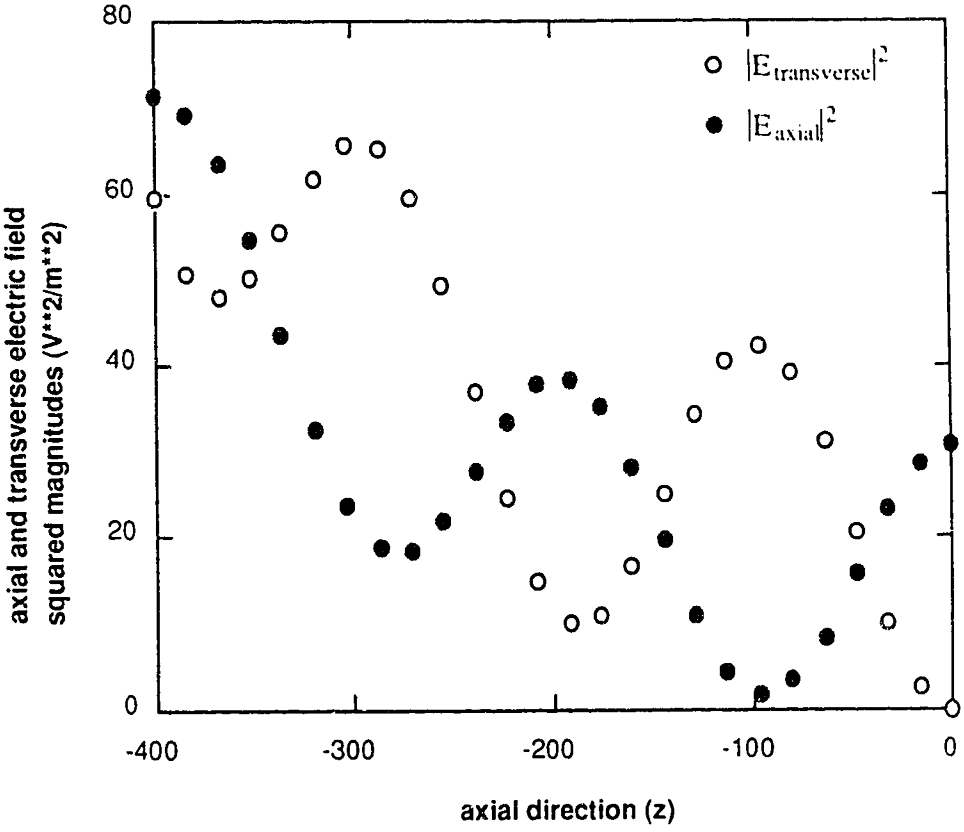

Figure 4. *An example of the axial interleaving of the axial and transverse electric field components in the saturated medium of the parallel-plate waveguide*

illustrated in Figure 4. The field distributions shown were computed at $x=13.2$m for a waveguide as illustrated in Figure 3 with properties: $a=15$m, $d=13.5$m, $\sigma_1=10^{-3}$ S/m, $\mu_1=\mu_0$, $\varepsilon_1=11\cdot\varepsilon_0$, $\sigma_2=10^{-6}$ S/m, $\mu_2=\mu_0$, $\varepsilon_2=3\cdot\varepsilon_0$ This interleaving tends to even out the overall heating pattern along the length of the waveguide.

The potential for uniform formation heating and substantial cost saving of the standing wave system configuration over the travelling wave configuration is a major motivation for the work described in this thesis.

1.3 Transmission Line Equivalent Circuit

A preliminary analytical investigation of the fields of the parallel-plate waveguide loaded with a layered medium was conducted by R. G. McPherson.[7] A notable result was that a lumped element TEM (transverse electromagnetic, no axial fields) equivalent circuit could be used to calculate the propagation constant even for waveguide configurations with large axial field components, provided the plate separation was not too large. A very desirable feature of the transmission line solution was a closed form expression for the axial propagation constant in terms of the basic physical system parameters. This permitted calculation of the dominant mode propagation constant for a given waveguide configuration quickly and effortlessly. Further, knowledge of the propagation constant allowed easy calculation of the axial power deposition along the guide for a forward travelling wave.

1.4 Research Objectives

The research presented involves detailed examination of the modified standing wave ohmic heating pattern in the two layer medium between the parallel-plate transmission line. Short and open-circuit waveguide terminations will be used to create the standing wave effects. In addition, a more rigorous investigation will be made of the TEM distributed impedance circuit than that conducted by McPherson.[7] The ability of the TEM distributed impedance circuit to predict the wave propagation constant and thus, the axial power deposition within the waveguide under certain conditions, will also be studied. Also, it will be closely compared to results obtained from the application of a small argument condition to the exact solution of the boundary value problem posed by the 2-layer medium parallel-plate waveguide configuration.

2. Discussion of Field Solutions

2.1 General

As mentioned earlier, the desired formation electrode configuration can be readily modelled by a parallel-plate transmission line loaded with a 2-layered medium, the upper layer representing the moisture-depleted oilsand and the lower layer representing the moisture-saturated oilsand. To obtain accurate results, it is essential to use system parameters representative of actual in-situ heating configurations. The inherent horizontal wellbore geometry, as shown in Figure 1, could be accurately modelled by a parallel-plate waveguide of infinite width, or practically speaking, a very large width in comparison with the plate separation. With all electrode pairs excited in phase, no field variation would occur across the width of the waveguide. Consequently, no y-dependence would appear in the field solution. To ensure a reasonable amount of axial field variation, the waveguide length from the source to the load was generally chosen to be one to two guided wavelengths, and thus ranged from as low as 50m to over 1000m. Present drilling technology allows horizontal wells of up to 1000m length to be drilled. Table 1 gives the relevant geometrical dimensions of the waveguide configuration as they are illustrated in Figure 3.

The electrical properties of the media in between the parallel-plates were also carefully chosen to approximate the actual properties of oilsand. Both layers were accurately modelled as exhibiting both conductive and dielectric behaviour. The actual electrical properties were obtained from data collected at the Applied Electromagnetics Laboratory at the University

SYSTEM PARAMETER	VALUE
Saturated Region Conductivity σ_1	10^{-2}-10^{-4} S/m
Depleted Region Conductivity σ_2	10^{-6} S/m
Saturated Region Dielectric Constant ε_1	$11\cdot\varepsilon_o$
Depleted Region Dielectric Constant ε_2	$3\cdot\varepsilon_o$
Magnetic Permeability μ_1 & μ_2	μ_o
Plate Separation 'a'	10-80m
Saturated Region Thickness 'd'	1.5-14.995m
Frequency	50kHz - 3MHz
Waveguide Length	λ - 2λ

Table 1. *Summary of the waveguide system properties.*

of Alberta and were values typical of Athabasca oilsand. All the system parameters used in this investigation are summarized in Table 1.

To create a data base of heating behaviour of oilsand, heating cross-sections were calculated for different combinations of plate separations, region thicknesses, electrical conductivities, waveguide lengths and frequencies of operation. The remaining system parameters, such as the magnetic permeability and the dielectric constants, were held fixed.

2.2 The Equivalent Circuit Approach

The main purpose of this research was to analyze the resistive heating behaviour in the parallel-plate waveguide to show that large volumes of oilsand could be fairly uniformly heated with radio-frequency power. Before the resistive heating could be studied, knowledge of the electromagnetic fields present in the waveguide was necessary. In order to gain some initial insight into the problem, analysis began with the development of an approximate solution. Following McPherson[8], the simplest model to consider was the familiar transmission line equivalent circuit model which is based on a TEM field pattern and is shown in Figure 5. The actual field pattern in the waveguide is not of the TEM form but, subject to certain system constraints (to be discussed later), the simple equivalent circuit transmission line model has been found to accurately predict the wave propagation constant, k_z.

2.2.1 Development of the Equivalent Circuit

The following is the derivation of the transmission line model. Figure 5 shows the inhomogeneously loaded parallel-plate waveguide and

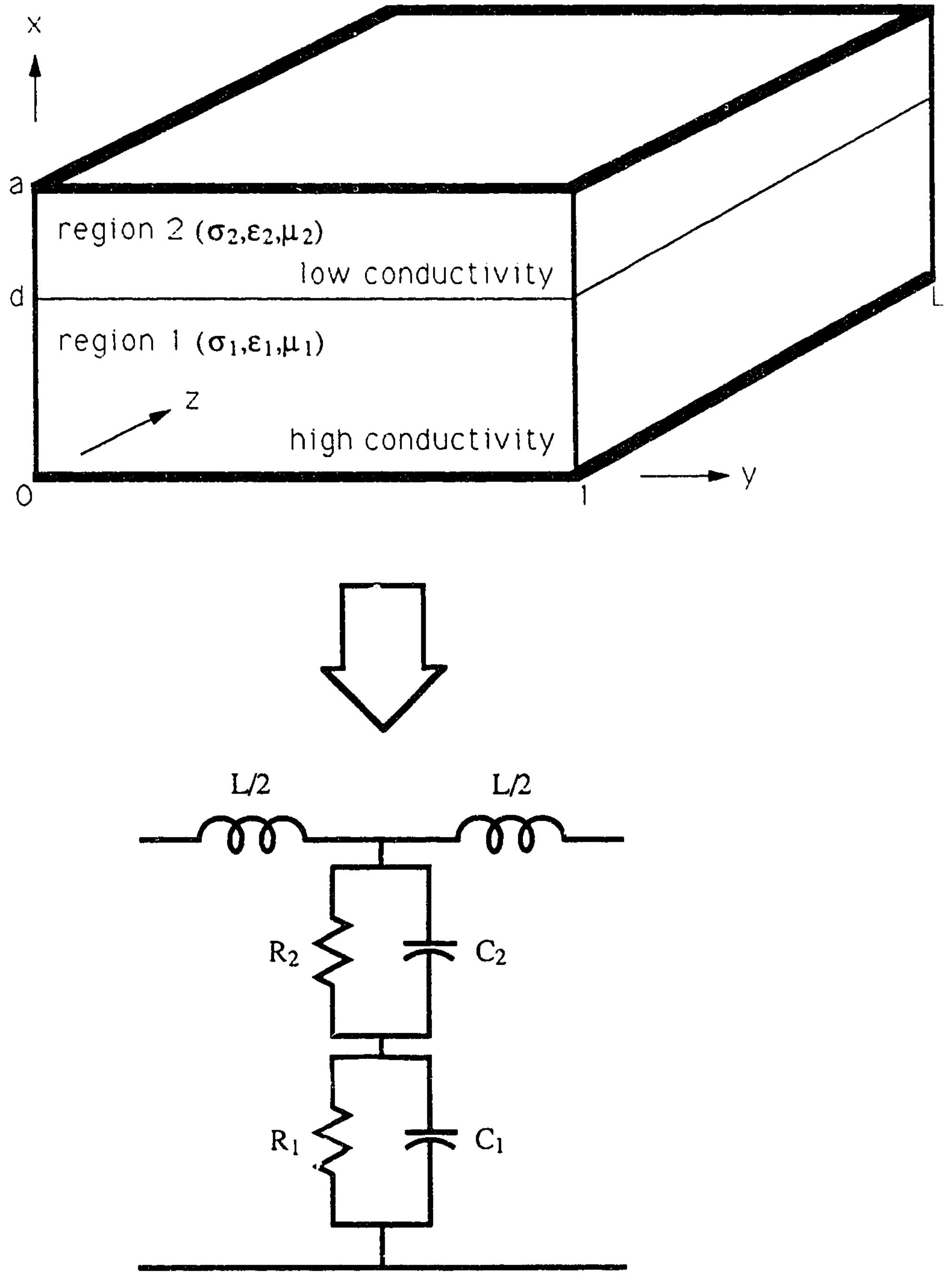
x
a
d
0
region 2 ($\sigma_2,\varepsilon_2,\mu_2$)
low conductivity
region 1 ($\sigma_1,\varepsilon_1,\mu_1$)
high conductivity
z
1
L
y
L/2
L/2
R_2
C_2
R_1
C_1

Figure 5. *The TEM transmission line equivalent circuit for the parallel-plate waveguide configuration shown.*

the corresponding equivalent circuit. This circuit is based on modelling each medium with a parallel resistor(R)/capacitor(C) component combination and modelling the line inductance as two series inductances(L/2).

The shunt impedance per-unit-length of waveguide is given by

$$Z_s=\frac{d}{(\sigma_1+j\omega\varepsilon_1)b}+\frac{(a-d)}{(\sigma_2+j\omega\varepsilon_2)b}$$

(where b is the plate width and is equal to one metre) from which the shunt admittance can be found using

$$Y_s=\frac{1}{Z_s}$$

The series reactance per-unit-length is

$$j\omega L=j\omega\mu(\frac{a}{b})$$

From transmission line theory, for perfectly conducting plates, the propagation constant $\gamma=\alpha+j\beta$, is given by

$$\gamma^2=j\omega LY_s=\frac{j\omega L}{Z_s}=-k_z^2$$

Substitution of the expressions for Z_s and $j\omega L$ into the above equations results in the final expression for k_z

$$k_z{}^2=-\gamma^2=-(\alpha+j\beta)^2=\frac{j\omega\mu(\sigma_1+j\omega\varepsilon_1)(\frac{a}{d})}{1+\frac{(\sigma_1+j\omega\varepsilon_1)(a-d)}{(\sigma_2+j\omega\varepsilon_2)d}}$$

This is the final expression for the propagation constant k_z as a function of physical system parameters. With its calculation, important quantities such as the wave attenuation α and the phase constant β are known.

2.3 The Analytic Field Solution

2.3.1 General

The preceding result is a first order solution and is meant to approximate the axial field behaviour in the waveguide. However, to thoroughly study the exact field behaviour, derivation of the analytic solution is required. Then, comparisons could be made with the first order transmission line solution. As a first step, the forward travelling wave solution is sought. This corresponded to the case of a matched load termination of the waveguide. Then, the standing wave field solutions corresponding to open and short-circuited waveguide terminations, as mentioned in section 1.2.4, are obtained. These were viewed as the superposition of forward and reverse travelling waves on the transmission line combined so as to obey the appropriate waveguide load boundary conditions. The solution procedure follows Harrington's approach[9] of solving the vector Helmholtz equation for the vector potentials from which the electric and magnetic field expressions are finally determined.

2.3.2 Derivation of the Vector Helmholtz Equation

Maxwell's equations may be stated in phasor form as

$$\nabla\times\mathbf{H}=(\sigma+j\omega\varepsilon)\mathbf{E} \qquad [1a]$$

$$\nabla\times\mathbf{E}=-j\omega\mu\mathbf{H} \qquad [1b]$$

$$\nabla\cdot\mathbf{E}=0 \qquad [1c]$$

$$\nabla\cdot\mathbf{H}=0 \qquad [1d]$$

Any vector with zero divergence is the curl of some other vector.

This leads to the definition of a magnetic vector potential A and an electric vector potential F, which can be related to the fields by the relations

$$\nabla\times\mathbf{A}=\mathbf{H} \qquad [2a]$$

$$\nabla\times\mathbf{F}=-\mathbf{E} \qquad [2b]$$

From now on only the magnetic vector potential A will be used in the derivations. Since A and F are duals, derivation for one will automatically apply to the other. (Note: The negative sign introduced in [2b] is for later convenience.)

Substituting $\mathbf{H}=\nabla\times\mathbf{A}$ into the second of Maxwell's equations[1b] gives

$$-\nabla\times\mathbf{E}=j\omega\mu(\nabla\times\mathbf{A}) \qquad [3a]$$

or

$$\nabla\times(\mathbf{E}+j\omega\mu\mathbf{A})=0 \qquad [3b]$$

By a vector identity, any vector whose curl is zero is the gradient of some scalar, so

$$\mathbf{E}+j\omega\mu\mathbf{A}=-\nabla\Phi \qquad [4]$$

where Φ is defined as the electric scalar potential.

Substituting equation [4] and [2a] into the first of Maxwell's equations [1a] yields

$$\nabla\times\nabla\times\mathbf{A}=-(\sigma+j\omega\varepsilon)(j\omega\mu\mathbf{A}+\nabla\Phi) \qquad [5]$$

Defi.ıing $-j\omega\mu(\sigma+j\omega\varepsilon)=k^2$ and using the identity

$$\nabla\times\nabla\times\mathbf{A}=\nabla(\nabla\cdot\mathbf{A})-\nabla^2\mathbf{A} \qquad [6]$$

in equation [5] results in

$$\nabla(\nabla\cdot\mathbf{A})-\nabla^2\mathbf{A}-k^2\mathbf{A}=-(\sigma+j\omega\varepsilon)\nabla\Phi \qquad [7]$$

$\nabla\cdot\mathbf{A}$ has not yet been specified. To simplify equation [7] use

$$\nabla\cdot\mathbf{A}=-(\sigma+j\omega\varepsilon)\Phi \qquad [8]$$

This leaves the equation

$$\nabla^2\mathbf{A}+k^2\mathbf{A}=0 \qquad [9]$$

This is the Vector Helmholtz equation for the magnetic vector potential **A**. Because **A** and **F** are duals, an identical equation applies to the electric vector potentia! **F**. In general, any field pattern can be expressed in terms of a superposition of fields derived from **A** and **F**.

2.3.3 Derivation of the Analytic Field Expressions

An equation has now been derivea om which the magnetic electric vector potential (**A** or **F**) for an electromagnetic problem can be obtained. It remains to find a general relationship between these potentials and the electric and magnetic field intensities **E** and **H** for a z-travelling wave in the waveguide configuration shown in Figure 3.

In a problem where the region is bounded, one can represent the field in terms of **A** or **F** or both, regardless of the actual source.

From the first two Maxwell equations, [1a] and [1b], the fields are given by

$$\mathbf{E}=(\nabla\times\mathbf{H})/(\sigma+j\omega\varepsilon)$$

and

$$\mathbf{H}=-(\nabla\times\mathbf{E})/j\omega\mu$$

where substitution of $\mathbf{H}=\nabla\times\mathbf{A}$ and $\mathbf{E}=-\nabla\times\mathbf{F}$, results in the following equations

$$\mathbf{E}=(\nabla\times\nabla\times\mathbf{A})/(\sigma+j\omega\varepsilon)$$

and

$$\mathbf{H}=(\nabla\times\nabla\times\mathbf{F})/j\omega\mu$$

Adding the contributions to **E** and **H** from both **A** and **F**, gives the fields due to both the electric and magnetic vector potentials

$$\mathbf{E}=-\nabla\times\mathbf{F} + (\nabla\times\nabla\times\mathbf{A})/(\sigma+j\omega\varepsilon) \quad [10]$$

$$\mathbf{H}=\nabla\times\mathbf{A} + (\nabla\times\nabla\times\mathbf{F})/j\omega\mu \quad [11]$$

for a bounded source-free region.

If $\mathbf{F}=0$ and $A=\Psi\cdot\bar{a}$ are chosen as the potential functions, where Ψ is a scalar and $\bar{a}$ is a unit vector in some given direction, then evaluation of the above expressions yields fields which are TM (transverse magnetic) to $\bar{a}$. Similarly, if $\mathbf{A}=0$ and $F=\Psi\cdot\bar{a}$ are chosen, the resulting fields are TE (transverse electric) to $\bar{a}$.

In order to minimize the possibility of searching for non-existent solutions, some common sense may be applied. Consider an inhomogeneously loaded parallel-plate waveguide configuration as shown in Figure 3. Current flows away from the source in the top plate, passes through the formation and returns through the bottom plate. The current through the infinitely wide plates creates a y-directed magnetic field between the plates. Therefore, no solutions TM to y are expected. For the same reasons, solutions TE to x can also be ruled out. Finally, the difference in the electrical properties of the two media necessitates axial currents to satisfy the interface boundary conditions, thus discounting field solutions which are TE to z.

Three remaining possibilities exist. TE to y, TM to x and TM to z. A vector potential corresponding to any one of these may be chosen to obtain the correct field expressions. In fact, it has been found that the actual fields obey all three of the above stipulations. However, finding solutions TM to x proves to be the easiest route. This requires choosing $\mathbf{F}=0$ and $A=\Psi\cdot\bar{x}$.

The field equations for this choice are

$$\mathbf{E}=(\nabla\times\nabla\times\mathbf{A})/(\sigma+j\omega\varepsilon)$$

and

$$\mathbf{H}=\nabla\times\mathbf{A}$$

Expanding, we obtain

$$E_x=\frac{1}{(\sigma+j\omega\varepsilon)}\left[\frac{\partial^2\Psi}{\partial x^2}+k^2\Psi\right] \qquad H_x=0 \qquad [12a]$$

$$E_y=\frac{1}{(\sigma+j\omega\varepsilon)}\left[\frac{\partial^2\Psi}{\partial x\partial y}\right] \qquad H_y=\frac{\partial\Psi}{\partial z} \qquad [12b]$$

$$E_z=\frac{1}{(\sigma+j\omega\varepsilon)}\left[\frac{\partial^2\Psi}{\partial x\partial z}\right] \qquad H_z=\frac{-\partial\Psi}{\partial y} \qquad [12c]$$

2.3.3.1 Travelling Wave solution

Now it is required to find solutions to the Vector Helmholtz equation

$$\nabla^2\mathbf{A}+k^2\mathbf{A}=0$$

The solution of the equation can be found by the implementation of the separation of variables technique in rectangular co-ordinates.

Two media are present in the waveguide. Different potentials and consequently, different field expressions are expected in each. Because the tangential **E** and **H** fields in each medium must match at all points on the media interface, the z-propagation constant k_z, must be the same for each medium.

For medium 1, the saturated medium, the form of the forward travelling wave may be found by assuming a solution of the form

$$A=\Psi=X(x)Z(z)$$

where X(x) and Z(z) are harmonic functions of their respective variables. As mentioned, appropriate waveguide excitation and a waveguide width much larger than the plate separation are sufficient conditions to neglect any y-coordinate dependence of the field solution.

Substituting the assumed solution back into the Helmholtz equation and dividing through by X(x)Z(z) yields

$$\frac{1}{X}(\frac{\partial^2 X}{\partial x^2}) + \frac{1}{Z}(\frac{\partial^2 Z}{\partial z^2}) + k^2 = 0 \qquad [13]$$

Each term depends on only one coordinate which may vary independently of the others. Therefore, a solution to equation [13] requires each term to be set equal to a constant. In addition, as mentioned previously, the boundary conditions at the media interface require the z-dependence of the fields in both regions to be equal. Therefore, for a forward travelling wave, the z-dependence of the field expressions for both media has the form $e^{-jk_z z}$.

The x-dependence of the fields may be sought. Although completely arbitrary, for later convenience the x-dependent term in equation [13] is equated to a negative constant $-k_{x1}^2$. Thus

$$\frac{1}{X}(\frac{\partial^2 X}{\partial x^2}) = -k_{x1}^2$$

The natural solution of this equation is

$$X(x)=K\sin(k_{x1}x)+L\cos(k_{x1}x)$$

However, the boundary condition for E_Z at x=0 using equation [12c] is

$$E_z = \frac{1}{(\sigma+j\omega\varepsilon)}[\frac{\partial^2\Psi}{\partial x\partial z}]_{x=0} = 0$$

Therefore

$$\frac{\partial X}{\partial x}_{x=0} = Kk_{x1}\cos(k_{x1}\cdot 0) - Lk_{x1}\sin(k_{x1}\cdot 0) = 0$$

which demands

$$K=0$$

and leaves

$$X(x) = L\cos(k_{x1}x)$$

Combining the X(x) and Z(z) solutions for the forward travelling wave results in the scalar potential

$$\Psi_1^+ = C_1^+ \cos (k_{x1}x) e^{-jk_z z}$$

where the positive superscript denotes a forward travelling wave and C_1^+ is a constant which depends on the excitation applied.

For medium 2, the depleted region, the same vector Helmholtz equation applies.

The general solution of the partial differential equation describing the x-dependence has the form

$$X(x) = M\cos (k_{x2}x) + N\sin (k_{x2}x)$$

The boundary condition states that $E_Z=0$ at x=a. Thus

$$E_z = 0 = \frac{1}{(\sigma+j\omega\varepsilon)}[\frac{\partial^2\Psi}{\partial x \partial z}]_{x=a}$$

This boundary condition will not eliminate any terms in the natural solution describing the x-dependence of the fields. Therefore, a variable transformation of x → (a-x) is made. Now

$$X(x) = M\cos (k_{x2}(a-x)) + N\sin (k_{x2}(a-x))$$

and

$$\frac{\partial X}{\partial x}_{x=a} = 0 = Mk_{x2}\sin (k_{x2}\cdot 0) + Nk_{x2}\cos (k_{x2}\cdot 0)$$

Therefore

$$N=0$$

and the complete solution for the potential describing the forward travelling wave in region 2 is

$$\Psi_2^+ = C_2^+ \cos (k_{x2}(a-x)) e^{-jk_z z}$$

If the constants $-k_{x1}^2$ and $-k_{x2}^2$ are substituted back into equation [13] for their respective medium, relations between the x-propagation constants k_{x1} and k_{x2}, the z-propagation constant k_z, and a constant of each medium, k, result. The relations are known as the separation equations and have the form

$$k_{x1}^2 + k_z^2 = k_1^2 \qquad [14a]$$

$$k_{x2}^2 + k_z^2 = k_2^2 \qquad [14b]$$

where $k_1^2=-j\omega\mu_1(\sigma_1+j\omega\varepsilon_1)$ and $k_2^2=-j\omega\mu_2(\sigma_2+j\omega\varepsilon_2)$.

By exactly the same procedure, but using $Z(z) = e^{+jk_z z}$ to describe waves directed in the negative z-direction, the scalar potentials for these waves are found to be

$$\Psi_1^- = C_1^- \cos (k_{x1}x)e^{+jk_z z}$$

$$\Psi_2^- = C_2^- \cos (k_{x2}(a-x))e^{+jk_z z}$$

Substitution of the above potentials into equations [12] results in the electric and magnetic field expressions. The positively z-directed travelling waves have the form

<u>For medium 1</u>

$$E_{x1}^+ = [C_1^+/(\sigma_1+j\omega\varepsilon_1)]k_z^2\cos (k_{x1}x)e^{-jk_z z} \qquad [15a]$$

$$E_{z1}^+ = [jC_1^+/(\sigma_1+j\omega\varepsilon_1)]k_{x1}k_z\sin (k_{x1}x)e^{-jk_z z} \qquad [15b]$$

$$H_{y1}^+ = -jC_1^+k_z\cos (k_{x1}x)e^{-jk_z z} \qquad [15c]$$

<u>For medium 2</u>

$$E_{x2}^+ = [C_2^+/(\sigma_2+j\omega\varepsilon_2)]k_z^2\cos (k_{x2}(a-x))e^{-jk_z z} \qquad [16a]$$

$$E_{z2}^+ = [-jC_2^+/(\sigma_2+j\omega\varepsilon_2)]k_{x2}k_z\sin (k_{x2}(a-x))e^{-jk_z z} \qquad [16b]$$

$$H_{y2}^+ = -jC_2^+k_z\cos (k_{x2}(a-x))e^{-jk_z z} \qquad [16c]$$

The negatively z-directed waves have the form

<u>For medium 1</u>

$$E_{x1}^- = [C_1^-/(\sigma_1+j\omega\varepsilon_1)]k_z^2\cos (k_{x1}x)e^{+jk_z z} \qquad [17a]$$

$$E_{z1}^{-} = [-jC_1^{-}/(\sigma_1+j\omega\varepsilon_1)]k_{x1}k_z\sin(k_{x1}x)e^{+jk_z z} \qquad [17b]$$

$$H_{y1}^{-} = jC_1^{-}k_z\cos(k_{x1}x)e^{+jk_z z} \qquad [17c]$$

<u>For medium 2</u>

$$E_{x2}^{-} = [C_2^{-}/(\sigma_2+j\omega\varepsilon_2)]k_z^2\cos(k_{x2}(a-x))e^{+jk_z z} \qquad [18a]$$

$$E_{z2}^{-} = [jC_2^{-}/(\sigma_2+j\omega\varepsilon_2)]k_{x2}k_z\sin(k_{x2}(a-x))e^{+jk_z z} \qquad [18b]$$

$$H_{y2}^{-} = jC_2^{-}k_z\cos(k_{x2}(a-x))e^{+jk_z z} \qquad [18c]$$

2.3.3.2 Short-Circuit Field Solution

Obtaining a travelling wave in an actual physical system requires either a matched load or a line length much greater than the skin depth of the guided wave. Neither of these conditions is practical nor cost effective in actual practice. When both positively and negatively z-directed waves occur simultaneously, they result in the formation of standing waves. In general, standing waves imply very non-uniform field distributions along the waveguide. However, as previously mentioned, the presence of axial electric fields in addition to the transverse electric fields, acts to level out these non-uniformities.

Both open-circuit and short-circuit waveguide terminations create similar overall electric field patterns. An open-circuit load on the buried electrode configuration is easier to build than a short-circuit. However, a short-circuit load can be modelled much more accurately than an open-circuit might. Leakage currents and radiation of energy at a buried open-circuit load may cause the load to behave differently than an ideal open-circuit. However, if the ratio of plate separation to waveguide width is very small, as is the case for this thesis, then the load approaches an ideal open-circuit and can be accurately modelled with a simple open-circuit. For this reason, the open-circuit solution will be emphasized when discussions of waveguide heating commence in Chapter 5.

The short-circuit gives rise to the boundary condition that $E_X=0$ at z=0. If this is so then

$$E_x = 0 = E_x^+ + E_x^- \text{ at } z=0$$

Substitution of the field expressions for region 1 yields

$$\frac{C_1^+ k_z^2 \cos(k_{x1}x) e^{-jk_z \cdot 0}}{(\sigma_1 + j\omega\varepsilon_1)} = \frac{-C_1^- k_z^2 \cos(k_{x1}x) e^{+jk_z \cdot 0}}{(\sigma_1 + j\omega\varepsilon_1)}$$

which leaves

$$C_1^+ = -C_1^-$$

The standing wave fields may then be calculated by the addition of the forward and reverse wave contributions and may be expressed in terms of only one constant, C_1^+. The fields then have the form

$$E_{x1} = E_{x1}^+ + E_{x1}^- = [-j2C_1^+/(\sigma_1+j\omega\varepsilon_1)]k_z^2\cos(k_{x1}x)\sin(k_z z) \qquad [19a]$$

$$E_{z1} = E_{z1}^+ + E_{z1}^- = [j2C_1^+/(\sigma_1+j\omega\varepsilon_1)]k_{x1}k_z\sin(k_{x1}x)\cos(k_z z) \qquad [19b]$$

$$H_{y1} = H_{y1}^+ + H_{y1}^- = -j2C_1^+k_z\cos(k_{x1}x)\cos(k_z z) \qquad [19c]$$

The same boundary condition applies for region 2.

$$E_{x2} = 0 = E_{x2}^+ + E_{x2}^- \text{ at } z=0$$

Substitution of the field expressions into the above equation yields

$$\frac{C_2^+ k_z^2 \cos(k_{x2}(a-x)) e^{-jk_z \cdot 0}}{(\sigma_2 + j\omega\varepsilon_2)} = \frac{-C_2^- k_z^2 \cos(k_{x2}(a-x)) e^{+jk_z \cdot 0}}{(\sigma_2 + j\omega\varepsilon_2)}$$

which leads to

$$C_2^+ = -C_2^-$$

and the standing wave field expressions in terms of one constant C_2^+ become

$$E_{x2} = E_{x2}^+ + E_{x2}^- = [-j2C_2^+/(\sigma_2+j\omega\varepsilon_2)]k_z^2\cos(k_{x2}(a-x))\sin(k_z z) \qquad [20a]$$

$$E_{z2} = E_{z2}^+ + E_{z2}^- = [-j2C_2^+/(\sigma_2+j\omega\varepsilon_2)]k_{x2}k_z\sin(k_{x2}(a-x))\cos(k_z z) \qquad [20b]$$

$$H_{y2} = H_{y2}^+ + H_{y2}^- = -j2C_2^+k_z\cos(k_{x2}(a-x))\cos(k_z z) \qquad [20c]$$

2.3.3.3 Open-Circuit Field Solution

Termination of the transmission line with an open-circuit and neglecting fringing allows use of the boundary condition stating $H_y=0$ at $z=0$. In region 1

$$H_{y1} = 0 = H_{y1}^+ + H_{y1}^- \text{ at } z=0$$

which, when the field expressions are substituted, gives,

$$-jC_1^+k_z\cos(k_{x1}x)e^{-jk_z \cdot 0} = -jC_1^-k_z\cos(k_{x1}x)e^{+jk_z \cdot 0}$$

which leads to:

$$C_1^+ = C_1^-$$

and the standing wave fields in region 1 in terms of one constant C_1^+ take on the form:

$$E_{x1} = E_{x1}^+ + E_{x1}^- = [2C_1^+/(\sigma_1+j\omega\varepsilon_1)]k_z^2\cos(k_{x1}x)\cos(k_z z) \qquad [21a]$$

$$E_{z1} = E_{z1}^+ + E_{z1}^- = [2C_1^+/(\sigma_1+j\omega\varepsilon_1)]k_{x1}k_z\sin(k_{x1}x)\sin(k_z z) \qquad [21b]$$

$$H_{y1} = H_{y1}^+ + H_{y1}^- = -2C_1^+k_z\cos(k_{x1}x)\sin(k_z z) \qquad [21c]$$

Similarly in region 2,

$$C_2^+ = C_2^-$$

and the standing wave fields in region 2 in terms of one constant C_2^+ take on the form,

$$E_{x2} = E_{x2}^+ + E_{x2}^- = [2C_2^+/(\sigma_2+j\omega\varepsilon_2)]k_z^2\cos(k_{x2}(a-x))\cos(k_z z) \qquad [22a]$$

$$E_{z2} = E_{z2}^+ + E_{z2}^- = [-2C_2^+/(\sigma_2+j\omega\varepsilon_2)]k_{x2}k_z\sin(k_{x2}(a-x))\sin(k_z z) \qquad [22b]$$

$$H_{y2} = H_{y2}^+ + H_{y2}^- = -2C_2^+k_z\cos(k_{x2}(a-x))\sin(k_z z) \qquad [22c]$$

The final forms of the fields have been found for both the open and short-circuit terminations. Examination of the field expressions reveals that the field magnitudes for both cases differ only in their axial dependence. For instance, whereas a particular field component for the short-circuit case may vary cosinusoidally, the corresponding field component for the open-circuit case will vary sinusoidally. What remains to be determined are the constants present in the field expressions. They are dependent on boundary conditions and the excitation applied.

2.3.3.4 Development of the Transcendental Equation

The propagation constants k_{x1},k_{x2} and k_z shall be determined through application of the boundary conditions. At the plane interface between the two media, the tangential components of the electric fields must be continuous.

$$E_{z1}^{+} = E_{z2}^{+} \text{ at } x=d$$

Substitution of the electric field expressions into the relation yields

$$\frac{-C_1^{+}k_{x1}\sin(k_{x1}d)}{(\sigma_1+j\omega\varepsilon_1)} = \frac{C_2^{+}k_{x2}\sin(k_{x2}(a-d))}{(\sigma_2+j\omega\varepsilon_2)}$$

The tangential components of the magnetic field must also be continuous at the media interface

$$C_1^{+}\cos(k_{x1}d) = C_2^{+}\cos(k_{x2}(a-d))$$

Division of the second equation into the first leads to the desired relationship

$$\frac{-k_{x1}\tan(k_{x1}d)}{(\sigma_1+j\omega\varepsilon_1)} = \frac{k_{x2}\tan(k_{x2}(a-d))}{(\sigma_2+j\omega\varepsilon_2)} \qquad [23]$$

The above transcendental equation, together with the separation equations derived earlier [14], which are repeated here for convenience, form a system of three equations in three unknowns (k_{x1},k_{x2},k_z).

$$k_{x1}^2 + k_z^2 = k_1^2 \qquad [14a]$$

$$k_{x2}^2 + k_z^2 = k_2^2 \qquad [14b]$$

This system of equations cannot, however, be solved algebraically and solutions must be found numerically. In addition, after the propagation constants are found, the constants C, which are dependent on the waveguide excitation, must also to be determined. It is convenient to express all the field expressions in terms of one of these constants. This can be done by using the continuity of the magnetic field at the media interface to yield

$$C_1^+/C_2^+ = \cos(k_{x2}(a-d))/\cos(k_{x1}d) \qquad [24]$$

Having calculated the analytic solutions, some comparisons can then be made with the TEM transmission line equivalent circuit. Examination of the exact field solutions has shown that the fields approach a TEM form when the x-directed propagation constants, k_{x1} and k_{x2}, become sufficiently small. With regard to the transcendentai system of equations discussed above, constraining the magnitudes of the propagation constants allows the replacement of the tangent functions by their respective arguments. This results in an algebraic system of equations which can be readily solved as discussed below.

2.4 The Small Argument Solution

2.4.1 The Small Argument of the Complex Tangent Function

In order to confidently replace the complex tangent function with its argument, the limiting behaviour of the complex tangent function must be considered. It can be shown that the complex tangent function and its complex argument are in agreement to within 10% error when the magnitude of both the real and imaginary components of the argument is not greater than 0.3. Subject to this restriction, comparison of the resultant fields using exact and approximate propagation constants has shown that they are also in agreement to within 10% error.

2.4.2 The Small Argument Propagation Constant Expressions

Given the transcendental equation [23], and limiting the arguments $k_{x1}d$ and $k_{x2}(a-d)$ to the small argument limits results in the expression

$$\frac{-k_{x1}(k_{x1}d)}{\sigma_1+j\omega\varepsilon_1}=\frac{k_{x2}(k_{x2}(a-d)}{\sigma_2+j\omega\varepsilon_2}$$

Using the separation equations [14] allows the derivation of the expressions for the propagation constants k_{x1},k_{x2} and k_z.

$$k_{x1}^2=\frac{j\omega\mu((\sigma_1+j\omega\varepsilon_1)-(\sigma_2+j\omega\varepsilon_2))}{1+\dfrac{(\sigma_2+j\omega\varepsilon_2)d}{(\sigma_1+j\omega\varepsilon_1)(a-d)}} \qquad [25]$$

$$k_{x2}^2=\frac{-j\omega\mu((\sigma_1+j\omega\varepsilon_1)+(\sigma_2+j\omega\varepsilon_2))}{1+\dfrac{(\sigma_1+j\omega\varepsilon_1)(a-d)}{(\sigma_2+j\omega\varepsilon_2)d}} \qquad [26]$$

$$k_z^2=\frac{j\omega\mu(\sigma_1+j\omega\varepsilon_1)\frac{a}{d}}{1+\dfrac{(\sigma_1+j\omega\varepsilon_1)(a-d)}{(\sigma_2+j\omega\varepsilon_2)d}} \qquad [27]$$

2.5 Comparison of the Approximate Model Field Solutions

Two models for approximating the wave propagation constants have been presented: the transmission line and small argument models. Although each model corresponds to fundamentally different field configurations, both models predict identical expressions for the axial propagation constant, k_z. The transmission line approximation is based on the assumption of perfect TEM (transverse electromagnetic) fields within the waveguide. On the other hand, the small argument approximation does not assume a particular field pattern. It is merely a simplification of the exact analytic field solution. Therefore, the fields associated with it are approximately those of the analytic solution.

Although both models predict the same axial propagation constant, subtle differences also exist between the two solutions. For instance, the TEM assumption of the transmission line equivalent circuit stipulates that no transverse field variations occur. As a result, the equivalent circuit cannot theoretically predict the transverse propagation constants k_{x1} and k_{x2}. On the other hand, the small argument approximation, which is based on the analytic solution, accounts for transverse field variation and consequently, gives expressions for k_{x1} and k_{x2}.

Practical application of either model requires knowledge of its region of validity. The transmission line model is based on pure TEM wave propagation. Thus, in theory, it can only be valid for situations of pure TEM wave behaviour. In the case of the small argument approximation, the model is valid provided the arguments of the complex tangent functions do not exceed the previously mentioned limiting value. These limits for the tangent arguments may be translated to limits in physical system parameters through the process of monitoring the tangent arguments as a function of system parameters. The important point is that since the axial propagation constant expressions for both models are identical, the range of validity of the small argument model can be directly applied to the transmission line equivalent circuit model. In other words, the TEM model may be valid for non-TEM field distributions. Finally, it should again be emphasized that both the TEM and small argument models are able to predict the axial heating distribution for forward travelling waves. However, the small argument model reveals the correct transverse heating distribution, whereas the TEM model does not. In fact, the TEM model assumes uniform heating of any cross-section.

3. Calculation of the Propagation Constants

3.1 General

In order for the electromagnetic field expressions derived in Chapter 2 to accurately describe the field behaviour within the waveguide for all possible combinations of system parameters, more accurate values of the propagation constants k_{x1}, k_{x2} and k_z must be known. These propagation constants determine the quantitative properties the electromagnetic fields inside the waveguide will have. They are complex numbers which contain information about wavelength and wave attenuation. In any waveguide problem, the propagation constants are determined by application of boundary conditions to the electromagnetic field expressions. As shown in Chapter 2, application of field boundary conditions at the plane interface between the waveguide media has resulted in equation [23]. Equations [23] and [14] form a system of equations which quantify the propagation constants k_{x1}, k_{x2} and k_z.

The periodicity of the complex tangent functions in equation [23] means that an infinite number of solutions exist. These solutions cannot be found analytically. They must be found using numerical methods. There are several techniques which can be used to solve for them. A common technique, known as the Muller method[10], has been chosen because it is known to work reliably, is well documented, and is easily implemented. (More information will be given on Muller's method in a later section) Muller's method is designed to solve a single transcendental equation in one variable having the form f(a)=0 where both 'a' and the function may be complex. Therefore, for the problem under study here, it is necessary to choose which of the 3 propagation constants should initially be determined. (The remaining two are found through back-substitution into equations [14]) Most wave propagation problems are concerned with the propagation constant describing wave behaviour in the direction of wave propagation. Thus, the axial propagation constant k_z is solved for. The resultant transcendental equation is

$$\frac{\sqrt{k_1^2-k_z^2}\cdot\tan\left[\sqrt{k_1^2-k_z^2}\cdot d\right]}{\sigma_1+j\omega\varepsilon_1}+\frac{\sqrt{k_2^2-k_z^2}\cdot\tan\left[\sqrt{k_2^2-k_z^2}\cdot(a-d)\right]}{\sigma_2+j\omega\varepsilon_2}=0 \qquad [28]$$

With an infinite number of solutions for k_z existing, a method of organizing them must be found. This will simplify the search for solutions and also help determine which solutions are important. In waveguide theory, a common graph used to organize the field configurations found in a typical waveguide consists of a plot of the frequency (f) versus the axial phase constant (β) as given in standard engineering textbooks. Figure 6 shows a graph of f vs. β for a waveguide with properties: a=15m, d=0.8a, σ_1=0 S/m, σ_2=0 S/m, $\varepsilon_1=11\varepsilon_0$, $\varepsilon_2=3\varepsilon_0$, and $\mu_1=\mu_2=\mu_0$. The graph consists of several curves which represent different sets of solutions for the phase constant (β). Each curve corresponds to a particular mode in the waveguide. The curve designated (a) is referred to as the dominant mode, curve (b) corresponds to the second order mode, curve (c) corresponds to the third order mode, and so on. Although curves up to only the third order mode are shown, in theory an infinite number of mode branches exist for a single waveguide.

Because a propagation constant is generally complex, with one component being the phase constant (β) and the other corresponding to the

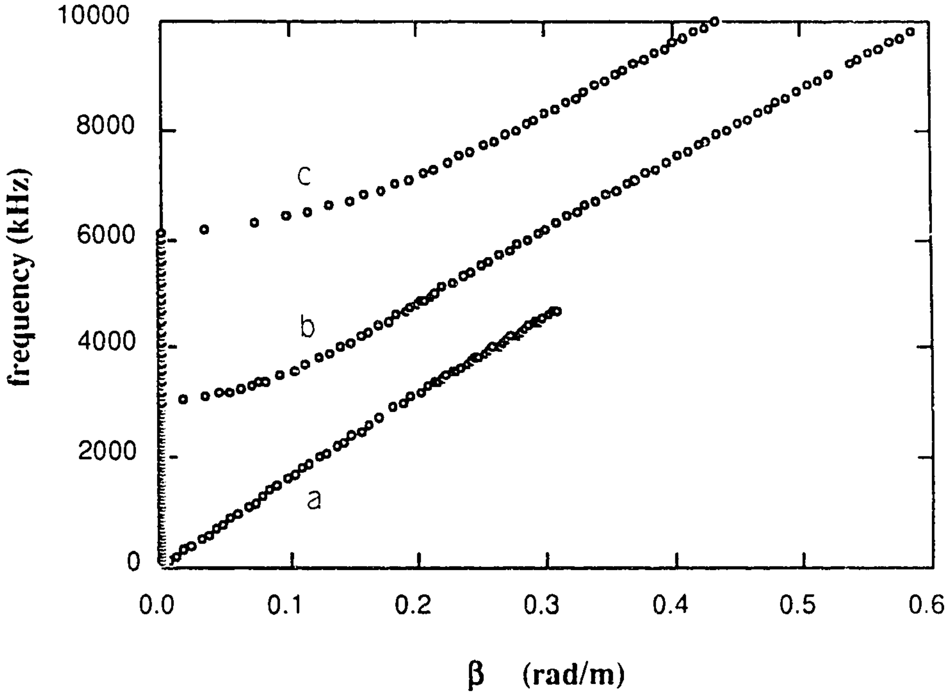

Figure 6. *A plot of frequency versus phase constant for a parallel-plate waveguide filled with a lossless layered medium. Plate separation =15m, saturated region thickness = 12.75m and remaining medium electrical parameters as given in Table 1.*

wave attenuation (α), a plot of the frequency (f) versus the wave attenuation (α) also exists. This plot, although very important to the understanding of wave propagation, is often omitted in textbooks. It consists of a set of curves which correspond to the wave attenuation (α) of the distinct modes present within a waveguide. Figure 7 shows a typical plot of frequency (f) versus axial attenuation (α) for a waveguide with the same properties as the one in Figure 6. The curve or branch lying directly on the vertical (f) axis and labelled (a) corresponds to the dominant mode. The curve marked (b) corresponds to the second order mode, and so on. If both the f vs. β and f vs. α graphs are shown together, they form a mode plot. Such a plot gives detailed information about the characteristics of the electromagnetic fields within a waveguide. Figure 8 shows such a plot for the same waveguide associated with Figures 6 and 7. However, to make the graph more compact, the negative of the attenuation ($-\alpha$), has been plotted, thereby allowing a common frequency axis to be used.

A large amount of information concerning the field behaviour within the waveguide may be extracted from the mode plot in Figure 8. Firstly, if the dominant mode curves (labelled (a)) are examined, it is seen that the wave attenuation is very small for all frequencies. Thus, the dominant mode fields are not significantly attenuated for any frequency shown. In addition, the corresponding phase constant (β) is linearly related to the frequency (f) and has a positive finite value (except at $f=0$ where $\beta=0$ always). In terms of the electromagnetic fields, the dominant mode propagates. It has a positive and finite phase constant (β) and, therefore, a positive and finite wavelength ($\lambda=2\pi/\beta$) for all frequencies except $f=0$. If the second order mode (designated (b)) is examined, a different behaviour

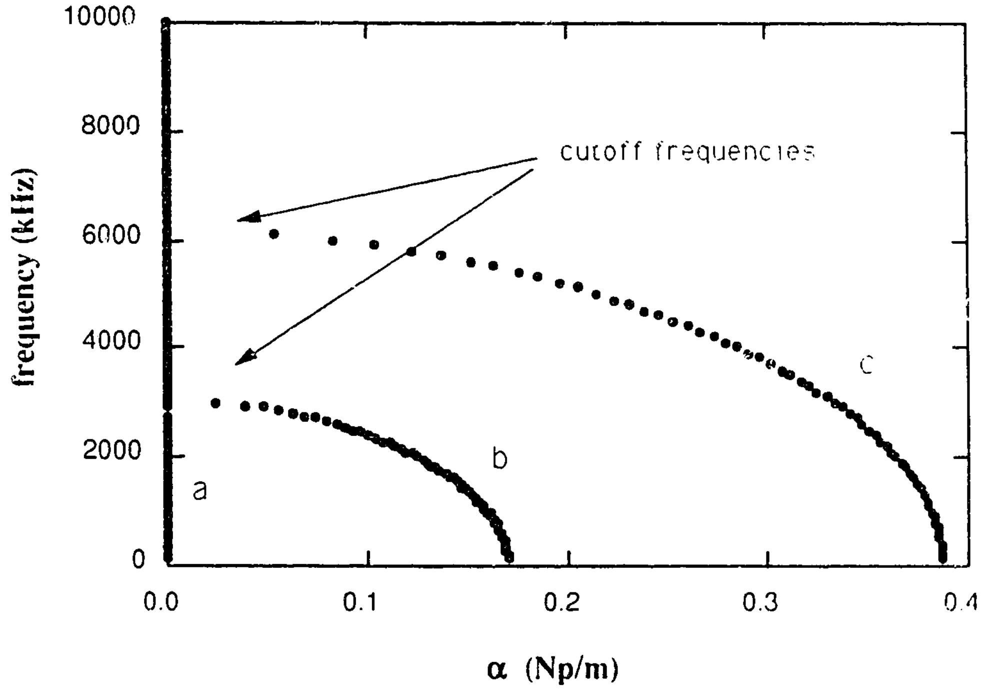

Figure 7. *A plot of frequency versus wave attenuation for a parallel-plate waveguide filled with a layered lossless medium. Plate separation = 15m, saturated region thickness = 12.75m and remaining medium electrical parameters as given in Table 1.*

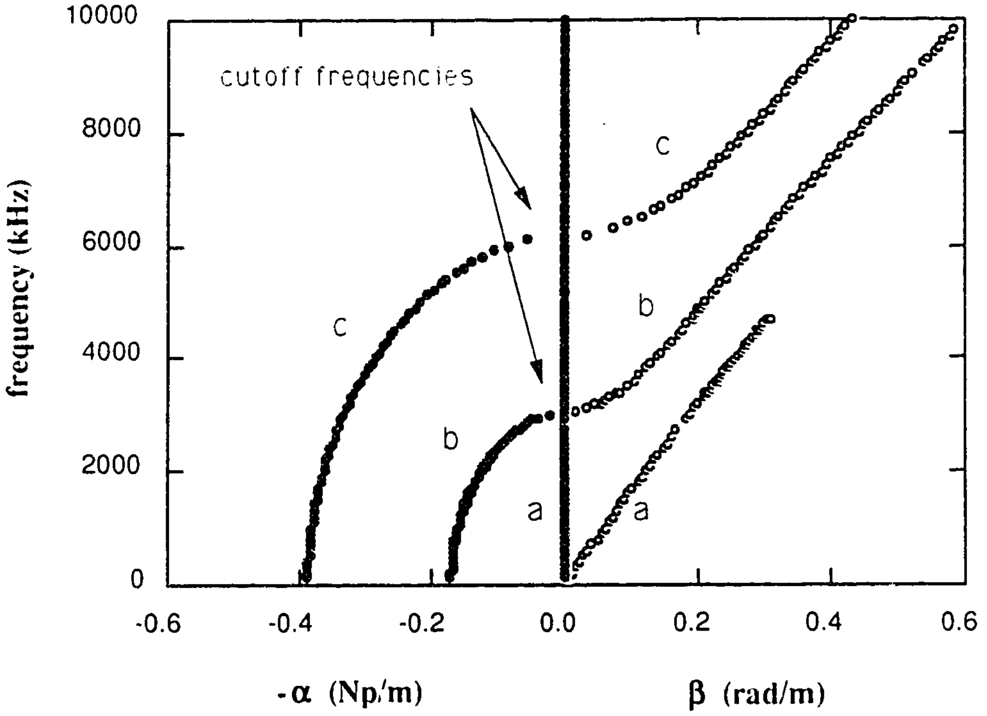

Figure 8. *A mode plot for a parallel-plate waveguide filled with a lossless layered medium. Plate separation = 15m, saturated region thickness = 12.75m and remaining electrical medium parameters as given in Table 1.*

is observed. Below a certain frequency of approximately 3 MHz, the phase constant (β) has a value that drops rapidly to zero at lower frequencies and the attenuation (α) has a finite, non-zero value. Below 3 MHz, the second order mode has a very large wavelength (λ) and is significantly attenuated. The mode does not truly propagate but rather exhibits only pure exponential decay. It is termed an 'evanescent' mode. However, above 3 MHz the opposite occurs. The attenuation (α) becomes very small and the phase constant (β) becomes large and positive. The second order mode propagates. The frequency 3 MHz is called the 'cutoff frequency' of the second order mode. The behaviour of higher order modes is similar, as shown by the third order mode curve (designated (c)).

To transmit energy down the waveguide in the most easily predictable fashion, it can be easily seen from Figure 8 that the operating frequency should be below 3 MHz. Only the dominant mode will then have a positive and finite phase constant and propagate energy down the waveguide. An analysis of the electromagnetic fields inside the waveguide (excluding the vicinity of the waveguide feedpoint where fields due to evanescent modes may be present) would require the evaluation of the field expressions with only the dominant mode propagation constants. For operating frequencies above 3 MHz, the dominant, second, and possibly higher order modes would propagate and, thus, the total field inside the waveguide would consist of a superposition of several modes. For most waveguide applications it is desired to operate in a frequency range such that only the dominant mode propagates.

Before proceeding, a brief word will be said concerning the common view of waveguide modes. In parallel-plate waveguides filled with a single medium, the transverse propagation constants are determined solely by application of the perfectly conducting plate boundary conditions. This results in transverse propagation constants for a given mode, which are dependent only on waveguide geometry and not on waveguide material properties or the frequency of operation. Therefore, in a parallel-plate waveguide filled with only one medium, only the axial propagation constant k_z, changes with frequency. The transverse propagation constant,k_x, is fixed for a given mode. However, if a second layer of material is introduced into the waveguide so that the plane interface between the two media is parallel to the conducting plates, the solution becomes more involved. The form of the transverse field variations still depends on the plate boundary conditions. However, each medium has its own distinct transverse propagation constant which can only be determined from the field boundary conditions at the media interface. Because the boundary conditions must be satisfied at all points on the interface, the wave velocities and, thus, the axial propagation constants for both media, must be equal. The axial propagation constants are frequency dependent. Therefore, examination of the separation equations (eqns. [14]) shows that the transverse propagation constants must be frequency dependent as well. This results in transverse field patterns which are not identical for all points on a particular mode branch. *In conclusion, for a waveguide filled with more than a single medium, the transverse field pattern is not a signature for a particular mode. The transverse field pattern changes with frequency for any given mode branch.*

3.2 The Muller Complex Rootfinding Method

A well known and widely used method for solving complex-valued transcendental functions is the Muller method. For purposes of understanding, the Muller method can be thought of as an extension of the Secant method for solving real-valued functions. As the name suggests, the Secant method is based on convergence to a function root by approximation of the function with successive secants, with each successive secant more accurately approximating the function in the vicinity of its root.

The Muller method improves upon this scheme by approximating function behaviour with successive parabolas. Not only does this improve convergence, it also allows complex roots to be determined through application of the familiar quadratic root finding formula. A parabola is only uniquely determined if three points on it are known. Therefore, three initial guesses must be supplied to the Muller algorithm. Experience has shown that the Muller method is almost always effective for the calculation of the complex roots of complex-valued functions as well as for the calculation of real roots of real functions.

It should be mentioned that a danger exists when solving for a root of a complex-valued function. It generally involves the calculation of complex square-roots. When performing the operation of 'complex square-root' on a complex number, an infinite number of valid solutions exist, all differing in their phases by an integer multiple of π. It is essential that the correct root be calculated. Generally, the root of interest is the principal root. It is the root which differs from its radicand by the smallest phase angle and whose imaginary component retains the same sign as the radicand. To arrive at the principal root, check statements are inserted at critical points within each complex root-solving program. The statements check the sign of the root and compare it to the radicand to ensure that the signs of the imaginary components are the same. This is equivalent to saying that the principal root is found. When solving the transcendental equation [28], the generation of non-principal roots generally results in subtle changes in the final answer. An unexpected sign change could occur in either the real or imaginary component, as well as a slight change in magnitude of the final answer.

3.3 The Numerical Procedure for Generation of Mode Plots

3.3.1 Dominant Mode Branch

Calculation of the dominant mode branch is the initial step in the construction of a mode plot. As detailed in Chapter 2, for the range of system parameters shown in Table 1, the approximate waveguide models result in closed form expressions which accurately predict the dominant mode propagation constants for frequencies of up to a few 100kHz. A computer program called 'RepeatingMuller' uses these models (eqn. [27]) for sources of initial guesses when calculating the dominant mode branch using the Muller algorithm. However, three distinct initial guesses are required by the algorithm, so two simple arithmetic algorithms are applied to eqn. [27] to generate two additional guesses. RepeatingMuller calculates the function roots over a range of frequencies, usually beginning at low frequencies (≈100 kHz). A complete documented listing of the program is given in Appendix 3.

3.3.2 Location of Higher Order Mode Branches

After calculation of the dominant mode branch, the higher mode branches remain to be mapped. As a first step, single points on each higher order mode branch must first be found. A program called 'SingleStepMuller' has been developed for this. It is similar to the program RepeatingMuller. However, it calculates roots only at singular frequencies using root approximations for k_z which are user entered. A documented program listing is given in Appendix 3.

It is logical to assume that the behaviours of the mode plots for the cases of interest here have essentially the same form as the mode plot for the air-filled waveguide shown in Figure 8. By carefully choosing initial guesses for k_z very near the dominant mode branch and then progressively moving in the direction of the higher order mode branches, points on the second and third order mode branches can be easily located. After these points are found, the entire branches may be generated.

3.3.3 Completion of Higher Order Mode Branches

With roots on the second and third order mode branches located, the entire branches may be constructed. If frequency is carefully changed in small increments from the known root, it is expected that the new root will not change significantly. By beginning at a frequency very near the the known root and using that root as an initial guess, the point on that branch at the nearby frequency can be found. This principle can be applied to generate the branch segments above and below a known root. The program developed for this purpose was another variation of RepeatingMuller and was called 'FillingMuller'. The segments either above or below the known point on a branch are calculated by carefully incrementing or decrementing the frequency and 'self-generating' that segment. A complete program listing is given in Appendix 3.

3.4 Discussion of Actual Mode Plots

By applying the procedure outlined in section 3.3 to waveguide configurations of interest, their mode plots were generated. These plots had a general form similar to the mode plot for the lossless waveguide shown in Figure 8. However, because of differences in the waveguide properties, the mode plot behaviour for the cases studied here also differed from Figure 8. Figure 9 shows the mode plot for a parallel-plate waveguide with properties: a=30m, d=0.5a, $\sigma_1=10^{-4}$S/m, $\sigma_2=10^{-6}$S/m, $\varepsilon_1=11\varepsilon_0$, $\varepsilon_2=3\varepsilon_0$, and $\mu_1=\mu_2=\mu_0$. The dominant, second and third order mode branches are shown and are labelled (a), (b) and (c), respectively. Regions which would correspond to the mode cutoff frequencies for the modeplot in Figure 8 are labelled the second and third order 'cutoff regions'. If the modeplots shown in Figures 8 and 9 are qualitatively compared, it is clear that the second and third order mode branches behave differently than those of Figure 8. For the lossless waveguide modeplot shown in Figure 8, the second and third order mode branches exhibit a very sharp transition at cutoff. Below cutoff, the phase constants are zero, and above cutoff the attenuation constants are zero. For the low loss case shown in Figure 9, the second and third order mode branches are slightly curved at cutoff. Slightly above cutoff the attenuation curves approach some minimum non-zero value, and then begin to slowly diverge as frequency increases. Below cutoff the phase constant branches asymptotically approach zero as frequency decreases. Theoretically wave cutoff is defined as the *transition point* between wave evanescence

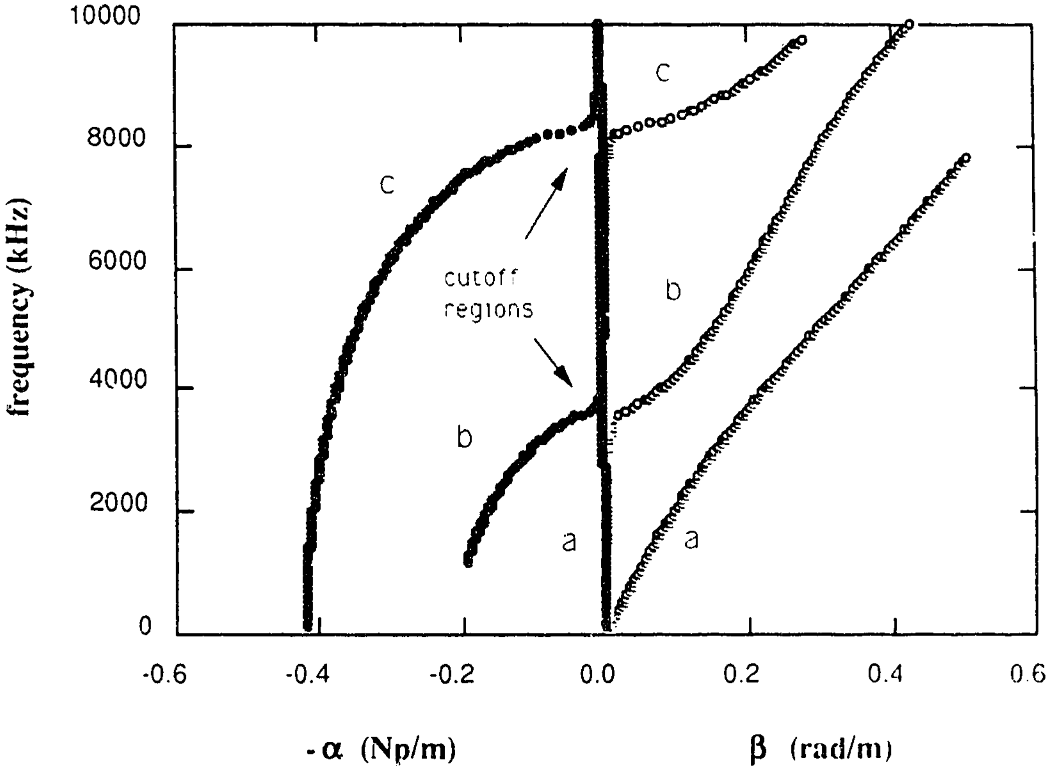

Figure 9. A mode plot for a parallel-plate waveguide filled with a low loss layered medium. Plate separation = 15m, saturated region thickness = 7.5m, saturated region electrical conductivity = 0.0001 S/m and the remaining medium electrical parameters are as given in Table 1.

(β=0,α≠0) and wave propagation (α=0,β≠0). It only truly exists for lossless waveguides. If a lossy medium is introduced into the waveguide, no definite transition point occurs. Regardless of frequency, the attenuation constant (α) and the phase constant (β) never become exactly zero (except at d.c. where β must be zero). If the losses are small, as for the waveguide with the modeplot shown in Figure 9, an approximate cutoff frequency can be defined. The waveguide behaviour will be very similar to the lossless case if it is not operated too near the cutoff frequencies.

If waveguide losses are increased, then the effects on the mode plot discussed above are intensified. Figure 10 shows the mode plot for a parallel-plate waveguide with properties: a=15m, d=0.85a, $\sigma_1=10^{-3}$S/m, $\sigma_2=10^{-6}$S/m, $\varepsilon_1=11\varepsilon_0$, $\varepsilon_2=3\varepsilon_0$, and $\mu_1=\mu_2=\mu_0$. The dominant, second and third order modes are shown and are designated (a), (b) and (c), respectively. Cutoff frequencies are difficult to determine because the second and third order mode branches do not exhibit a sharp transition. Determination of a *cutoff region* is more appropriate for this case. All modes generally exhibit significant attenuation at all frequencies, above or below the cutoff region. Therefore, even the dominant mode, propagating at a frequency well below the second order mode cutoff, would significantly attenuate. With high losses, the difference in attenuation constants between the dominant mode and the second order mode can become small enough so that the second order mode can no longer be neglected. This requires any calculation of resistive heating in the waveguide to include dominant and second order contributions. For the cases studied in this thesis, however, the attenuation constants of second order modes were generally much greater than the dominant mode propagation constants. They could therefore be neglected.

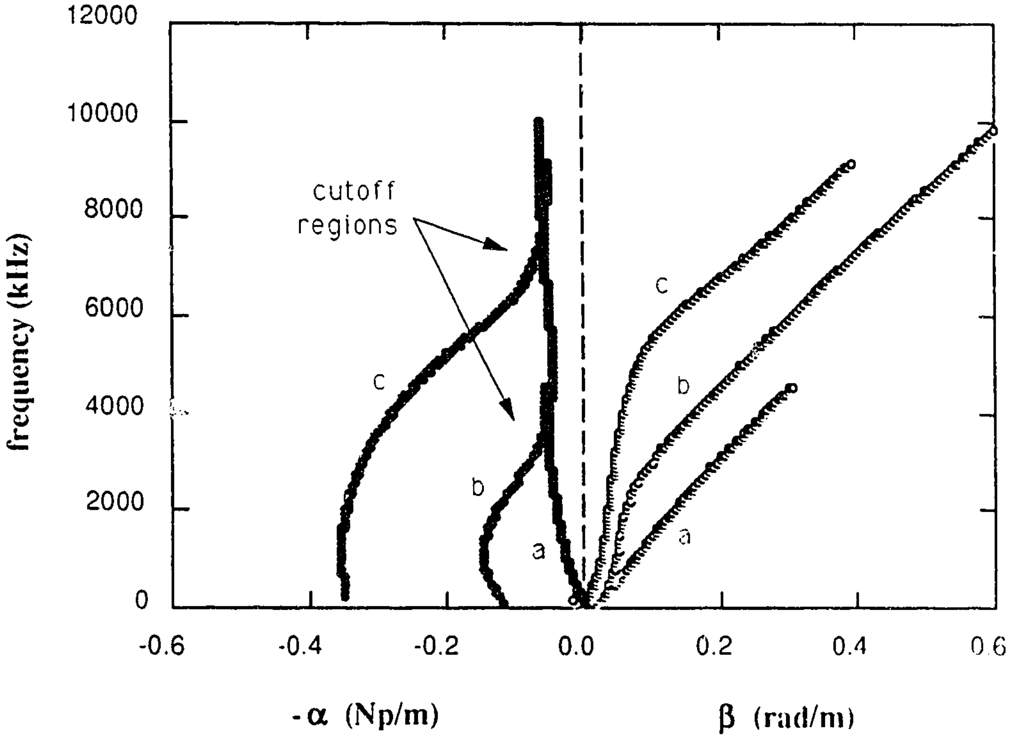

Figure 10. *A mode plot for a parallel-plate waveguide filled with a highly lossy layered medium. Plate separation = 15m, saturated region thickness = 12.75m. Saturated region electrical conductivity = 0.001 S/m and remaining medium electrical parameters are as given in Table 1*

4. Numerical Evaluation of the Analytic Electromagnetic Field Expressions

Before resistive heating inside the parallel-plate waveguide can be studied, the electric field behaviour must be known. Analysis of the electric fields may start by analyzing the forward travelling wave fields in a parallel-plate waveguide filled with only saturated oilsand. Then, a layer of dried oilsand may be introduced and the changes in the fields studied. This provides a starting point from which to proceed with more complicated wave scenarios. For instance, the phasor standing wave fields may then be examined. These are important fields because they directly relate to the time-averaged resistive heating behaviour in the waveguide. Finally, the time-domain standing waves, which provide valuable information about the current patterns inside the waveguide, can be evaluated. Examining these different forms of the fields gives important information about the resistive heating in the waveguide.

In order to calculate the various forms of the electric fields, several parameters must be determined. All of the physical system parameters are restricted to the values given in Table 1. Examination of all the field expressions derived in Chapter 2 reveals that all of the variables are known except C_1^+ and C_2^+. These constants are not independent and are related through equation [24]. To set the constants requires the waveguide excitation to be set. The simplest method is to perfectly excite the dominant mode. For the cases studied here, the waveguide excitation is chosen to be a 1kV potential placed across the plates at the waveguide input.

4.1 Evaluation of the Time-Domain Forward Travelling Wave Electric Fields

For any waveguide, the forward travelling wave electromagnetic fields are the most simple. Analysis begins by examining the forward wave fields for the parallel-plate waveguide filled with a single lossy medium, say, saturated oilsand. The electric field is purely TEM and decays exponentially as it propagates down the waveguide. Now, consider the introduction of a layer of a different medium placed adjacent to the top conducting plate as shown schematically in Figure 3. Let this medium be given the electrical properties of dried-out oilsand as given in Table 1. The fields now will no longer be of the TEM form. Figure 11 shows a picture of these fields at an arbitrary instant in time inside a waveguide with properties: $a = 15$m, $d = 0.85a$, $\sigma_1 = 10^{-3}$ S/m, $\sigma_2 = 10^{-6}$ S/m, $\varepsilon_1 = 11 \cdot \varepsilon_0$, $\varepsilon_2 = 3 \cdot \varepsilon_0$, $\mu_1 = \mu_2 = \mu_0$, $f = 340$ kHz and $L = 400$m. Each line segment represents the electric field magnitude and direction at that coordinate inside the waveguide. The program used to generate the data for the plot in Figure 11 is called 'ForwardE'. It evaluates the time-domain forward travelling axial and transverse electric field expressions given in Chapter 2 at a given time and at pre-determined gridpoints throughout the waveguide x-z plane. The plot shown in Figure 11 was generated using a program called 'Efields', which creates a picture of the electric field lines within the waveguide. At each gridpoint, the electric field is represented by a line segment of length proportional to the field magnitude and orientation corresponding to field direction. Complete program listings and operating information for the aforementioned programs is given in Appendix 3.

The electric field is predominantly transverse in the depleted region.

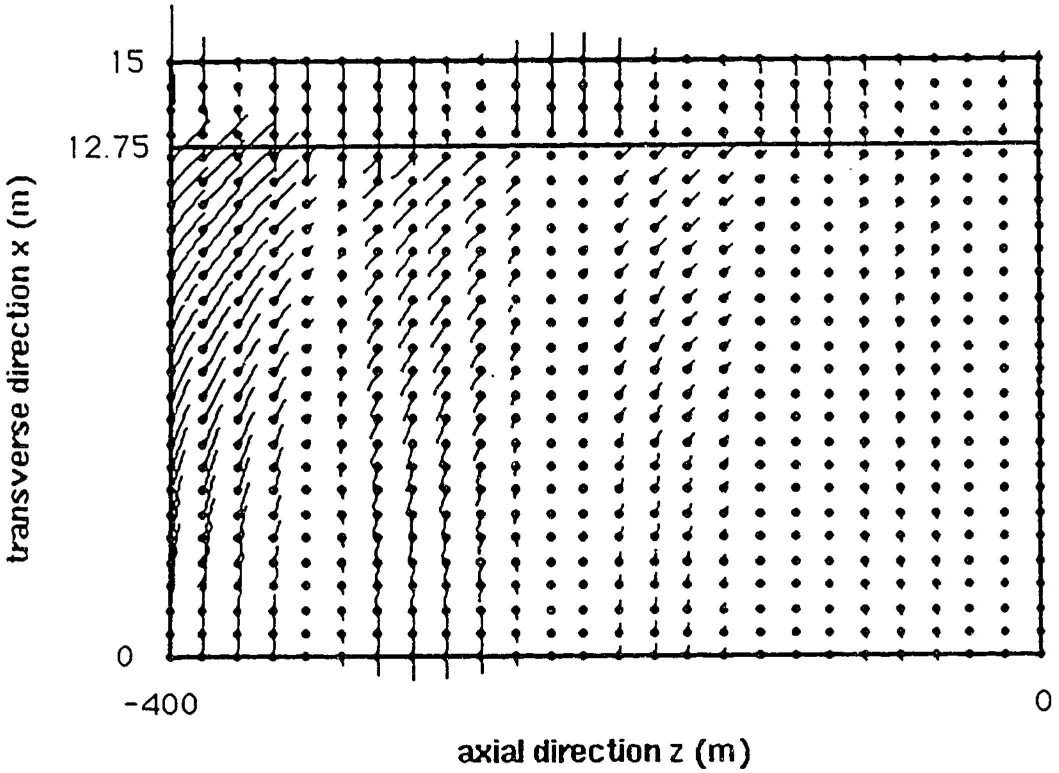

Figure 11. *A picture of the forward wave electric field lines in a waveguide with parameters: a=15m, d=0.85a, saturated region conductivity=0.001S/m, L=400m, frequency=340kHz and time=0sec. In order to view the electric fields in both regions, the electric fields of the depleted region have been scaled to 1/10 of their original value.*

In the saturated region, the transverse and axial field components are more comparable in magnitude. Near the media interface, the axial field component is generally the largest. As the lower conducting plate is approached, the axial electric field decreases until it is forced to zero at the lower plate by the perfectly conducting boundary condition. In simple terms, the increase in the ratio of the axial to transverse electric field components in the saturated region as compared to the depleted region may be explained by viewing the saturated region as a thick conducting plate adjacent to the depleted region. Thus, current from the upper plate, flowing through the depleted region, tends to return to the source through the highly conducting saturated region. If the bottom plate were absent, the field would not become transverse at the bottom of the saturated region but rather, would become entirely axial. The presence of the lower plate, with its inherent boundary conditions, forces the axial field to become zero at the bottom plate as Figure 11 shows.

Because of the simple and easily understood form of the travelling wave (i.e. the fields exhibit no axial magnitude variation other than simple wave decay), it may be used to extrapolate the behaviour of the more complex standing wave fields. For instance, the relative amount of axial current present in the saturated region can be measured via the ratio of axial to transverse electric field magnitudes, $|E_z|/|E_x|$. A ratio of one denotes equal axial and transverse field magnitudes. A ratio greater than one denotes a comparatively larger axial field and a ratio less than one denotes a larger transverse field. The usefulness of the ratio $|E_z|/|E_x|$ for the forward travelling wave comes from the possibility of it being used as a criterion for predicting the occurrence of uniform axial heating for the standing wave case. This is further discussed in Chapter 5.

4.2 Evaluation of the Phasor Standing Wave Electric Fields

An objective of this thesis is to analyze resistive heating due to standing waves in the parallel-plate waveguide. The standing waves are created by terminating the waveguide with either an open or short-circuit. The field solutions for both cases are very similar. However, for practical reasons mentioned in Chapter 2, emphasis has been placed on the open-circuit solution.

The introduction of a low-loss layer of material into an oilsand filled open-circuited parallel-plate waveguide creates an axial E-field interleaving effect which results in an overall levelling of the electric field magnitudes throughout the saturated region. The low-loss layer also acts to reduce the overall axial wave attenuation of the guide, therefore allowing energy to penetrate deeper to more effectively heat a greater volume of oilsand than would otherwise be possible. Figure 12 shows the squared electric field variations as a function of the axial distance inside a parallel-plate waveguide with parameters: a = 15m, d = 0.85a, $\sigma_1 = 10^{-3}$ S/m, $\sigma_2 = 10^{-6}$ S/m, $\varepsilon_1 = 11\cdot\varepsilon_0$, $\varepsilon_2 = 3\cdot\varepsilon_0$, $\mu_1 = \mu_2 = \mu_0$, f = 340 kHz and L = 400m. Plot (a) shows the transverse squared electric field magnitude variation, plot (b) gives the axial squared electric field magnitude variation and plot (c) gives the total squared electric field magnitude variation. The squared electric field magnitude in plots (a) and (b) show large variations versus axial distance along the waveguide. The resistive heating which would occur from each component separately would result in very non-uniform axial heating. However, further examination of plots (a) and (b) reveals that the

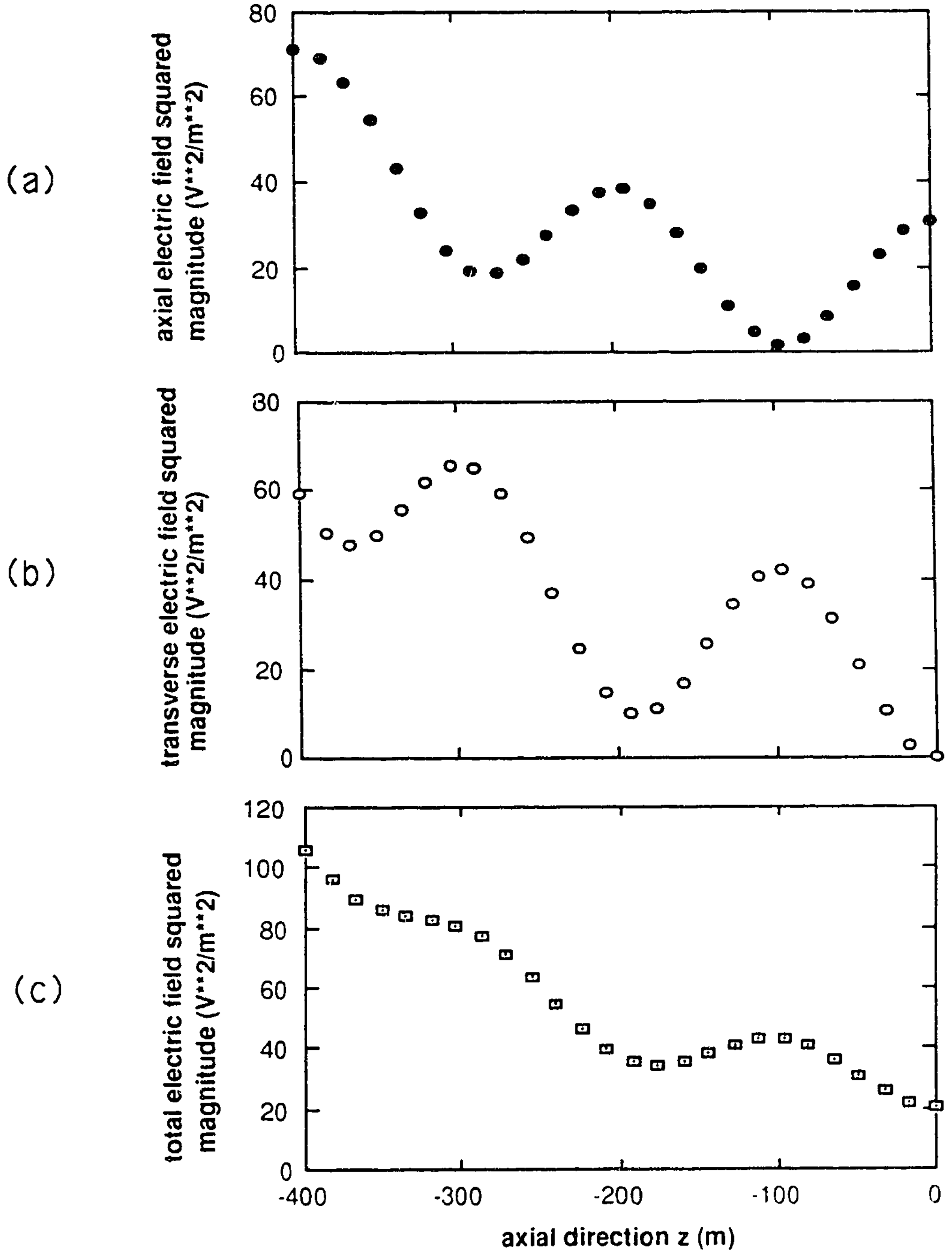

Figure 12 *The axial variation of the squared magnitudes of the transverse (upper graph), axial (middle graph) and total electric fields (lower graph) as measured at x = 13.2m for a short-circuited waveguide with plate separation = 15m, saturated region thickness = 13.5m, saturated region conductivity = 0.001 S/m, L = 400m, f = 150 kHz and all remaining medium electrical properties as given in Table 1*

axial and transverse squared field components have their maximum values at alternate points along the waveguide. They are out of phase spatially by $\lambda/4$. Plot (c) shows the squared electric field resulting from a superposition of the two standing wave squared component fields from plot (a) and (b). The overall axial field variation is levelled out. If the axial and transverse standing wave electric fields are of comparable magnitude, then an optimal interleaving will occur which will lead to an optimally uniform axial heating pattern.

The electric field components inside the waveguide also interleave in the transverse coordinate direction. However, because the plate separation is generally much smaller than a wavelength, the interleaving is generally not apparent. Figure 13 shows the transverse variation of the field components in a waveguide with properties: a=15m, d=0.9a,$\sigma_1=10^{-4}$ S/m, $\sigma_2 = 10^{-6}$ S/m, $\varepsilon_1 = 11 \cdot \varepsilon_0$, $\varepsilon_2 = 3 \cdot \varepsilon_0$ and $\mu_1 = \mu_2 = \mu_0$. In the saturated region, the axial electric field has a value of zero at the bottom plate (x=0) in order to match the perfect conductor boundary condition there. (see eqn. [19]) It then increases as a complex sine function to a maximum value at the media interface. On the other hand, the transverse electric field exhibits a maximum value at the bottom plate and decreases as a complex cosine function until it reaches its minimum value at the media interface. The combined effect of both of these components *generally* results in a maximum value of electric field ratio $|E_z|/|E_x|$ near the media interface, gradually decreasing to a minimum value at the bottom plate. In the depleted region, both electric field components experience a similar axial and transverse variation. However, the transverse field component generally has a much greater magnitude than the axial field and the effect is

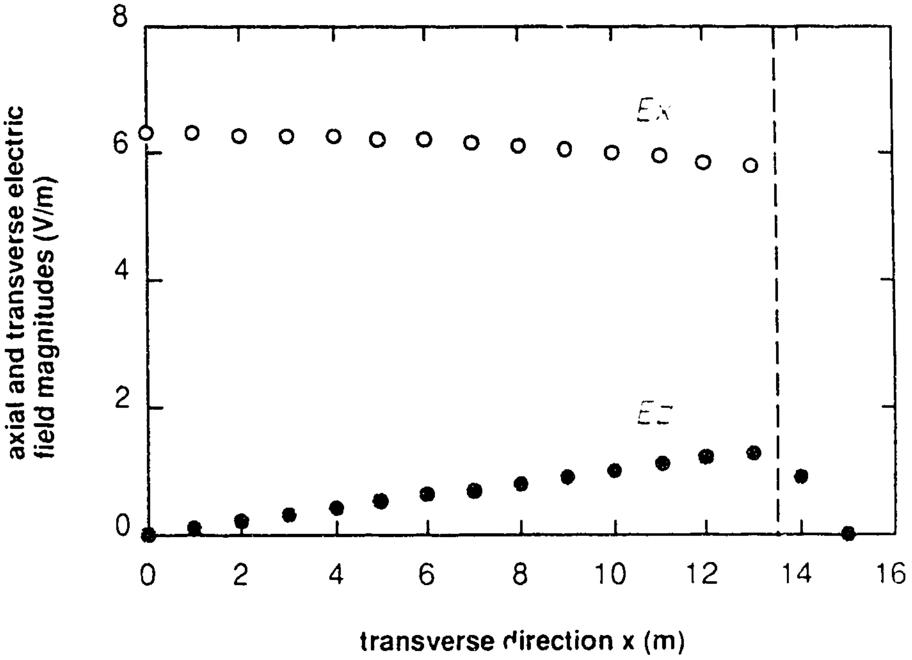

Figure 13 Axial and transverse electric field variations along the transverse coordinate direction in a parallel-plate waveguide filled with a layered lossy medium. Plate separation = 15m, saturated region thickness (snown as a dashed line) = [illegible] m, and saturated region electrical conductivity = 0.0001 S/m. All remaining medium parameters are as given in Table 1. Note Magnitude of Ex in the depleted region is approximately [illegible]

masked. In Figure 13, the transverse field component magnitude in the depleted region is approximately constant at 23 V/m.

The data for Figures 12 and 13 was generated using a program called 'StandingE.' The values of the transverse and axial electric fields were calculated over a user specified grid of the x-z plane in an open-circuited waveguide. Program input consisted of the physical system parameters (as outlined in Table 1) and the number and spacing of points in a rectangular grid. Complete program listings and operating information is given in Appendix 3.

4.3 Evaluation of the Time-Domain Standing Wave Electric Fields

To further understand the behaviour of the fields in the waveguide, the form of the lines of current for the standing wave case were investigated. However, because of the lossy nature of the waveguide media. pure standing waves did not occur. Instead, a mixture of standing and travelling waves resulted. Thus, the lines of current within the waveguide were not simple in form as for the case of the forward travelling wave shown in Figure 11. They were a function of axial position, z, and time.

In order to determine the lines of current within the waveguide, the time-domain expressions of the standing wave electric fields had to be evaluated. A program called 'TimeStandingE' was developed for this. The program calculates the time-dependent axial and transverse electric field magnitudes over a specified grid density of the waveguide x-z plane. This data was then displayed using the graphics program Efields explained earlier. Field line plots were generated for 24 equal time increments spanning one complete time cycle and strung together using a special program called VideoWorks[11] to create an animation of the electric field behaviour inside the waveguide versus time.

Observation of the electric field animation revealed several important characteristics of the fields. Figure 14 is a 'snapshot' of the electric field lines at t=0.83 μsec. inside a waveguide with properties: a = 15m, d = 0.9a, $\sigma_1 = 10^{-3}$ S/m, $\sigma_2 = 10^{-6}$ S/m, $\varepsilon_1 = 11 \cdot \varepsilon_0$, $\varepsilon_2 = 3 \cdot \varepsilon_0$, $\mu_1 = \mu_2 = \mu_0$, f = 150 kHz and L = 400m. It does not show any of the time dependent behaviour but it does give some indication of the forms of the fields. In the low-loss depleted region, the current is very nearly TEM. In the high-loss saturated region, current does not flow in a simple way from the media interface to the bottom plate. Currents also appear to flow from points on the bottom plate, through the saturated medium, to other points on the same plate. The overall field behaviour as a function of time is a complex mixture of standing and travelling waves.

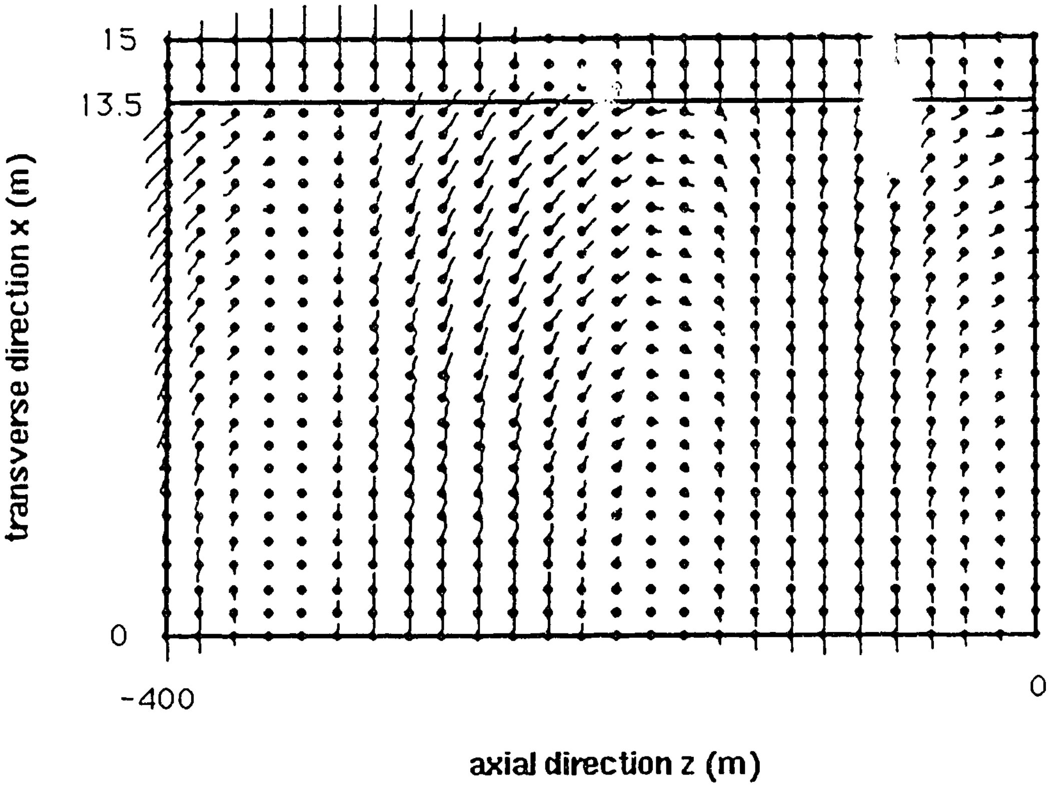

Figure 14. *A picture of the forward wave electric field lines in a short-circuited parallel-plate waveguide with parameters: a=15m, d=0.9a, saturated region conductivity=0.001S/m, L=400m, f=150kHz and time=0.83 microsec. In order to view the electric fields in both regions, the electric fields of the depleted region have been scaled to 1/30 of their original value.*

5 Resistive Heating Within the Waveguide

5.1 General

The main objective of this research is to explore the nature of the distribution of power within the waveguide structure and to show that radio frequency power can be used to heat reasonably large volumes of oilsand with satisfactory uniformity. However, before discussion of the actual waveguide heating, a short, but necessary, explanation of the various methods of power calculation is presented.

Several methods exist for calculating the power dissipated within the waveguide structure. The simplest of methods calculate the power lost per-unit-length for forward travelling waves and only require the axial attenuation constant, α. In their region of validity, both the transmission line equivalent circuit model and the small argument model accurately predict α, and thus the power loss per-unit-length along the waveguide. However, as discussed in Chapter 4, each model is based on very different field forms, and thus predicts different volumetric heating distributions. The TEM nature of the transmission line solution predicts incorrectly uniform transverse power placement. On the other hand, the small argument model, which is based on the analytic field solution, more accurately predicts the transverse power placement.

For a forward travelling wave, the total dissipated power between two axial points along a waveguide can be calculated from the wave attenuation constant, α. If the power flow at the point nearest the source, P_{in}, is known, then the total power dissipated in the waveguide length Δz is given by

$$P_{diss}=P_{in}(1-e^{-2\alpha\Delta z})$$

In its region of validity, the equivalent TEM transmission line circuit shown in Figure 5 can also approximate the power dissipated in the same volume. The resistive power loss in the transmission line circuit occurs in the shunt resistors representing each lossy layer of media. Given the excitation applied to the waveguide, the shunt current through each resistor may be calculated and the dissipated power found through the relation

$$P=|I|^2R$$

These expressions give fast and accurate results for the power dissipated in a particular length of waveguide. Unfortunately, they can only be used for forward travelling waves and cannot predict the volumetric heat distribution in the transverse plane. Most of the research undertaken in this project is concerned with standing wave fields and the associated axial and transverse power distribution and the above methods are, therefore, of limited usefulness.

More useful methods exist for calculation of the power transferred to the waveguide media. Unlike the previous methods, they are able to calculate the power distribution for forward or standing waves as a function of both the x (transverse) and z (axial) coordinates.

The first method uses Poynting's vector which describes the power flow density through a plane.

$$\mathbf{P}=\frac{1}{2}\mathrm{Real}(\mathbf{E}\times\mathbf{H}^*)$$

If the electromagnetic field expressions are known, the total dissipated power in a given volume can be calculated by integrating Poynting's vector over the entire surface bounding the volume. The difference between inflowing and outflowing power, as calculated by the integration procedure, results in the total dissipated power within the volume.

The second method involves direct integration of the dissipated power density over the volume of interest within the waveguide.

$$P = \int_{\text{volume}} \sigma|E|^2 \cdot dV$$

This method for calculating the dissipated power is best suited to a computer solution because it essentially sums the power dissipated at all points inside a volume. It therefore is capable of easily providing information about the power distribution within a volume.

5.2 The Analytic Power Dissipation Expressions

The direct integration method has been chosen to calculate the resistive heating of the waveguide configuration shown in Figure 3. With the aid of a computer, a picture of the power distribution can be created by dividing the waveguide volume into grid blocks and calculating the resistive heating in each one separately. The total power dissipated within a volume of waveguide can then be calculated by simply summing the power dissipated in all the gridblocks within the volume.

The resistive power density is given by the expression

$$\sigma|E|^2$$

For a short-circuit termination, the power density in the lower and upper oilsand regions is given respectively by

$$\sigma_1|E_1|^2=\sigma_1(|E_{x1}|^2+|E_{z1}|^2)=\sigma_1|[-j2C_1^+/(\sigma_1+j\omega\varepsilon_1)]k_z^2\cos(k_{x1}x)\sin(k_zz)|^2+ \\ \sigma_1|[j2C_1^+/(\sigma_1+j\omega\varepsilon_1)]k_{x1}k_z\sin(k_{x1}x)\cos(k_zz)|^2 \qquad [29]$$

and

$$\sigma_2|E_2|^2=\sigma_2(|E_{x2}|^2+|E_{z2}|^2)=\sigma_2|[-j2C_2^+/(\sigma_2+j\omega\varepsilon_2)]k_z^2\cos(k_{x2}(a-x))\sin(k_zz)|^2+$$
$$\sigma_2|[-j2C_2^+/(\sigma_2+j\omega\varepsilon_2)]k_{x2}k_z\sin(k_{x2}(a-x))\cos(k_zz)|^2 \qquad [30]$$

In the case of an open-circuit termination, the expressions have the form

$$\sigma_1|E_1|^2=\sigma_1(|E_{x1}|^2+|E_{z1}|^2)=\sigma_1|[2C_1^+/(\sigma_1+j\omega\varepsilon_1)]k_z^2\cos(k_{x1}x)\cos(k_zz)|^2+$$
$$\sigma_1|[2C_1^+/(\sigma_1+j\omega\varepsilon_1)]k_{x1}k_z\sin(k_{x1}x)\sin(k_zz)|^2 \qquad [31]$$

and

$$\sigma_2|E_2|^2=\sigma_2(|E_{x2}|^2+|E_{z2}|^2)=\sigma_2|[2C_2^+/(\sigma_2+j\omega\varepsilon_2)]k_z^2\cos(k_{x2}(a-x))\cos(k_zz)|^2+$$
$$\sigma_2|[2C_2^+/(\sigma_2+j\omega\varepsilon_2)]k_{x2}k_z\sin(k_{x2}(a-x))\sin(k_zz)|^2 \qquad [32]$$

The heating profiles are similar for the open and short-circuit but are shifted axially with respect to one another by one-quarter wavelength. Because an open-circuit is more easily physically realized and would likely be used in practice, the open-circuited waveguide solution is emphasized in the following discussion.

5.3 Justification for High-Frequency Heating

Intuitively one might expect that a good method for uniformly heating a large oilsand formation would be to excite a set of electrodes, as shown in Figure 1, with low-frequency (60 Hz) power. This would create very uniform electric fields throughout the bulk of the formation and, thus, give very uniform power dissipation. Figure 15 clearly indicates the degree of uniformity of heating achievable in such a configuration. It shows the relative power dissipation through a profile of a homogeneously-filled

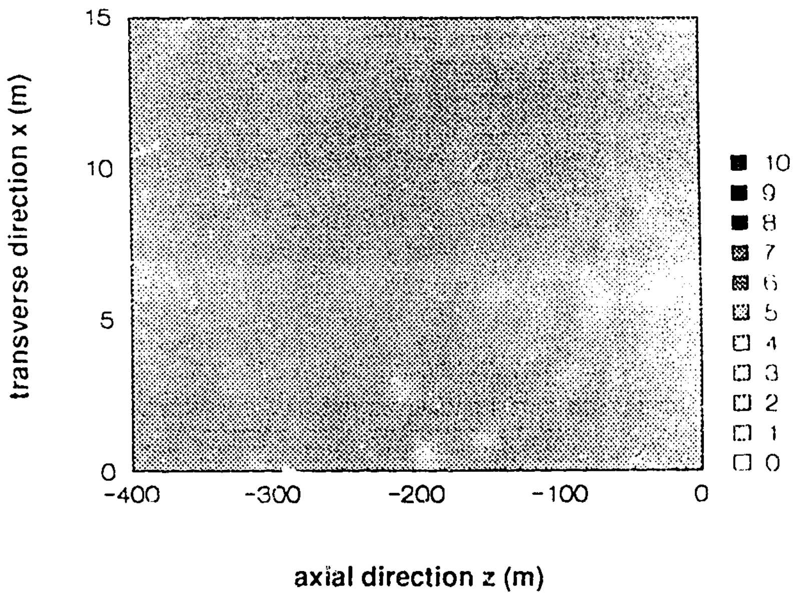

Figure 15. *Heating distribution in Watts per cubic metre inside an oilsand-filled open-circuited parallel-plate waveguide excited at 60Hz.*

parallel-plate waveguide with physical properties: plate separation a=15m. oilsand electrical conductivity $\sigma=10^{-3}$S/m, oilsand dielectric constant $\varepsilon=11\cdot\varepsilon_0$. oilsand magnetic permeability $\mu=\mu_0$, waveguide length L=400m, terminated with an open-circuit and excited with 1kV between the plates at 60Hz. The data for Figure 15 was generated using a program called 'Heater.' Complete information pertaining to and a listing of the program are given in Appendix 3.

Although the bulk of the formation would experience relatively uniform current density as Figure 15 shows, in actuality the regions near the cylindrical electrodes would experience much higher than average current densities. These would tend to rapidly evaporate the current-carrying moisture in these regions, effectively decoupling the electrodes from the formation and halting any further heating.

In order to re-establish the heating rates experienced before the creation of the dried out region, prohibitively high excitation voltages would have to be applied. For instance, consider the waveguide configuration described in Figure 3, excited with 1kV across the plates at 60Hz. The creation of a layer of moisture-depleted oilsand ($\sigma=10^{-6}$S/m, $\varepsilon=3\cdot\varepsilon_0$) near the top electrode only 10cm thick would require the excitation voltage to be approximately 8kV to resume heating of the moisture-saturated region at the previous rate. Almost all of the input power would be dissipated in the 10cm 'gap.' So, for any reasonable applied voltage, insignificant (albeit uniform) heating would occur in the moist oilsand region. Figure 16 shows the resistive power dissipation for the waveguide configuration of Figure 15, but with the 10cm moisture-free gap region introduced near the top plate. The dark regions near the top

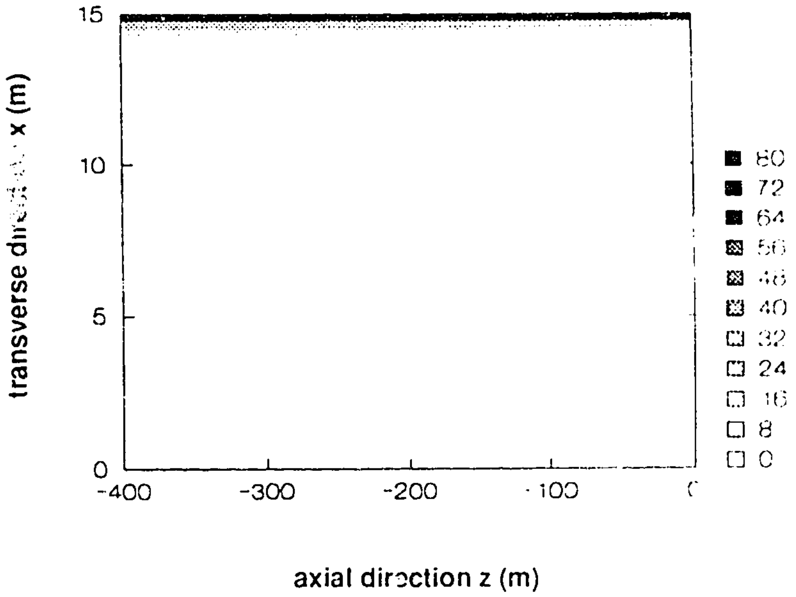

Figure 16. *Heating distribution (in Watts per cubic metre) inside an open-circuited oilsand-filled parallel-plate waveguide at 60Hz after the introduction of a 10cm layer of dried oilsand adjacent to the top plate*

plate signify intense heating in the gap region while heating of the water-saturated oilsand region has effectively ceased. (Note: Although the gap region is only 10cm thick, the finite grid spacing in the calculation and graphing routines has shown the intense heating to be over a larger thickness.)

The low frequency method does give the best heating uniformity. However, for practical applications, high current densities near the electrodes generally create non-conducting gap regions around each electrode. These gaps decouple the formation and make it very difficult to significantly raise the temperature of a formation in a reasonable time. However, if the frequency of operation were to be increased sufficiently, then current could be capacitively coupled across the gap, and heating of the moist oilsand could be resumed.

Consider exciting the system described above at higher frequencies. Initially, before any gap has been created by moisture evaporation, the fields decay very rapidly in the axial direction as induced eddy currents act to shield the bulk of the material between the plates from the applied field. Figure 17 shows the axial variation of the electric field and the corresponding resistive heating for an oilsand-filled parallel-plate waveguide with physical properties: plate separation a=15m, oilsand conductivity $\sigma=10^{-3}$S/m, oilsand dielectric constant $\varepsilon=11\cdot\varepsilon_0$, magnetic permeability $\mu=\mu_0$, waveguide length L=200m, terminated with an open-circuit and excited with 1kV between the plates at 200kHz. (A waveguide length of 200m is used, instead of the 400m length used in other graphs, to more effectively illustrate the wave decay effect.) As can be seen, the fields decay very rapidly in the axial direction and a very small volume of the formation is effectively heated even though it is now possible to continue

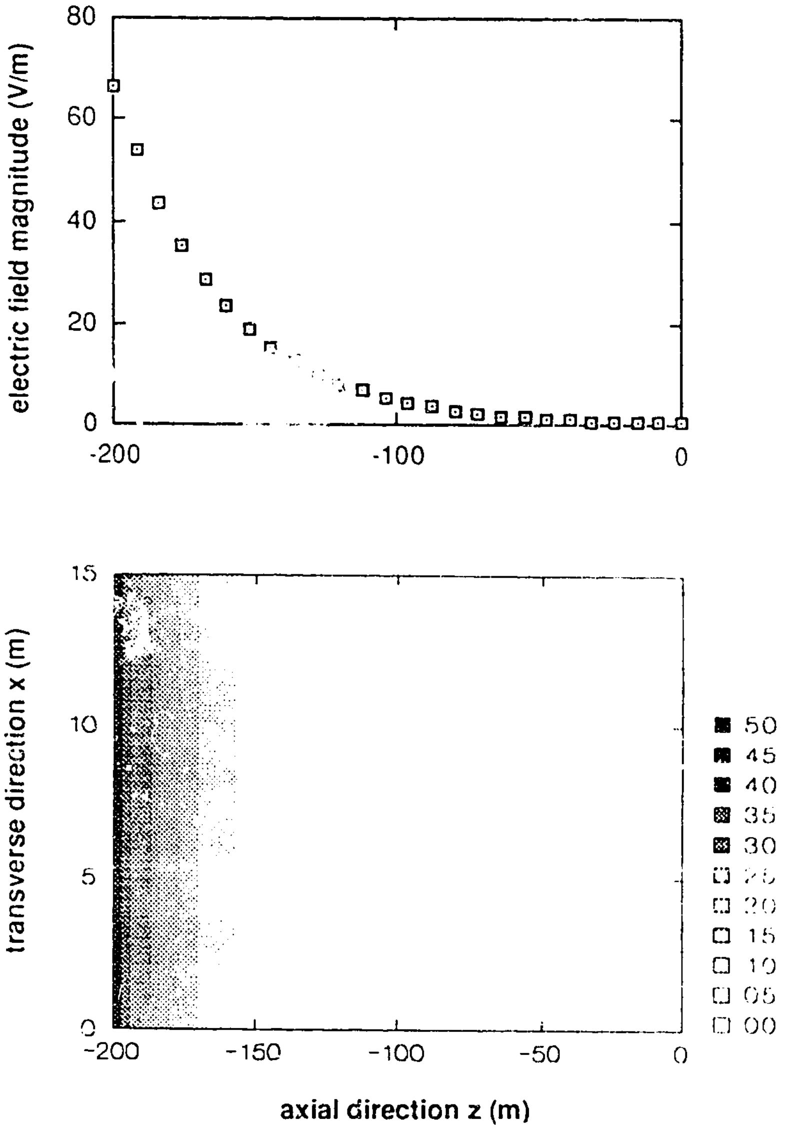

Figure 17. *Axial variation of electric field (upper graph) and heating distribution in Watts per cubic metre (lower graph) inside an oilsand-filled open-circuited parallel-plate waveguide excited at 200kHz.*

continue heating after the formation of a dried-out zone.

However, once a dried out gap region has formed around each buried electrode, the guided wavelength is increased and the axial wave attenuation is sharply reduced. Figure 18 shows the axial variation of the electric fields and the corresponding resistive heating for the same waveguide system as that of Figure 17, but with a moisture-free oilsand gap of thickness=1.5m introduced adjacent to the top plate. It is evident that the creation of the dried out gap has significantly increased the depth of penetration of the signal and much larger volumes of oilsand can be effectively heated.

Thus, operation at high frequencies has permitted heating to be continued after the formation of a dried region. The changes in effective wavelength and depth-of-penetration associated with the presence of the gap then result in extensive heating of the formation between the plates. The following sections will examine the nature of this heating more closely. It should be noted again that it may be advantageous to design a system with a dielectric insulating region around the electrodes in order to take advantage of the improved heating, without having to rely on the self-generation of a dried-out zone to provide the appropriate increase in wave penetration.

5.4 Discussion of the Electric Field and Heating Behaviour

The standing wave field expressions for both the open and short-circuit termination are given as equations [19]-[22]. Examination of these equations reveals that the electric field components vary axially and transversely in a sinusoidal or cosinusoidal fashion. Figures 12 and 13

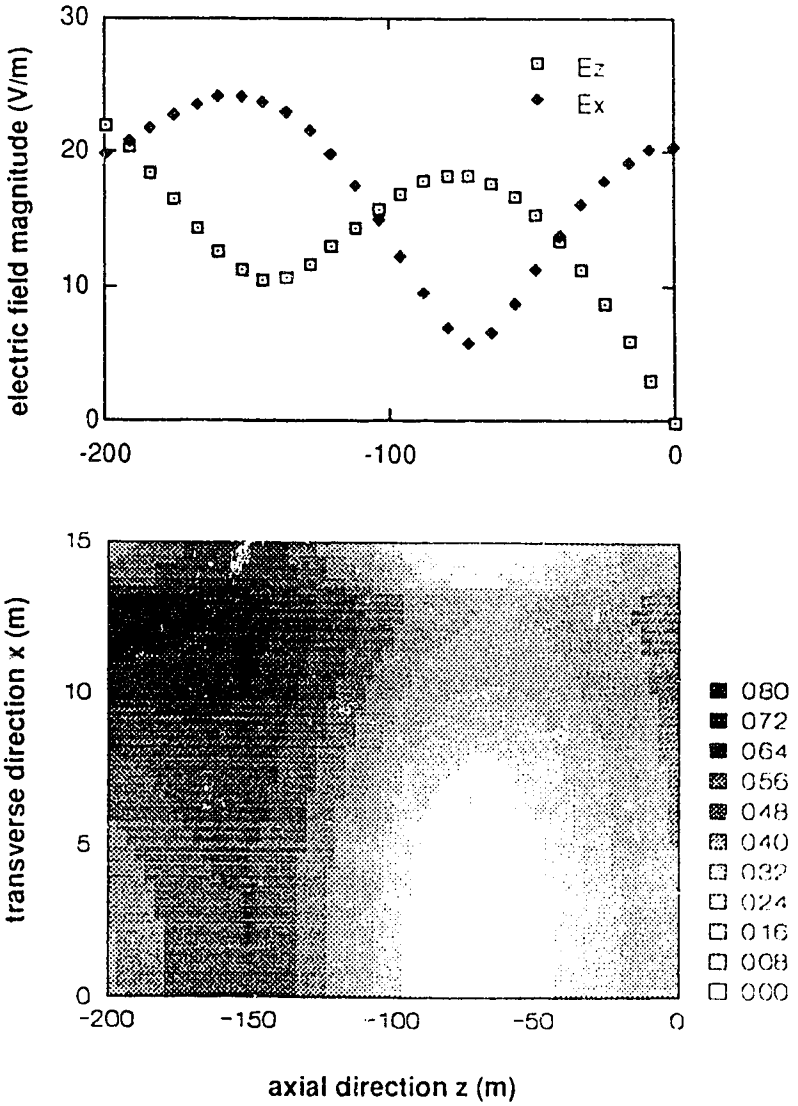

Figure 18. *Axial variation of electric field (upper graph) and heating distribution in Watts per cubic metre (lower graph) inside an open-circuited, oilsand-filled, parallel-plate waveguide at 200Khz after the introduction of a 1.5m layer of dried oilsand adjacent to the top plate.*

show typical behaviour of the electric field components along each coordinate direction. Figure 13 shows typical transverse electric field variation, which is identical for both travelling and standing waves. The variation of E_X is cosinusoidal with a maximum value at each parallel-plate and the variation of E_Z is sinusoidal with a value of zero at each plate. Figure 12 shows a typical axial standing wave occurring in the waveguide. It consists of interleaved E_X and E_Z standing waves positioned a quarter wavelength out of phase with each other. Uniformity of heating along the waveguide axis depends on the relative peak magnitudes of these waves. If the peak magnitude of one component is much stronger than the other, then a non-uniform axial heating pattern occurs. If both peaks are relatively equal in magnitude, then the superposition of both standing waves tends to level out the overall electric field as the bottom graph in Figure 12 illustrates. This leads to more uniform axial heating.

If only the forward wave component of the standing wave is examined (eqns. [15]-[16]), it can be seen that both electric field components vary axially in an identical manner. Examination of the forward and standing wave field expressions reveals that the ratio of the forward wave field component magnitudes at x=d, $|E_z|/|E_x|=|[k_{x1}/k_z]\tan k_{x1}d|$, is the same as the ratio of the peak magnitudes for the standing wave field components if the axial decay, which is common for both components, is momentarily neglected. A ratio of forward wave field components $|E_z|/|E_x| = 1$ signifies equal standing wave peak component magnitudes. A ratio of $|E_z|/|E_x| > 1$ corresponds to a standing wave where the axial electric field component peak magnitude is greater than the transverse electric field peak magnitude and vice-versa.

All of the waveguide configurations studied, for both travelling and standing waves, have revealed a special feature of the actual transverse field variations in the saturated region. In the frequency range of interest, their sine or cosine variation is generally smaller than a quarter-cycle, as exemplified in Figure 13. In the saturated region, the ratio of the field component magnitudes ($|E_z|/|E_x|$) experiences a maximum value at the media interface (x=d) and monotonically decreases to a value of zero at the lower plate (x=0). A ratio $|E_z|/|E_x| = 1$ in the saturated medium near the interface signifies uniform axial heating near the interface. The trend of $|E_z|/|E_x|$ to decrease towards the bottom conducting plate means a progressively weaker axial electric field, and consequently less uniform heating of the saturated region towards the lower electrode. $|E_z|/|E_x| > 1$ near the interface results in non-uniform axial heating near the interface where the axial field dominates. Further down, where $|E_z|/|E_x|$ approaches unity, a region of uniform axial heating occurs. Near the lower conducting plate, the transverse field dominates and the axial heating again becomes non-uniform. $|E_z|/|E_x|<1$ near the media interface results in non-uniform heating throughout the saturated region because the axial and transverse standing wave peaks are never comparable in magnitude.

To determine some basic conditions for which axially uniform heating could occur in the saturated medium of the waveguide, $|E_z|/|E_x|$ (as measured in the saturated region at x=d) was plotted as a function of several variables such as the plate separation 'a', saturated region thickness 'd' and the operating frequency 'f.' Typical saturated zone parameters were chosen ($\sigma_1=10^{-3}$S/m, $\varepsilon_1=11\cdot\varepsilon_0$, $\mu_1=\mu_0$), as were typical gap region parameters ($\sigma_2=10^{-6}$S/m, $\varepsilon_2=3\cdot\varepsilon_0$, $\mu_2=\mu_0$).

Figure 19 shows plots of $|E_z|/|E_x|$ vs. f while varying the plate

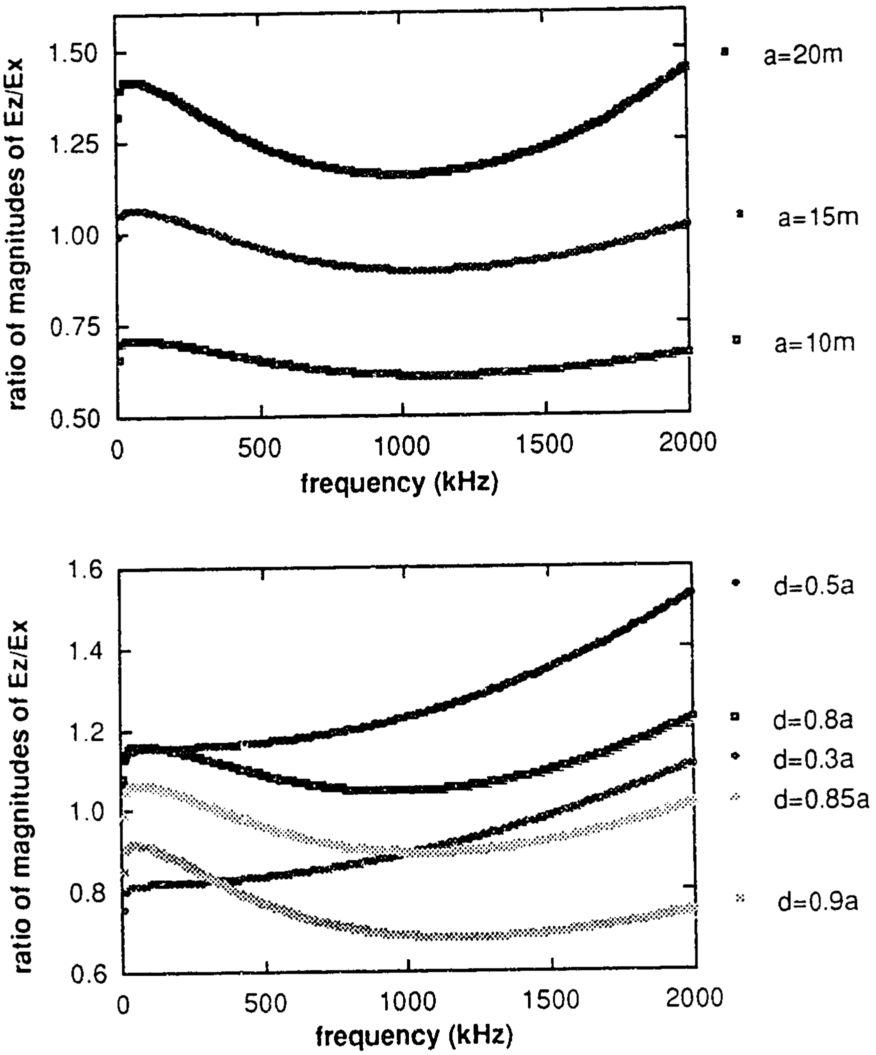

Figure 19. The magnitude of Ez/Ex for a forward travelling wave, measured at the medium interface in the saturated region, versus frequency for a parallel-plate waveguide with saturated region conductivity = 0.001S/m and the remaining medium electrical parameters as given in Table 1. The upper graph shows a family of curves for values of plate separation = 10, 15 and 20m and a constant saturated region thickness = 85% of the plate separation. The lower graph shows a family of curves for values of saturated region thickness = 30, 50, 80 and 85% of the plate separation for a constant plate separation of 15m.

separation 'a' and the saturated region thickness 'd.' The data for Figure 19 was generated using a Fortran program called ERatioMuller which is listed in Appendix 3. The upper graph in Figure 19 shows the behaviour of $|E_z|/|E_x|$ vs. f for plate separations equal to 10, 15 and 20m with the saturated region thickness d = 85% of the plate separation. The lower graph in Figure 19 shows $|E_z|/|E_x|$ vs. f for saturated region thicknesses d=30, 50, 80, 85 and 90% of the plate separation a=15m.

In order to show that relatively uniform heating can be achieved at radio frequencies, one particular system configuration will be chosen and its electric field forms and heating will be examined as a function of frequency. In addition, by plotting the heating as a function of frequency, other factors which affect heating uniformity and efficiency can be identified. Examination of the plots in Figure 19 reveal that a parallel-plate waveguide with properties, a=15m, d=0.85a, $\sigma_1=10^{-3}$S/m, $\sigma_2=10^{-6}$S/m, $\varepsilon_1=11\cdot\varepsilon_0$, $\varepsilon_2=3\cdot\varepsilon_0$, and $\mu_1=\mu_2=\mu_0$ will exhibit a ratio $|E_z|/|E_x|\approx1$ at the media interface over the entire frequency range of interest (except near f=0 where $|E_z|/|E_x|\rightarrow0$). This waveguide configuration, terminated with an open-circuit and excited with 1kV between the parallel-plates at various frequencies, will be analyzed in the remainder of this section.

We begin by examining the waveguide configuration at the power frequency of 60Hz. Figure 20 shows the transverse and axial field variations along with the resistive power dissipation for this waveguide configuration. The middle graph of this figure and all following figures plots the transverse variation of the fields for the forward travelling case and only shows the relative field magnitudes. The axial field variation and heating plots of this figure and all following figures do, however, present

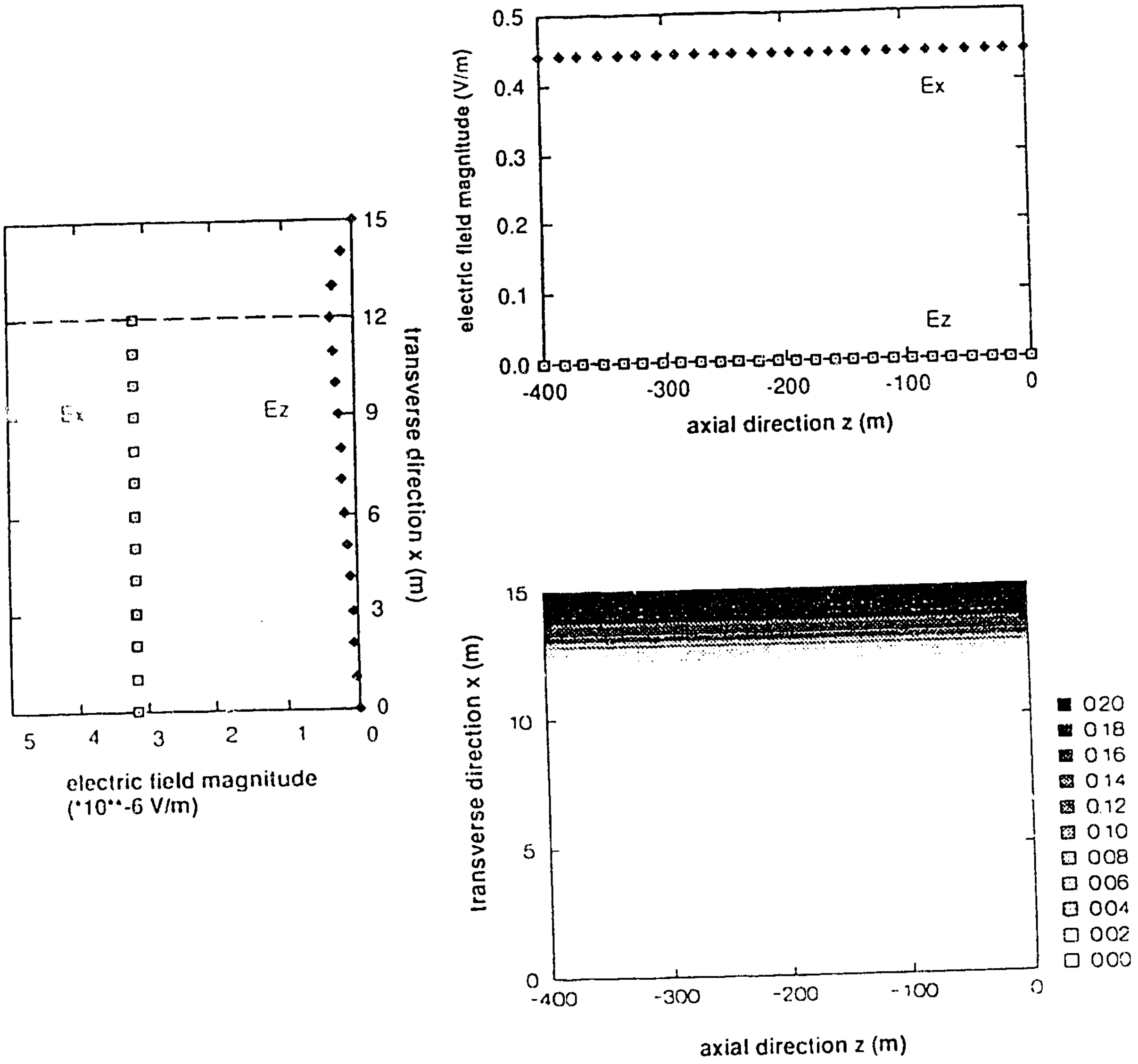

Figure 20. The transverse variation of the electric field (middle graph), the axial variation of the electric field measured at x=12.6m (upper graph) and the corresponding resistive heating pattern in Watts per cubic metre (lower graph) for the parallel-plate waveguide configuration excited at 60Hz. Note: Ex in the gap region (middle graph) is 0.003 V/m.

actual numerical values that correspond to the aforementioned 1kV excitation between the plates.

The middle graph of Figure 20 reveals that the field variation is very TEM-like. In the saturated region, the transverse electric field magnitude is relatively large while the axial electric field magnitude is almost zero. In the gap region, this effect is magnified. The value of the transverse electric field cc nponent, which is too large to be included in the plot, has an average magnitude of approximately 0.003 V/m or approximately 1000 times the magnitude of the transverse field in the saturated region.

The TEM-like behaviour is also reflected in the axial electric field variation in the saturated region, as measured near the gap interface and shown in the upper graph of Figure 20. Because the wavelength is very large at this frequency, little axial variation is evident. The transverse electric field exhibits a finite non-zero value of about 0.44 V/m which remains relatively constant along the waveguide length while the axial electric field component is near zero everywhere.

The resistive power dissipation is depicted in the lower graph of Figure 20. At 60Hz, effectively all of the heating occurs in the gap region. No heating occurs in the saturated region because the dried out gap layer has decoupled the moisture-saturated region from the electrodes. If the waveguide is modelled by the TEM equivalent circuit of Figure 5, then it can be said that the impedance of the resistor/capacitor combination representing the gap region is much larger than that of the saturated region. Thus, most of the applied voltage is dropped across the gap. Also, the capacitive impedance of the gap region is much larger than its resistive impedance. Most of the current flows through the resistor and results in high power dissipation. At low frequency, the intended or accidental introduction of a gap region near the electrode terminates the heating of the saturated region.

Figure 21 shows the transverse and axial field variation as well as the power dissipation for the waveguide excited at 10kHz. The electric field variation along the transverse direction is shown in the middle graph of Figure 21. It reveals that the ratio of the electric field magnitudes $|E_z|/|E_x|$ is approximately one near the gap interface in the saturated region. Thus, the necessary criterion for possible uniform heating of the saturated region is satisfied. The form of the transverse variation of the fields can be explained by applying the small argument condition to the forward wave field expressions (eqns. [15]-[16]). The small argument condition states that a complex trigonometric function can be replaced by its argument if the magnitude of the argument's real and imaginary components does not exceed 0.3. For this waveguide configuration the value $k_{x1}d=0.008-j0.004$ satisfies the small argument condition. When subjected to the small argument, the transverse variation of the transverse electric field component is $\cos k_{x1}x \approx 1$, and the transverse variation of the axial electric field is $\sin k_{x1}x \approx k_{x1}x$. This behaviour is clearly seen in the middle graph of Figure 21.

The axial behaviour of the electric fields of Figure 21 (upper graph) shows a change from the 60Hz case of Figure 20. The wavelength has decreased from 157790m to 6472m and the axial variation of the field components is now apparent.

The lower graph of Figure 21 shows the resistive power dissipation for the waveguide excited at 10kHz. The saturated region is still almost

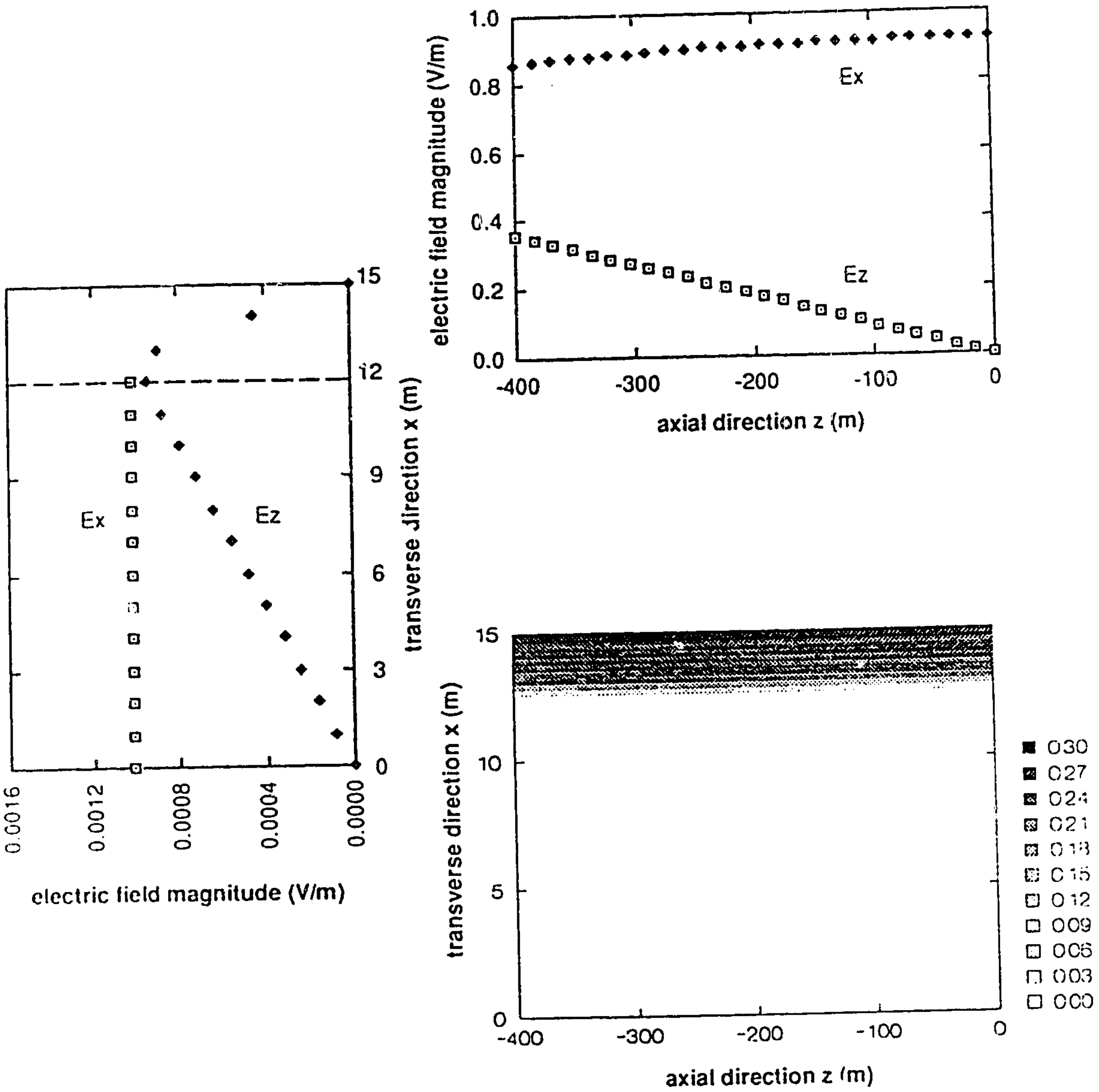

Figure 21. *The transverse variation of the electric field (middle graph), the axial variation of the electric field measured at x=12.6m (upper graph) and the corresponding resistive heating pattern in Watts per cubic metre (lower graph) for the parallel-plate waveguide configuration excited at 10kHz. Note: Ex in the gap region (middle graph) is 0.5 V/m.*

completely decoupled from the electrodes and any heating which may be occurring there is negligible compared to the excessive heating in the gap region.

The field behaviour and heating pattern for the waveguide excited at a higher frequency of 100kHz is shown in Figure 22. The field behaviour along the transverse direction, shown as the middle graph, is similar to that of Figure 21. However, the magnitudes of the electric field components in the saturated region have increased. This increase may be explained in the context of the equivalent circuit of Figure 5. As a result of the increase in frequency, the diminishing capacitive impedance of the gap region is beginning to short-circuit the parallel resistor. This results in less current effectively flowing through the resistor and also a higher proportionate voltage drop across the moisture-saturated region. The higher voltage yields stronger electric fields and, thus, greater power dissipation in the moist region and less in the gap.

The electric field variation along the waveguide axis shows expected differences from the 10kHz case of Figure 21. For instance, a further decrease in wavelength from 6472m to 680m results in the axial interleaving, as measured in the saturated region near the gap, to become visible. The equality of the separate standing wave component reveals that axial heating should be relatively uniform, at least near the gap.

The heating pattern shown in the lower graph of Figure 22 for a frequency of 100kHz shows that heating in the saturated region is beginning to occur. Heating of the gap has decreased significantly, but it is still very strong and overshadows most of the heating occurring the saturated region. The important effect to note is that current is beginning to couple across the gap and cause significant heating in the saturated

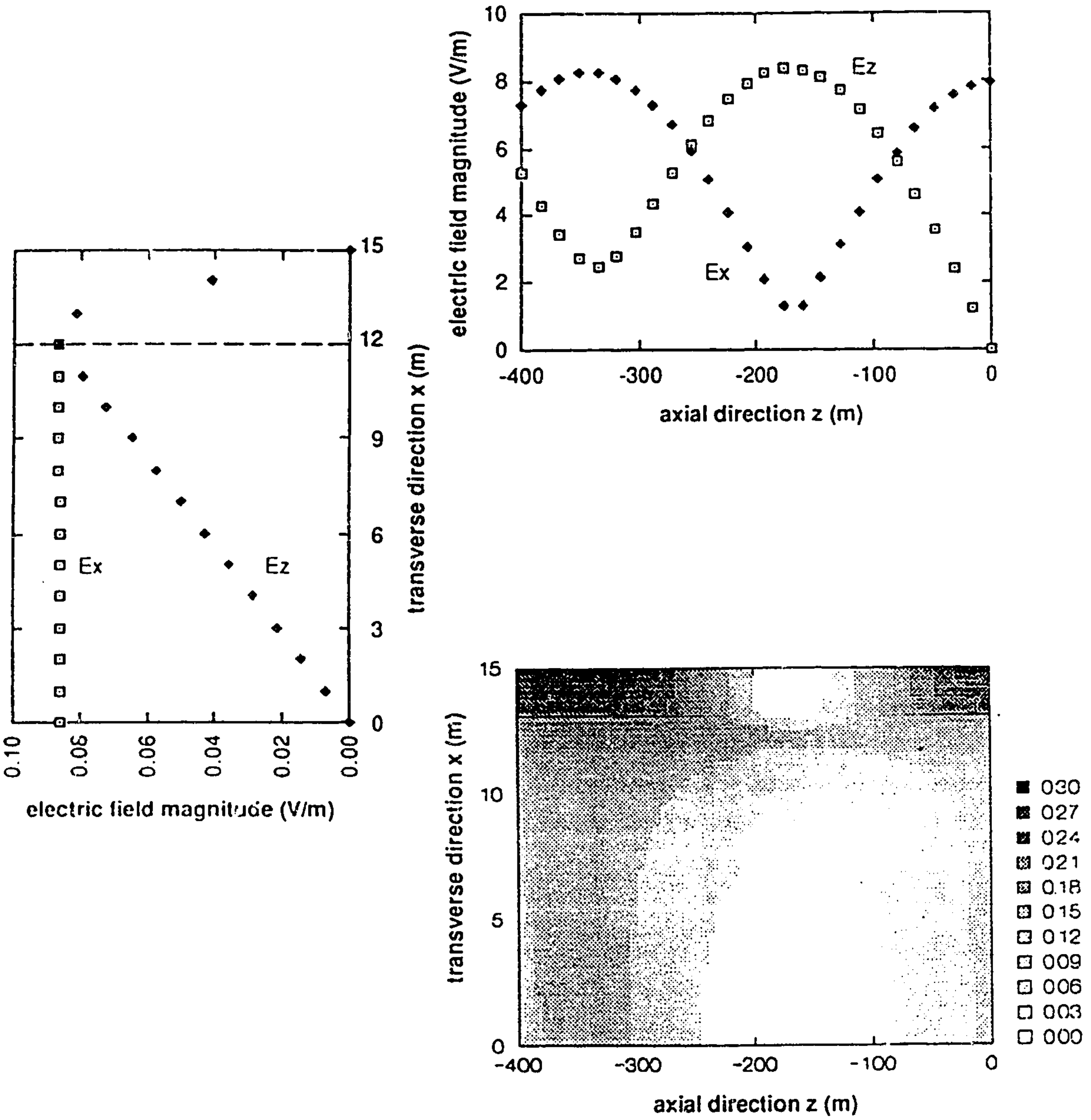

Figure 22. *The transverse variation of the electric field (middle graph), the axial variation of the electric field measured at x=12.6m (upper graph) and the corresponding resistive heating pattern in Watts per cubic metre (lower graph) for the parallel-plate waveguide configuration excited at 100kHz. Note: Ex in the gap region (middle graph) is 5 V/m.*

region Although its uniformity is not as good as could be achieved by heating at 60Hz as shown in Figure 15, it is much more reliable heating since it does not critically depend on moisture near the electrodes for the passage of current through and, therefore, heating of the formation.

The electric field variations and the heating patterns for the same waveguide excited at 160kHz are shown in Figure 23.

The middle plot of Figure 23, showing the transverse variation of the electric field components, has the same form as for the frequencies of 10kHz and 100kHz. As explained previously, the magnitudes of the electric field components have increased with frequency, as expected.

The axial variation of the electric field components is shown in the upper plot of Figure 23. The wavelength has decreased from 680m to 432m as compared to the 100kHz case of Figure 22. In addition, the wave decay effect is becoming noticeable. An examination of the mode plot shown in Figure 10 shows that the axial wave decay increases with frequency for the dominant mode. As the frequency increases, more current is coupled across the gap and more power is dissipated in the saturated region. However, as a consequence, there is an increase in the axial wave decay constant and corresponding reduction in the wave depth-of-penetration.

The lower plot in Figure 23 shows the heating pattern for the waveguide configuration at 160kHz. Heating of the gap region has been reduced significantly. Moreover, the non-uniform heating normally resulting from TEM standing waves in a homogeneous system, has been levelled out over most of the saturated region as a result of the gap. Also, by taking into account the non-zero thermal conductivity of the saturated

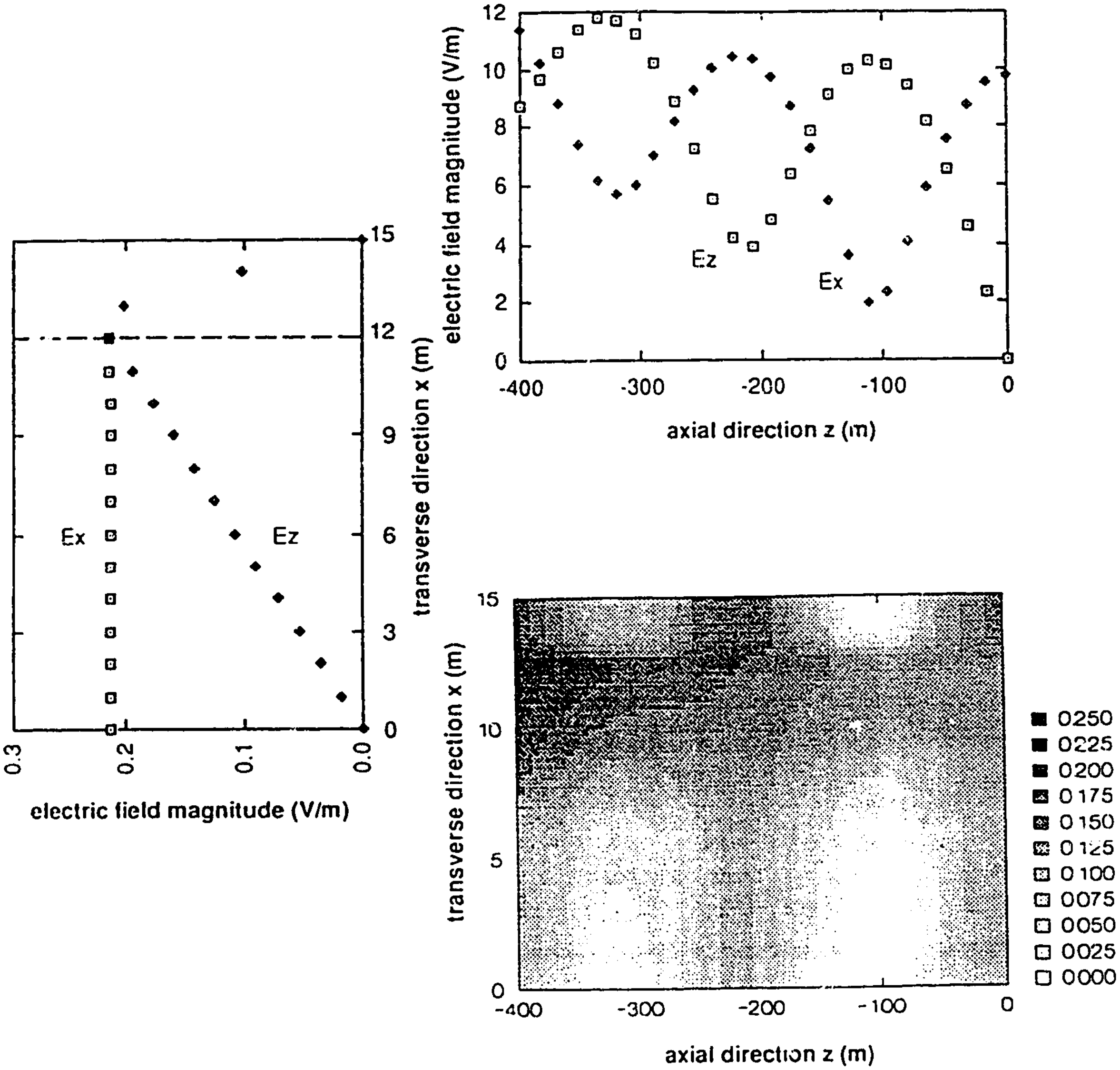

Figure 23. *The transverse variation of the electric field (middle graph), the axial variation of the electric field measured at x=12.6m (upper graph) and the corresponding resistive heating pattern in Watts per cubic metre (lower graph) for the parallel-plate waveguide configuration excited at 160kHz. Note: Ex in the gap region (middle graph) is 5 V/m.*

oilsand, the improved uniformity and depth-of-penetration achieved by the introduction of the gap is further enhanced. The heat generated in warmer regions will have shorter distances to travel than in the TEM case to distribute itself uniformly throughout the formation. This will improve efficiency by reducing the time needed to heat all parts of a formation to a required minimum temperature.

This plot shows the very good uniformity which can be achieved by introducing a low-conductivity region adjacent to the formation electrodes and then operating at radio frequencies. Introduction of the gap also greatly increases the effective depth-of-penetration over what would be achieved if a homogeneous medium were excited at radio frequencies (i.e. see Figure 17).

If the frequency were to be increased even more, more current would be coupled across the gap and less heating would occur there. However, at the same time, heating uniformity in the saturated region would be diminished because of the decreased wave skin-depth.

Figure 24 shows the axial and transverse field variations and the corresponding resistive heating for the waveguide excited at 250kHz. The middle plot shows the familiar transverse variation of the fields and is consistent with the trend shown in previous plots.

The axial variation of the field components (upper graph) shows an intensification of the wave decay effect. The field magnitude at the open-circuit is much lower than at the waveguide input. This excessive decay has partially negated the benefits of having the interleaving of the electric field components along most of the waveguide. The wave decay effect has essentially put an upper bound on the frequencies which can be used to

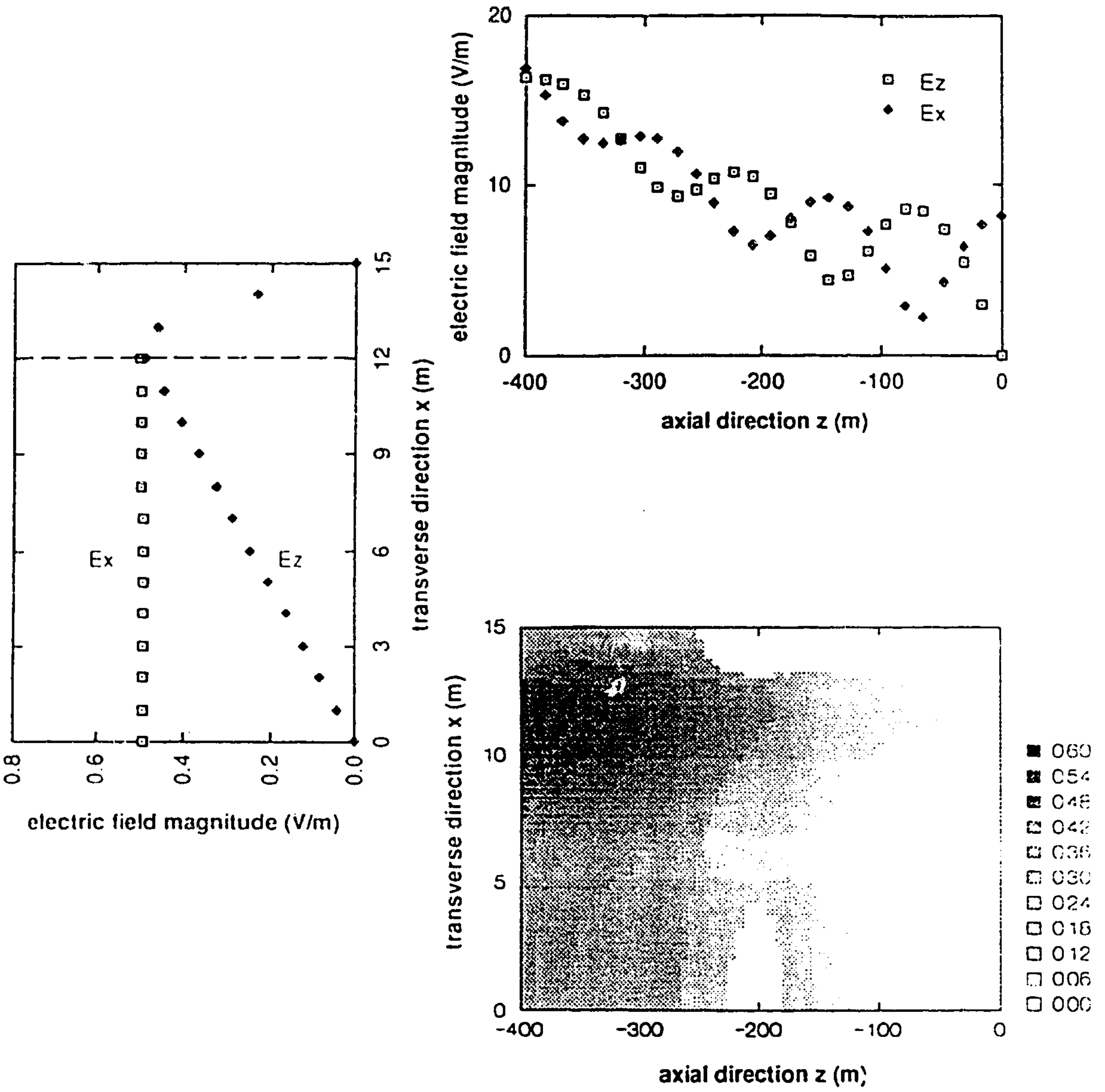

Figure 24 *The transverse variation of the electric field (middle graph), the axial variation of the electric field measured at x=12.6m (upper graph) and the corresponding resistive heating pattern in Watts per cubic metre (lower graph) for the parallel-plate waveguide configuration excited at 250kHz. Note: Ex in the gap region (middle graph) is 12 V/m.*

achieve uniform heating over the indicated volume.

The resistive heating pattern of Figure 24 reflects the behaviour of the axial variation of the fields discussed above. Although less heating occurs in the gap region, better heating uniformity in the saturated region cannot be achieved with this higher frequency because of the significant decrease in the depth-of-penetration.

We have shown that relatively uniform heating of volumes of oilsand with dimensions much larger than the skin-depth of a plane wave in the saturated region can be achieved. By introducing a low-loss region of material adjacent to the formation electrodes and operating at radio frequencies, relatively uniform heating can be obtained without fear of decoupling the formation and terminating the heating process. The previous discussion has also shown some of the concerns that are associated with this heating scheme. By operating at too low a frequency, current will not effectively couple across the gap region. Consequently, the majority of heating will occur in the gap and heating of the saturated region will be minimal. On the other hand, by operating at too high a frequency, skin-depth effects may limit the depth-of-penetration of the electromagnetic energy and reduce the volume of formation being effectively heated. Although these effects are of concern, it has been seen that it is possible to heat large blocks of oilsand relatively uniformly and efficiently.

Depending on which oilsand heating and bitumen recovery scheme is used, the thickness of the gap region may increase as heating proceeds. As the gap region becomes larger, a higher operating frequency will be necessary to ensure coupling of current across the gap and heating of the saturated formation. At the same time, a larger gap region results in a decrease in the wave attenuation constant. This decrease tends to counteract the increase in the attenuation constant attributed to the increase in operating frequency. The overall effect is to shift the frequency range of uniform heating higher. So, if a heating scheme using, say, gravitational drainage were to be used, then the excitation frequency would have to increase with increases in gap thickness to preserve optimal heating. It should be noted, however, that the condition of $|E_z|/|E_x| \geq 1$ in the saturated region x=d still must be satisfied. If not, then heating uniformity will suffer. In addition, operating at too high a frequency may allow higher order modes to propagate, in which case a more complex analysis of the heating would have to be undertaken.

5.5 System Design Using Electromagnetic Scaling

With knowledge of a waveguide configuration which effectively heats a formation of given properties, such as the one shown in Figure 23, other waveguide systems may be designed to heat oilsand formations of different electrical properties (i.e. conductivity) through the implementation of electromagnetic scaling theory. The design procedure basically involves specifying the essential parameters required in the new system, followed by application of electromagnetic scaling theory to the known base case to determine the remaining parameters required for the new system. The design parameters cannot be chosen independently and must be chosen with reference to the scaling equations. For example, the choice of one parameter may result in another parameter becoming fixed by the scaling equations. The following paragraph states the necessary equations for electromagnetic scaling.[12]

The electrical parameters for a full scale system are denoted by σ, ε and μ. For the corresponding scale model, the system parameters are denoted by σ', ε' and μ'. $x=px'$, $y=py'$, and $z=pz'$ relate the full scale and model geometries and the operating frequencies are determined by the relation $\omega'=\gamma\omega$. The saturated regions of the model and full scale system achieve electromagnetic simiItitude if their parameters satisfy the equations

$$p=\frac{\sigma_1'}{\sigma_1}\sqrt{\frac{\varepsilon_2}{\varepsilon_2'}} \quad \text{and} \quad \gamma=\frac{\sigma_1'\varepsilon_2}{\sigma_1\varepsilon_2'}$$

provided that $\sigma_2/\omega\varepsilon_2 \ll 1$ and $\sigma_1/\omega\varepsilon_1 \gg 1$. A complete example of the scaling procedure, as applied to an oilsand filled parallel-plate waveguide is given in Appendix 1.

6. Conclusions and Further Work

Nearly all of the Athabasca oilsand formations in place are deeply buried under 150m or more of overburden and cannot be readily surface-mined. At their average in-situ temperature of 10-15^0 C, the oilsand formations are very viscous and have a low permeability and thermal conductivity. These conditions render it very difficult to recover the bitumen by conventional steam and fluid injection techniques. Prior research has shown that the viscosity of the oil in the oilsand decreases dramatically with moderate temperature increases. Therefore, preheating the oilsand prior to steam or fluid injection recovery has been suggested. Due to the low thermal conductivity and injectivity of oilsand, electrical heating, which does not rely on thermal conduction or diffusion of hot fluids in a medium, has been suggested as a possible heating method.

In general, two methods of electrical preheating have emerged. These are low frequency (60 Hz) and radio-frequency methods. Prior research done in the Applied Electromagnetics Laboratory at the University of Alberta has shown that radio-frequency heating methods can theoretically be applied to heat reasonably large volumes of deeply buried oilsand formations efficiently.

Initial investigations done at the University of Alberta have shown horizontal electrode placement to be preferred. Two rows of electrodes, each straddling the top and bottom of the oilsand payzone, can readily be approximated by a parallel-plate waveguide. Filling the waveguide is a layered lossy medium, the upper layer representing the moisture-depleted oilsand region and the lower layer representing the saturated oilsand region. For waveguides longer than the guided wavelength, standing wave guided wavelength, standing wave effects may create highly non-uniform heating patterns which result in inefficient bitumen recovery. With proper waveguide termination and careful system design, the normally detrimental standing wave effects may be modified to create uniform field strengths and heating rates throughout much of the saturated oilsand volume, and thus maximize the efficiency of the following bitumen recovery stage.

This research began with the derivation of the analytic solution for the electromagnetic fields inside the parallel-plate waveguide. This allowed the calculation of the resistive heating patterns in the waveguide. Evaluation of the resistive heating patterns required numerical solution of a complex-valued transcendental equation describing the wave propagation constants. Two models were developed to approximate the propagation constants and, thus, simplify the solution procedure. A TEM transmission line equivalent circuit describing the parallel-plate waveguide led to a simple expression for the dominant mode propagation constant in terms of various physical system parameters. A second model, based on the analytic field solutions, but constraining the transverse propagation constants k_{x1} and k_{x2} to small values, gave identical expressions for the waveguide propagation constants but made no field assumptions for the waveguide. Better than 90% accuracy was achieved by the approximate expressions if the complex arguments $k_{x1}d$ and $k_{x2}(a\text{-}d)$ appearing in the transcendental equation were limited in magnitude to no more than 0.3 in real and imaginary components. The waveguide models also accurately predicted the total power dissipated within a section of waveguide for forward travelling waves. However, the transmission line model incorrectly predicted uniform transverse heating within the waveguide, whereas the small argument approximation, which is based on the analytic field solutions, predicted the correct power distribution as described by the field expressions themselves.

The majority of the research work done involved detailed investigation of the resistive heating behaviour in the waveguide as governed by the analytic field solutions. Investigation of the resistive heating patterns for a given waveguide configuration as a function of frequency has revealed that relatively uniform power dissipation in the saturated region of the waveguide may be achieved. Obtaining uniform heating, however, requires proper line termination and careful choice of system parameters and frequency. System design can also be carried out if a system configuration exhibiting uniform heating is known. Then systems with desired properties can be designed through application of electromagnetic scaling theory. For instance, given a waveguide configuration exhibiting uniform heating (see Figure 23), a system designed to heat oilsand of different conductivity can be found easily through application of electromagnetic scaling theory.

Proper initial system design is very important. Adjusting individual system parameters such as frequency in efforts to compensate for non-uniform heating can be effective. However, heating uniformity is very dependent on, for instance, saturated region conductivity. Therefore, knowledge of such factors as the temperature dependence of oilsand conductivity is very important in designing a system to optimize the overall preheat efficiency.

As well as providing some answers to questions concerning the heating of oilsand formations, the research presented has given rise to many further questions which merit additional research. The waveguide system investigated has been idealized. In order to form a clearer picture of the power dissipation behaviour in an actual in-situ heating scenario, several aspects of the waveguide excitation must be addressed. Beginning with the waveguide input, the form of the excitation fields play a major role in the heating adjacent to the waveguide input. The type of excitation applied to the waveguide affects which higher order modes are launched in addition to the dominant mode and, thus, consequently determines the evanescent heating pattern near the waveguide input. Determination of this heating is important to the overall understanding of the waveguide heating behaviour and also helps to determine preferred methods of excitation. In addition to using the generation of higher order modes as a criterion for choosing a particular form of waveguide input, the efficiency of power transfer from the source to the waveguide requires examination. In order to maximize the efficiency of the overall system, using a waveguide excitation which is matched to the waveguide is also necessary. Further, given that the waveguide input network has been chosen, questions concerning the actual power levels to be applied and the resultant heating times and temperatures in the saturated oilsand need to be addressed. Design of a system to give the required temperature rise in the oilsand in a reasonable period of time will, in turn, determine the size and type of radio-frequency electrical generators and related hardware required, such as electrode sizes, matching hardware, connecting cable specifications, etc. For proper application of the oilsand preheat methods to practical situations, the above system design parameters must be known and, therefore, merit further research.

Again, it is emphasized that the research done has been for an idealized parallel-plate waveguide geometry. Obtaining heating results for non-ideal systems requires the use of numerical programs. An important step in further research concerning radio-frequency heating of oilsand formations, or any medium that is to be heated, is the development of a numerical program for calculation of the resistive heating within a parallel-plate waveguide loaded with an arbitrarily inhomogeneous medium. Further enhancements to account for such factors as thermal conduction, fluid flow and temperature sensitive electrical parameters should also be incorporated into the program. A basic version of such a program has been developed and is presented in Appendix 2. Comparison of the results from the numerical program to those from the analytic solution reveals excellent agreement between the two. Many more questions require attention. However, the ones mentioned here are of main concern for the eventual successful application of radio-frequency heating to various media.

References

1 Vermeulen, F. E., Chute, F. S., Electromagnetic Techniques in the In-Situ Recovery of Heavy Oils., J. Microwave Power, 18 (1), 1983.

2 Chute, F. S., Vermeulen, F. E., (1985) Electromagnetic Heating of Oil Sands, Final Report: AOSTRA Agreement No. 296, pp. 9-10.

3 Carlson, R. D., Blase, E. F. and McLendon, T. R., Development of the IIT Research Institute RF Heating Process for In Situ Oil Shale/Tar Sand Fuel Extraction - An Overview., 14th. Oil Shale Symposium Proceedings, Colorado School of Mines, 1981, pp. 142-44.

4 Bridges, J. E., Taflove, A. and Snow, R. H., Net Energy Recoveries for the In Situ Dielectric Heating of Oil Shale., Proceedings of the 11th. Oil Shale Symposium, Colorado School of Mines, 1978, p. 327.

5 Ibid.

6 McPherson, R. G., Chute, F. S. and Vermeulen, F. E., The Electromagnetic Flooding Process for In-Situ Recovery of Oil From Athabasca Oil Sand. J. Microwave Power, 1986, pp. 129-47.

7 McPherson, R. G., High Frequency Heating of the Athabasca Oil Sands., Ph. D. Thesis, University of Alberta, 1985, pp. 137-46.

8 McPherson, pp. 149-53.

9 Harrington, R.F., Time Harmonic Electromagnetic Fields, (McGraw Hill, New York, 1961), pp. 158-63.

10 Muller, D., A Method for Solving Algebraic Equations

Using an Automatic Computer, MTAC, 10, 1956, p. 208

11 VideoWorks II. Macromind Inc., 1987

12 Vermeulen, F. E., Chute, F. S. and Cervenan, M. R. Physical Modelling of the Electromagnetic Heating of Oil Sand and Other Earth Type Materials. Can. Elec. Eng. J., 4(4), 19-28, 1979.

Works Consulted

Bridges, J.E., Krstansky, J.J., Taflove, A., Sresty, G., "The IITRI In-situ RF Fuel Recovery Process", Journal of Microwave Power, 18 (1), 1983.

Bridges, J., Sresty, G., Taflove, A., Snow, R., "Radio-Frequency Heating to Recover Oil From Utah Tar Sands", Future of Heavy Crude and Tarsands, 1st. International Conference, Edmonton, Canada, June, 1979.

Burden, R.L., Faires, J.D., Numerical Analysis, 3rd. ed., (Prindle, Weber, and Schmidt, Boston, 1985).

Chute, F.S., Vermeulen, F.E., McPherson, R.G., Hiebert, A., Fearn, J., Electromagnetic Heating of Oil Sands, AOSTRA Research Agreement #296, April 1985, Edmonton, Alberta.

Dahlquist, G. and Bjorck, A., Numerical Methods, translated by Ned Anderson (Prentice Hall, New Jersey, 1974)

Fuller,B., Microwaves, (Permagon Press, Oxford, 1969).

Harrington, R.F., Time Harmonic Electromagnetic Fields,(McGraw Hill, New York, 1961).

Hayt, W.H., Engineering Electromagnetics, 4th. ed., (McGraw Hill, New York, 1981)

Jordan, E.C. and Balmain, K.G., Electromagnetic Waves and Radiating Systems (Prentice Hall, New Jersey, 1968)

Kong, J.A., Theory of Electromagnetic Waves (John Wiley and Sons, New York, 1975)

Kurokawa, K., An Introduction to the Theory of Microwave Circuits, (Academic Press, New York, 1969).

Liao, S.Y.,Microwave Devices and Circuits, 2nd. ed., (Prentice

Hall, New Jersey, 1983)

Liboff, R. and Dalman, G., Transmission Lines, Waveguides and Smith Charts (MacMillan, New York, 1985)

Liu M., Rosenbaum, F.J., Pickard, W.F., "Electric Field Distribution Along Finite Length Lossy Dielectric Slabs in Waveguide", IEEE Transactions on Microwave Theory and Techniques, April 1976.

McPherson, R.G., "High Frequency Electric Heating of the Athabasca Oil Sands", Ph.D. Thesis, University of Alberta, 1985.

Metaxas, A.C. and Meredith, R.J., Industrial Microwave Heating (Peter Peregrinus, London, 1983)

Mrozowski, M., Mazur, J., "General Analysis of a Parallel-Plate Waveguide Inhomogeneously Filled with Gyromagnetic Media", IEEE Transactions on Microwave Theory and Techniques, MTT-34, #4, April 1986.

Sresty, G.C., Dev, H., Snow, R.H., Bridges, J.E., "Recovery of Bitumen from Tar Sand Deposits with the Radio Frequency Process", SPE Reservoir Engineering, January, 1986.

Sresty, G.C., Snow, R.H., Bridges, J.E., "The IITRI RF Process to Recover Bitumen from Tar Sand Deposits-A Progress Report",Future of Heavy Crude and Tarsands, 2nd International Conference, Venezuela, February, 1982.

Steele, C.W., Numerical Computation of Electric and Magnetic Fields (Van Nostrand Reinhold Company, New York, 1987)

Zahn, M., Electromagnetic Field Theory (John Wiley and Sons, New York, 1979)

Appendix 1.

Electromagnetic Scaling Example

This appendix gives a detailed example of the electromagnetic scaling procedure used for the design of parallel-plate waveguide systems which will exhibit uniform heating.

Consider an inhomogeneously loaded parallel-plate waveguide configuration akin to the one shown in Figure 1 in Chapter 1. The electrical conductivity of region 1, the saturated oilsand region, is dependent on the grade of oilsand being heated. Having obtained a set of system conditions required to achieve uniform heating in a saturated oilsand layer of particular conductivity, a method of choosing system parameters to achieve uniform heating for a saturated region of different conductivity is desirable. One available method is to use electromagnetic scaling theory to quantify the new system.[12]

For the problem considered here, only two scaling equations are necessary.

$$\frac{x}{x'} = p = \frac{\sigma_1'}{\sigma_1}\sqrt{\frac{\varepsilon_2}{\varepsilon_2'}}$$

where the primed quantity denotes the model and the unprimed quantity denotes the full scale system. p is the mechanical scale factor relating the x,y and z dimensions of the model and full scale system.

The second equation is

$$\gamma = \frac{\sigma_1'\varepsilon_2}{\sigma_1\varepsilon_2'} = \frac{\omega'}{\omega}$$

where ω is the radian frequency and γ is termed the frequency scaling factor.

For illustrative purposes, we may choose a system configuration with the following properties:

$\sigma_1=10^{-3}$ S/m	$\sigma_2=10^{-6}$ S/m
$\varepsilon_1=11\cdot\varepsilon_\circ$	$\varepsilon_2=3\cdot\varepsilon_\circ$
$\mu_1=\mu_\circ$	$\mu_2=\mu_\circ$
a=15m	d=0.5a
L=400m	f=150kHz

The system may represent a configuration with desirable heating characteristics. If it is wished to heat a formation of saturated region conductivity of $\sigma_1=10^{-2}$S/m (for which $\varepsilon_2'=3\cdot\varepsilon_0$ and $\varepsilon_1'\approx30\cdot\varepsilon_0$) with the same heating pattern, we may apply the scaling equations as follows

$$p=\frac{\sigma_1'}{\sigma_1}\sqrt{\frac{\varepsilon_2}{\varepsilon_2'}}=\frac{10^{-2}}{10^{-3}}=10$$

therefore

$$x = 10x'$$

so

$$L=\frac{400}{10}=40\text{m}$$

$$a=\frac{15}{10}=1.5\text{m}$$

$$d=\frac{7.5}{10}=0.75\text{m}$$

Applying the second scaling equation gives

$$\gamma=\frac{\omega'}{\omega}=\frac{\sigma_1'\varepsilon_2}{\sigma_1\varepsilon_2'}=\frac{10^{-2}\cdot3\cdot\varepsilon_\circ}{10^{-3}\cdot3\cdot\varepsilon_\circ}=10$$

so

$$f' = 10f = 10(150\text{kHz}) = 1.5\text{MHz}$$

This new system will accurately reproduce the heating pattern of the original system provided

$$\frac{\sigma_1}{\omega\varepsilon_1} \gg 1$$

and

$$\frac{\sigma_2}{\omega\varepsilon_2} \ll 1$$

For typical oilsand, the electrical conductivity varies from 10^{-2}-10^{-4} S/m. However, the conductivity of depleted oilsand is generally constant at about 10^{-6} S/m. Thus, in the present case

$$\frac{\sigma_1}{\omega\varepsilon_1} = 10.9 \quad \text{and} \quad \frac{\sigma_1'}{\omega'\varepsilon_1'} = 4$$

and

$$\frac{\sigma_2}{\omega\varepsilon_2} = 0.04 \quad \text{and} \quad \frac{\sigma_2'}{\omega'\varepsilon_2'} = 0.004$$

The scaling approximation becomes less accurate for smaller saturated region conductivities since the loss tangents of the model and full-scale saturated regions becomes prohibitively small. For the cases considered here though, the application of scaling theory can generally be applied with accurate results.

Appendix 2.

Numerical Field-Solving Program CARHERTZ

This appendix is devoted to explanation of the numerical field-solving program CARHERTZ. The program has been developed by the author to numerically calculate the electromagnetic fields for a parallel-plate waveguide loaded with an arbitrary medium. This program represents the basic version of a program ultimately capable of calculating the electromagnetic heating within a parallel-plate waveguide taking into account such factors as temperature dependent material properties, thermal conduction and fluid flow. The two-dimensional Cartesian coordinate based program divides the solution domain into grid blocks of specified size (see Figure A). Then an equation describing the magnetic field H for a given grid block is generated from the difference form of Maxwell's curl equations. Applying the equation to every grid block in the domain creates a coefficient matrix for a linear equation in H. With a suitable excitation vector applied to the waveguide, the eigenvalue problem can be solved by Gaussian elimination to yield the magnetic field solution. The corresponding electric fields may be found by back substitution.

In order to numerically calculate the fields in the parallel-plate waveguide, the domain of interest and a strip of the surrounding medium are firstly divided into grid blocks. The surrounding or 'phantom' blocks are assigned the appropriate properties, in this case, the properties of the perfectly conducting waveguide walls. The nature of the co-efficient generating algorithm is that it deals with all

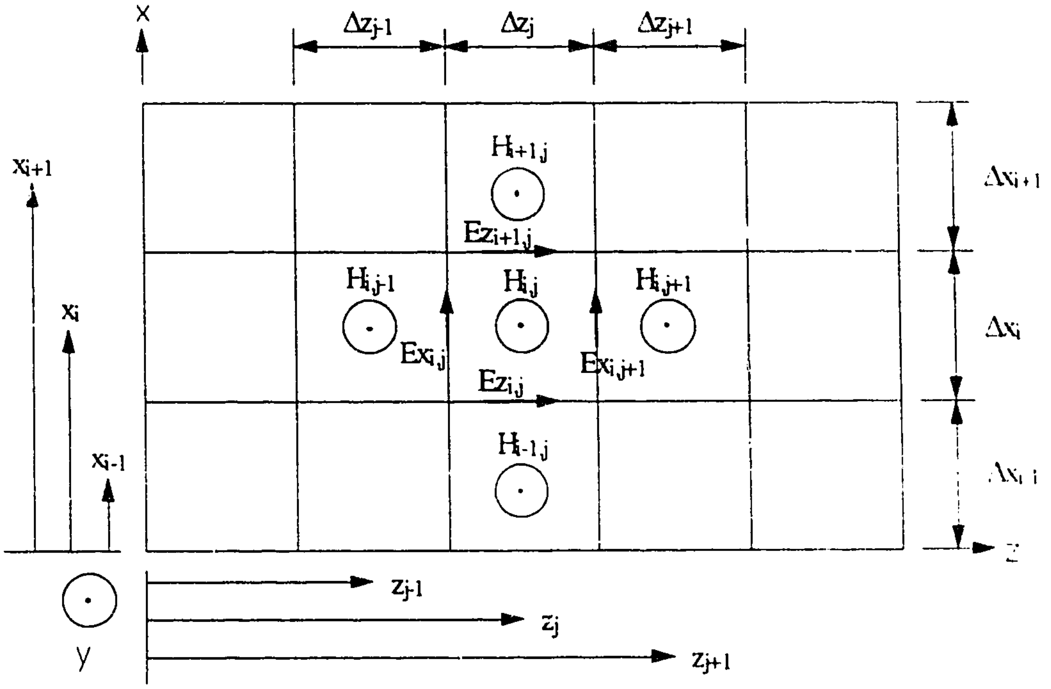

Figure A. *General grid arrangement and field conventions for the numerical program CARHERTZ.*

domain blocks in the same fashion. There are no special algorithms to handle special cases(i.e. boundary blocks). When the program evaluates the co-efficients for a boundary block, it uses the properties of the adjacent phantom blocks directly, as if they were ordinary blocks.

The derivation of the general equations for a grid block follows. Consider the general picture of the grid arrangement shown in Figure A.

For the grid block in question we can write the central equation

$$\int \mathbf{E}\cdot d\mathbf{l} = -j\omega\mu \int \mathbf{H}\cdot d\mathbf{S}$$

or, in difference form applied to the block

$$Ez_{i,j}\Delta z_j + Ex_{i,j+1}\Delta x_i - Ez_{i+1,j}\Delta z_j - Ex_{i,j}\Delta x_i = -j\omega\mu_{i,j}H_{i,j}\Delta x_i\Delta z_j$$

Rearranging gives

$$Ez_{i,j}\Delta z_j + Ex_{i,j+1}\Delta x_i - Ez_{i+1,j}\Delta z_j - Ex_{i,j}\Delta x_i + j\omega\mu_{i,j}H_{i,j}\Delta x_i\Delta z_j = 0$$

For the same block we may write the perimeter equation

$$\int \mathbf{H}\cdot d\mathbf{l} = \int (\sigma + j\omega\varepsilon)\mathbf{E}\cdot d\mathbf{S}$$

or in difference form

$$H_{i,j}\Delta y - H_{i-1,j}\Delta y = (\sigma+j\omega\varepsilon)_{i,j}\left(\frac{\Delta x_i}{2}\Delta y\right)Ez_{i,j} + (\sigma+j\omega\varepsilon)_{i-1,j}\left(\frac{\Delta x_{i-1}}{2}\Delta y\right)Ez_{i,j}$$

Rearranging gives

$$Ez_{i,j} = \frac{H_{i,j} - H_{i-1,j}}{(\sigma+j\omega\varepsilon)_{i,j}\left(\frac{\Delta x_i}{2}\right) + (\sigma+j\omega\varepsilon)_{i-1,j}\left(\frac{\Delta x_{i-1}}{2}\right)}$$

We may write the perimeter equation for the rest of the surrounding grid blocks

$$H_{i,j}\Delta y - H_{i,j+1}\Delta y = (\sigma+j\omega\varepsilon)_{i,j}\left(\frac{\Delta z_j}{2}\Delta y\right)Ex_{i,j+1} + (\sigma+j\omega\varepsilon)_{i,j+1}\left(\frac{\Delta z_{j+1}}{2}\Delta y\right)Ex_{i,j+1}$$

Rearranging gives

$$Ex_{i,j+1}=\frac{H_{i,j}-H_{i,j+1}}{(\sigma+j\omega\varepsilon)_{i,j}(\frac{\Delta z_j}{2})+(\sigma+j\omega\varepsilon)_{i,j+1}(\frac{\Delta z_{j+1}}{2})}$$

$$H_{i,j}\Delta y-H_{i+1,j}\Delta y=-(\sigma+j\omega\varepsilon)_{i,j}(\frac{\Delta x_i}{2}\Delta y)Ez_{i+1,j}-(\sigma+j\omega\varepsilon)_{i+1,j}(\frac{\Delta x_{i+1}}{2}\Delta y)Ez_{i+1,j}$$

Rearranging gives

$$-Ez_{i+1,j}=\frac{H_{i,j}-H_{i+1,j}}{(\sigma+j\omega\varepsilon)_{i,j}(\frac{\Delta x_i}{2})+(\sigma+j\omega\varepsilon)_{i+1,j}(\frac{\Delta x_{i+1}}{2})}$$

$$H_{i,j}\Delta y-H_{i,j-1}\Delta y=-(\sigma+j\omega\varepsilon)_{i,j}(\frac{\Delta z_i}{2}\Delta y)Ex_{i,j}-(\sigma+j\omega\varepsilon)_{i,j-1}(\frac{\Delta z_{j-1}}{2}\Delta y)Ex_{i,j}$$

Rearranging gives

$$-Ex_{i,j}=\frac{H_{i,j}-H_{i,j-1}}{(\sigma+j\omega\varepsilon)_{i,j}(\frac{\Delta z_j}{2})+(\sigma+j\omega\varepsilon)_{i,j-1}(\frac{\Delta z_{j-1}}{2})}$$

The perimeter equations may be substituted into the central equation which results in a linear equation in H.

$$\{\ \}H_{i-1,j}+\{\ \}H_{i,j+1}+\{\ \}H_{i+1,j}+\{\ \}H_{i,j-1}+\{\ \}H_{i,j}=0$$

where { } designates the various coefficients.

This equation may be written for every domain block in terms of the properties of the surrounding blocks, including phantom blocks. Doing this results in a matrix equation in H with a voltage excitation applied at each grid block; however, in practice usually only blocks at the waveguide input are excited. The matrix may then be solved numerically for the magnetic fields with all the remaining fields obtained through back-substitution.

All of the relevant system parameters such as the electrical properties for each grid block (including phantom blocks), grid dimensions, operating frequency and waveguide excitation vectors are entered at the beginning of the program. A detailed explanation of the variables is also given there. Further clarification can be made with reference to Figure A. Theoretically, any domain grid block may be excited. However, the present form of the program allows for the common excitation of the waveguide input grid blocks only. In addition, although electric or magnetic field excitation is possible, the current program provides for transverse electric field excitation. Finally, the waveguide termination is short-circuited for the reason that it can be accurately modelled numerically.

CARHERTZ was developed in Fortran and taylored for execution using MPW Fortran and Fortran 77 with no modification and also using MacFortran 020 with slight I/O modification.

The program output occurs on two levels: one for monitoring intermediate results for obvious programming errors and one for the output of final results. On output unit 7, intermediate program results such as the co-efficient matrix for the magnetic field system of equations and the grid block electrical properties are displayed for troubleshooting purposes. The solution set for the transverse and axial electric fields with the corresponding grid co-ordinates are output to unit 9. Unit 9 output is formatted for direct transfer of the data to Macintosh graphics programs such as 'Cricketgraph' or SYSTAT.

```
C PROGRAM CARHERTZ
C
C
      PROGRAM CAHERZ
C
C     THIS PROGRAM CALCULATES THE 2-DIMENSIONAL TIME-HARMONIC MAGNETIC
C     AND ELECTRIC FIELD DISTRIBUTIONS FOR PARALLEL-PLATE WAVEGUIDE
C     PROBLEMS IN CARTESIAN CO-ORDINATES.  NO VARIATION ACROSS THE
C     WIDTH OF THE WAVEGUIDE IS ASSUMED.  THE DIRECTION PERPENDICULAR
C     TO THE PLATES IS TERMED EITHER THE X OR R DIRECTION AND THE
C     AXIAL DIRECTION IS Z.  THE PROBLEM DOMAIN IS SUBDIVIDED INTO
C     GRIDBLOCKS HAVING VARIABLE RADIAL AND AXIAL DIMENSIONS.  THE
C     ELECTRICAL CONDUCTIVITY, RELATIVE DIELECTRIC CONSTANT, AND
C     RELATIVE MAGNETIC PERMEABILITY OF EACH GRID BLOCK ARE
C     SPECIFIED.  THE GRID BLOCKS MAY HAVE ARBITRARY ELECTRICAL
C     PROPERTIES.
C
C     A SINGLE LAYER OF "PHANTOM" GRID BLOCKS BORDERS ON EACH OF THE
C     FOUR SIDES OF THE PROBLEM DOMAIN.  EACH OF THESE GRID BLOCKS IS
C     ASSIGNED A "PHANTOM" ELECTRICAL CONDUCTIVITY AND RELATIVE
C     DIELECTRIC CONSTANT, WHOSE VALUES DEPEND UPON THE LOCAL BOUNDARY
C     CONDITION.  A COMMON ALGORITHM IS NOW REPEATEDLY USED TO
C     GENERATE AS MANY EQUATIONS FOR THE MAGNETIC FIELD H AS THERE
C     ARE GRID BLOCKS IN THE PROBLEM DOMAIN.  THESE EQUATIONS HAVE THE
C     FORM KK*H=F, WHERE KK IS THE CO-EFFICIENT MATRIX AND F IS THE
C     EXCITATION VECTOR.  THE CO-EFFICIENTS KK OF THESE EQUATIONS
C     ARE STORED IN A COMPRESSED CO-EFFICIENT MATRIX K WHICH IS THEN
C     USED TO SOLVE THE EQUATIONS BY GAUSSIAN ELIMINATION TO
C     YIELD H.  THE MAGNETIC FIELD DISTRIBUTION H IS THEN USED TO SOLVE
C     FOR THE ELECTRIC FIELD DISTRIBUTION E.
C
C*********************************************************************
C  INPUT VARIABLES
C*********************************************************************
C
C     NR - NUMBER OF TRANSVERSE GRID BLOCKS IN THE PROBLEM DOMAIN
C     NZ - NUMBER OF AXIAL GRID BLOCKS IN THE PROBLEM DOMAIN
C     DELTAR(NR) - VECTOR CONTAINING THE BLOCK DIMENSIONS IN THE
C                  TRANSVERSE DIRECTION, STARTING WITH THE BLOCK
C                  NEAREST THE AXIS (M).
C     DELTAZ(NZ) - VECTOR CONTAINING THE BLOCK DIMENSIONS IN THE
C                  AXIAL DIRECTION, STARTING WITH THE BLOCK NEAREST
C                  THE EXCITATION.(M)
C     NREG - THE NUMBER OF REGIONS OF DIFFERENT ELECTRICAL PROPERTIES
C            THAT ARE OVERLAID TO DEFINE THE ELECTRICAL PROPERTIES
C            OF THE PROBLEM DOMAIN.
C     DOMIND(NREG,4) - ARRAY THAT DEFINES THE GRID INDICES OF THE
C                      BOUNDARIES OF ALL REGIONS. ITS N'TH ROW
C                      CONSISTS OF MINI, MAXI, MINJ, MAXJ OF THE
```

```
C                N'TH REGION, WHERE
C               MINI - THE INTEGER INDEX THAT DEFINES FOR ANY
C                    REGION THE LINE OF GRID BLOCKS PARALLEL
C                    TO AND NEAREST THE AXIS.
C               MAXI - THE INTEGER INDEX THAT DEFINES FOR ANY
C                    REGION THE LINE OF GRID BLOCKS PARALLEL
C                    TO AND FURTHEST FROM THE AXIS.
C               MINJ - THE INTEGER INDEX THAT DEFINES FOR ANY
C                    REGION THE LINE OF GRID BLOCKS PERPENDICULAR
C                    TO THE AXIS AND NEAREST THE ORIGIN.
C               MAXJ - THE INTEGER INDEX THAT DEFINES FOR ANY
C                    REGION THE LINE OF GRID BLOCKS PERPENDICULAR
C                    TO THE AXIS AND FURTHEST FROM ORIGIN.
C    DOMPRP(NREG,3) - ARRAY WHOSE N'TH ROW CONSISTS OF SIGMA
C                (ELECTRICAL CONDUCTIVITY S/M), EPSREL (RELATIVE
C                DIELECTRIC CONSTANT), AND MUREL (RELATIVE
C                MAGNETIC PERMEABILITY) OF THE N'TH REGION.
C    FREQ - FREQUENCY OF OPERATION (HZ)
C    EXCMAG(NR) - VECTOR CONTAINING THE MAGNITUDES OF THE TRANSVERSELY
C             DIRECTED ELECTRIC EXCITATION FIELDS IN A PLANE
C             PERPENDICULAR TO THE AXIS AND THROUGH THE ORIGIN.
C             THE FIRST VECTOR COMPONENT CORRESPONDS TO THE
C             EXCITATION FIELD AT THE GRID BLOCK NEAREST THE
C             AXIS AND THE LAST VECTOR COMPONENT CORRESPONDS TO
C             THE EXCITATION FIELD AT THE GRID BLOCK FURTHEST
C             FROM THE AXIS (V/M).
C    EXCANG(NR) - VECTOR CONTAINING THE PHASES CORRESPONDING TO
C             EXCMAG(NR) (DEGREES)
C
C*********************************************************************
C  OUTPUT VARIABLES
C*********************************************************************
C
C    HMAG(NR,NZ) - ARRAY CONTAINING THE MAGNITUDES OF MAGNETIC
C              FIELD INTENSITIES AT THE CENTERS OF ALL GRID BLOCKS (A/M)
C    HANG(NR,NZ) - ARRAY CONTAINING THE ANGLES OF MAGNETIC FIELD
C              INTENSITIES AT THE CENTERS OF ALL GRID BLOCKS (DEGREES)
C    ERMAG(NR,NZ) - ARRAY CONTAINING THE MAGNITUDES OF RADIAL ELECTRIC
C              FIELD INTENSITY COMPONENTS AT THE CENTERS OF ALL
C              GRID BLOCKS (V/M)
C    ERANG(NR,NZ) - ARRAY CONTAINING THE ANGLES OF TRANSVERSE ELECTRIC
C              FIELD INTENSITY COMPONENTS AT THE CENTERS OF ALL
C              GRID BLOCKS (DEGREES)
C    EZMAG(NR,NZ) - ARRAY CONTAINING THE MAGNITUDES OF AXIAL ELECTRIC
C              FIELD INTENSITY COMPONENTS AT THE CENTERS OF ALL
C              GRID BLOCKS (V/M)
C    EZANG(NR,NZ) - ARRAY CONTAINING THE ANGLES OF AXIAL ELECTRIC
C              FIELD INTENSITY COMPONENTS AT THE CENTERS OF ALL
C              GRID BLOCKS (DEGREES)
```

```
C     HTRATE(NR,NZ) - HEATING RATE AT THE CENTERS OF ALL GRID
C                BLOCKS (W/M**3)
C
C**********************************************************************
C VARIABLES USED WITHIN THE PROGRAM
C**********************************************************************
C
C     NR1 - NR+1
C     NZ1 - NZ+1
C     NRNZ - NR*NZ
C     SIGJWE(0:NR1,0:NZ1) - ARRAY CONTAINING SIGMA + JW*EPSILON OF
C                    EVERY DOMAIN AND PHANTON GRID BLOCK
C     WMU(NR,NZ) - ARRAY CONTAINING W*MU OF EVERY DOMAIN GRID BLOCK
C     R(0:NR1) - VECTOR THAT CONTAINS ALL THE RADIAL DISTANCES FROM
C             THE AXIS TO THE CENTER OF EVERY GRID BLOCK.  ARBITRARY
C             VALUES FOR THE RADIAL DISTANCES TO THE CENTERS OF THE
C             INNERMOST AND OUTERMOST PHANTOM GRID BLOCKS ARE
C             LOCATED IN R(0) AND R(NR1).
C     Z(0:NZ1) - VECTOR THAT CONTAINS ALL THE AXIAL DISTANCES FROM
C             THE AXIS TO THE CENTER OF EVERY GRID BLOCK.  ARBITRARY
C             VALUES FOR THE AXIAL DISTANCES TO THE CENTERS OF THE
C             INNERMOST AND OUTERMOST PHANTOM GRID BLOCKS ARE
C             LOCATED IN Z(0) AND Z(NZ1).
C     DELR(0:NR1) - VECTOR THAT CONTAINS ALL RADIAL GRID BLOCK DIMENSIONS.
C               ARBITRARY TRANSVERSE GRID BLOCK DIMENSIONS FOR THE
C               INNERMOST AND OUTERMOST PHANTOM GRID BLOCKS ARE IN
C               DELR(0) AND DELR(NR1).
C     DELZ(0:NZ1) - VECTOR THAT CONTAINS ALL AXIAL GRID BLOCK DIMENSIONS.
C               ARBITRARY AXIAL GRID BLOCK DIMENSIONS FOR THE
C               INNERMOST AND OUTERMOST PHANTOM GRID BLOCKS
C               ARE IN DELZ(0) AND DELZ(NZ1).
C     KK(NRNZ,NRNZ) - ARRAY OF COMPLEX CO-OFFICIENTS KK OF THE EQUATIONS
C                 KK*H = F. THE CO-EFFICIENT MATRIX KK IS COMPUTED
C                 AND STORED ONLY FOR INITIAL TESTING OF THIS PROGRAM.
C     K(NRNZ,NR1) - ARRAY OF THE COMPLEX CO-OFFICIENTS OF THE COMPRESSED
C               MATRIX K. NOTE THAT NR1 IS THE HALF BANDWIDTH OF THE
C               MATRIX KK (INCLUDING THE DIAGONAL).
C     F(NRNZ) - COMPLEX VECTOR F WHOSE FIRST NR ELEMENTS CONSIST OF
C             THE REAL AND IMAGINARY PARTS OF THE COMPLEX EXCITATION
C             VOLTAGES CONSTRUCTED FROM EXCMAG(NR), AND EXCANG(NR)
C             AND DELTAR(NR). THE REMAINING ELEMENTS ARE ZERO.
C     H(NRNZ) - COMPLEX VECTOR H WHOSE ELEMENTS ARE THE SOLUTION
C             TO KK*H = F
C
C**********************************************************************
C  DIMENSIONING AND DECLARATION OF INPUT AND OUTPUT VARIABLES
C**********************************************************************
      INTEGER NR,NZ,NREG
      PARAMETER (NR=20,NZ=25,NREG=2)
      INTEGER DOMIND(NREG,4)
      REAL DELTAR(NR), DELTAZ(NZ), DOMPRP(NREG,3)
      REAL FREQ, EXCMAG(NR), EXCANG(NR)
      REAL HMAG(NR,NZ), HANG(NR,NZ), ERMAG(NR,NZ), ERANG(NR,NZ)
      REAL EZMAG(NR,NZ), EZANG(NR,NZ), HTRATE(NR,NZ)
C
C**********************************************************************
C DIMENSIONING AND DECLARATION OF VARIABLES USED WITHIN THE PROGRAM
C**********************************************************************
C
      INTEGER NR1,NZ1
      PARAMETER (NR1=NR+1, NZ1=NZ+1)
      COMPLEX SIGJWE(0:NR1,0:NZ1)
      REAL WMU(0:NR1,0:NZ1), R(0:NR1),Z(0:NZ1)
      REAL DELR(0:NR1),DELZ(0:NZ1),MU
      PARAMETER (NRNZ=NR*NZ)
      COMPLEX KK(NRNZ,NRNZ)
      COMPLEX K(NRNZ,NR1),KCOPY(NRNZ,NR1), F(NRNZ),FCOPY(NRNZ), H(NRNZ)

C THESE ARE DUMMY VARIABLES
      COMPLEX EX(NR1,NZ1), EZ(NR1,NZ1), HH(0:NR1,0:NZ1)
         COMPLEX EXAUG(NR,NZ), EZAUG(NR,NZ)
         CHARACTER*1 TAB
      TAB=CHAR(9)
C
C**********************************************************************
C INPUT OF REMAINING DATA
C**********************************************************************
C
      DATA DELTAR /20*0.75/
      DATA DELTAZ /25*16./
      DATA ((DOMIND(I,J),J=1,4),I=1,NREG)
C     NEXT LINE:  MINI,MAXI,MINJ,MAXJ
     C         /  1,  2,  1, 25,
     C            3, 20,  1, 25/
      DATA ((DOMPRP(I,J),J=1,3),I=1,NREG)
C     NEXT LINE:  SIGMA,EPSREL,MUREL
     C         / 0.01,   11.0, 1.0,
     C           0.000001, 3.00, 1.0/
      DATA FREQ /9.E5/
         DATA PI/3.141592654/
      DATA EXCMAG /1*0.,100.,18*0./
      DATA EXCANG /20*0./
C
C     OPEN(UNIT=7,FILE='MATRIX',STATUS='NEW')
C     OPEN(UNIT=10,FILE='OUTH',STATUS='NEW')
C     OPEN(UNIT=11,FILE='OUTEX',STATUS='NEW')
C     OPEN(UNIT=12,FILE='OUTEZ',STATUS='NEW')
      OPEN(UNIT=13,FILE='PDENS',STATUS='NEW')
```

```
C
C**********************************************************************
C MAIN PROGRAM
C**********************************************************************
C
C ASSIGNMENT OF ELECTRICAL PROPERTIES TO ALL DOMAIN AND PHANTOM BLOCKS
C
      CALL ASSIGN
     C    (NR,NR1,NZ,NZ1,NREG,DOMIND,DOMPRP,FREQ,SIGJWE,WMU)
C
C CALCULATION OF RADIAL DISTANCES TO CENTERS OF GRID BLOCKS
C
      SUM=0.0
      DO 10 I=1,NR
      R(I)=SUM+DELTAR(I)/2.0
      SUM=SUM+DELTAR(I)
   10 CONTINUE
C
C CALCULATION OF AXIAL DISTANCES TO CENTERS OF GRID BLOCKS
C
      SUM=0.0
      DO 20 I=1,NZ
      Z(I)=SUM+DELTAZ(I)/2.0
      SUM=SUM+DELTAZ(I)
   20 CONTINUE
C
C ASSIGNMENT OF ARBITRARY RADIAL AND AXIAL DISTANCES TO THE CENTERS OF
C PHANTOM GRID BLOCKS THAT ARE ADJACENT TO THE AXES AND FURTHEST
C FROM THE AXES
C
      R(0)=1.0
      R(NR1)=1.0
      Z(0)=1.0
      Z(NZ1)=1.0
C
C CONSTRUCTION OF DELR(0:NR1) FROM DELTAR(NR)
C
      DO 30 I=1,NR
      DELR(I)=DELTAR(I)
   30 CONTINUE
      DELR(0)=1.0
      DELR(NR1)=1.0
C
C CONSTRUCTION OF DELZ(0:NZ1) FROM DELTAZ(NZ)
C
      DO 40 I=1,NZ
      DELZ(I)=DELTAZ(I)
   40 CONTINUE
      DELZ(0)=1.0
      DELZ(NZ1)=1.0
C
C  COMPUTATION OF THE REGULAR COEFFICIENT ARRAY KK
C
      CALL KAK(NR,NR1,NZ,NZ1,NRNZ,DELR,DELZ,SIGJWE,WMU,KK)
C
C
C  COMPUTATION OF THE COMPRESSED COEFFICIENT ARRAY K
C
      CALL COEFFK(KK,NR,NR1,NRNZ,K)
C
C CREATE A COPY OF K TO BE USED BY 'GAUSEL'
C
      DO 21 J=1,NR1
         DO 22 I=1,NRNZ
         KCOPY(I,J)=K(I,J)
   22 CONTINUE
   21 CONTINUE
C
C  COMPUTATION OF THE COMPLEX EXCITATION VOLTAGES F
C
      CALL VOLTF(EXCMAG,EXCANG,NR,NRNZ,DELTAR,F)
C
C CREATE A COPY OF F TO BE USED BY 'GAUSEL'
C
      DO 24 I=1,NRNZ
         FCOPY(I)=F(I)
   24 CONTINUE
C
C  COMPUTATION OF THE VECTOR H BY GAUSSIAN ELIMINATION
C
      CALL GAUSEL(NRNZ,NR1,KCOPY,FCOPY,H)
C
C CALCULATE HMAG AND HANG FROM H
C
      DO 330 M=1,NRNZ
         I=M-((M-1)/NR)*NR
         J=(M-1)/NR+1
      HMAG(I,J)=SQRT((REAL(H(M))**2)+(AIMAG(H(M))**2))
         REELH=REAL(H(M))
C
C IF THE REAL PART OF H IS VERY SMALL THEN TAKE THE ABSOLUTE
C VALUE OF IT WHEN CALCULATING THE ANGLE TO AVOID POSSIBLE PHASE ERRORS
C DO TO COMPUTER ROUNDOFF
C
         IF(REELH.LE.1.E-20) REELH=ABS(REELH)
      HANG(I,J)=180/PI*ATAN(AIMAG(H(M))/REELH)
  330 CONTINUE
C
```

```
C COMPUTATION OF EH(NR,NZ) AND EZ(NR,NZ) FROM H
C
   CALL EFIELD(NR,NR1,NZ,NZ1,NRNZ,SIGJWE,WMU,
  *ERMAG,ERANG,EZMAG,EZANG,H,DELR,DELZ,EHCMAG,EHCANG,
  *EH,EZ,HH,EHAUG,EZAUG)
C
C
C COMPUTATION OF THE RESISTIUE HEATING RATE
C
   CALL POWER(NR,NR1,NZ,NZ1,ERMAG,EZMAG,SIGJWE,HTRATE)
C
C_____________________________________________________________________
C OUTPUT OF THE PROGRAM RESULTS TO THE SCREEN
C_____________________________________________________________________
C
C PRINT OUT THE COEFFICIENT MATRIH KK
C
C   DO 5 I=0,NR1
C  5 WRITE(7,102)(SIGJWE(I,J),J=0,NZ1)
C 102 FORMAT(' ',24E10.3)
C
C   DO 6 I=0,NR1
C  6 WRITE(7,103)(WMU(I,J),J=0,NZ1)
C 103 FORMAT(' ',12E10.3)
C
C   DO 50 I=1,NRNZ
C  50 WRITE(7,100)(KK(I,J),J=1,NRNZ)
C 100 FORMAT(' ',40E10.3)
C
C PRINT OUT THE COEFFICIENT MATRIH K
C
C   DO 51 I=1,NRNZ
C  51 WRITE(7,101)(K(I,J),J=1,NR1)
C 101 FORMAT(' ',22E10.3)
C
C PRINT OUT THE EHCITATION UECTOR F
C
C   DO 52 I=1,NR
C  52 WRITE(7,104)(F(I))
C 104 FORMAT(' ',2E10.3)
C
C PRINT OUT THE SOLUTION UECTOR H
C
C   DO 56 I=1,NRNZ
C  56 WRITE(7,106)I,(H(I))
C 106 FORMAT(' ',I3,2E10.3)
C
C OUTPUT THE MATRIH EH
C
```

```
C     DO 60 I=1,NR
C  60 WRITE(7,200)(R(I),(Z(J),ERMAG(I,J),ERANG(I,J)),J=1,NZ)
C 200 FORMAT(' ','AT (',F7.3,',',F7.3,') EX=',E12.5,' AT ',F6.2,' DEGS')
C
C  OUTPUT THE MATRIX EZ
C
C     DO 70 I=1,NR
C  70 WRITE(7,300)(R(I),(Z(J),EZMAG(I,J),EZANG(I,J)),J=1,NZ)
C 300 FORMAT(' ','AT (',F7.3,',',F7.3,') EZ=',E12.5,' AT ',F6.2,' DEGS')
C
C  OUTPUT THE HEATING RATE
C
C     DO 80 I=1,NR
C  80 WRITE(7,400)(R(I),(Z(J),HTRATE(I,J)),J=1,N2)
C 400 FORMAT(' ','AT (',F7.3,',',F7.3,') THE HEATING RATE=',E12.5,' W/M3')
C
C
C______________________________________________________________________
C FILES FOR OUTPUT TO CRICKETGRAPH
C______________________________________________________________________
C
C
C PRINT THE MAGNETIC FIELD TO FILE 'OUTH' FOR OUTPUT ON CRICKETGRAPH
C
C     DO 199 M=1,NRNZ
C     I=M-((M-1)/NR)*NR
C     J=(M-1)/NR+1
C     WRITE(10,*) R(I),TAB,Z(J),TAB,HMAG(I,J),TAB,HANG(I,J)
C 199 CONTINUE
C
C PRINT THE X-ELECTRIC FIELD TO FILE 'OUTEX' FOR OUTPUT ON CRICKETGRAPH
C
C     DO 201 I=1,NR
C     DO 202 J=1,NZ
C     WRITE(11,*) R(I),TAB,Z(J),TAB,ERMAG(I,J),TAB,ERANG(I,J)
C 202 CONTINUE
C 201 CONTINUE
C
C PRINT THE Z-ELECTRIC FIELD TO FILE 'OUTEZ' FOR OUTPUT ON CRICKETGRAPH
C
C     DO 203 I=1,NR
C     DO 204 J=1,NZ
C     WRITE(12,*) R(I),TAB,Z(J),TAB,EZMAG(I,J),TAB,EZANG(I,J)
C 204 CONTINUE
C 203 CONTINUE
C
C PRINT THE HEATING RATE TO FILE 'PDENS' FOR OUTPUT ON CRICKETGRAPH
C
      DO 205 I=1,NR
      DO 206 J=1,NZ
      WRITE(13,*) R(I),TAB,Z(J),TAB,HTRATE(I,J)
  206 CONTINUE
  205 CONTINUE

      STOP
      END
C
C***********************************************************************
C  SUBROUTINES
C***********************************************************************
C
      SUBROUTINE ASSIGN
     C     (NR,NR1,NZ,NZ1,NREG,DOMIND,DOMPRP,FREQ,SIGJWE,WMU)
      COMPLEX SIGJWE(0:NR1,0:NZ1),CMPLX
      REAL WMU(0:NR1,0:NZ1), DOMPRP(NREG,3), MU
      INTEGER DOMIND(NREG,4)
C
C  INITIALIZE ALL BLOCKS TO HAVE SIGJWE="INFINITY"
C
      DO 18 I=0,NR1
      DO 19 J=0,NZ1
      SIGJWE(I,J)=CMPLX(1.E30,1.E30)
   19 CONTINUE
   18 CONTINUE
C
C  INITIALIZE ALL BLOCKS TO HAVE WMU="WMU-NOT"
C
      DO 10 II=0,NR1
      DO 20 JJ=0,NZ1
      WMU(II,JJ)=2*(3.141592654**2)*FREQ*4*1.E-7
   20 CONTINUE
   10 CONTINUE
C
C  ASSIGN TO ALL BLOCKS THE APPROPRIATE PARAMETERS
C
      DO 30 I=1,NREG
      MINI=DOMIND(I,1)
      MAXI=DOMIND(I,2)
      MINJ=DOMIND(I,3)
      MAXJ=DOMIND(I,4)
      W=2*3.141592654*FREQ
      SIG=DOMPRP(I,1)
      EPS=DOMPRP(I,2)*8.8542E-12
      MU=DOMPRP(I,3)*4*3.141592654*1.E-7
      DO 21 K=MINI,MAXI
         DO 11 J=MINJ,MAXJ
         SIGJWE(K,J)=CMPLX(SIG,W*EPS)
         WMU(K,J)=W*MU
   11 CONTINUE
   21 CONTINUE
   30 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE KAK(NR,NR1,NZ,NZ1,NRNZ,DELX,DELZ,SIGJWE,WMU,KK)
      COMPLEX SIGJWE(0:NR1,0:NZ1),KK(NRNZ,NRNZ),DEN1,DEN2,DEN3,DEN4
      REAL DELX(0:NR1),DELZ(0:NZ1),WMU(0:NR1,0:NZ1)
      COMPLEX CMPLX
C
C INITIALIZE THE MATRIX KK
C
      DO 8 K=1,NRNZ
      DO 7 L=1,NRNZ
      KK(K,L)=CMPLX(0.0,0.0)
    7 CONTINUE
    8 CONTINUE
C
      DO 5 M=1,NRNZ
      I=M-((M-1)/NR)*NR
      J=(M-1)/NR+1
C
      DEN1=(SIGJWE(I,J)*DELX(I)+SIGJWE(I-1,J)*DELX(I-1))/2
      DEN2=(SIGJWE(I,J)*DELZ(J)+SIGJWE(I,J+1)*DELZ(J+1))/2
      DEN3=(SIGJWE(I,J)*DELX(I)+SIGJWE(I+1,J)*DELX(I+1))/2
      DEN4=(SIGJWE(I,J)*DELZ(J)+SIGJWE(I,J-1)*DELZ(J-1))/2
C
      IF(I.EQ.1) GOTO 10
      KK(M,M-1)=-DELZ(J)/DEN1
   10 IF(I.EQ.NR) GOTO 20
      KK(M,M+1)=-DELZ(J)/DEN3
   20 IF(J.EQ.1) GOTO 30
      KK(M,M-NR)=-DELX(I)/DEN4
   30 IF(J.EQ.NZ) GOTO 40
      KK(M,M+NR)=-DELX(I)/DEN2
   40 KK(M,M)=DELZ(J)/DEN1+DELX(I)/DEN2+DELZ(J)/DEN3+DELX(I)/DEN4
     C      +CMPLX(0.0,WMU(I,J))*DELX(I)*DELZ(J)
    5 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE COEFFK(KK,NR,NR1,NRNZ,K)
      COMPLEX KK(NRNZ,NRNZ), K(NRNZ,NR1),CMPLX
C
```

```
C INITIALIZE THE MATRIH K
C
      DO 17 II=1,NRNZ
         DO 16 JJ=1,NR1
         K(II,JJ)=CMPLH(0.0,0.0)
   16 CONTINUE
   17 CONTINUE
C
      DO 20 J=1,NR1
      DO 10 I=1,NRNZ
      IF(I.GT.(NRNZ-J+1)) GO TO 100
      K(I,J) = KK(I,I+J-1)
      GOTO 10
  100 K(I,J) = CMPLH(0.0,0.0)
   10 CONTINUE
   20 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE UOLTF(EHCMAG,EHCANG,NR,NRNZ,DELTAR,F)
      REAL EHCMAG(NR),EHCANG(NR),DELTAR(NR)
      COMPLEH F(NRNZ),CMPLH
         PI=3.141592654
C
C  INITIALIZE ALL F'S TO ZERO
C
      DO 10 I=1,NRNZ
      F(I)=CMPLH(0.0,0.0)
   10 CONTINUE
C
C  CONUERT EHCMAG AND AHCANG TO RECTANGULAR FORM AND
C  THEN CALCULATE F
C
      DO 20 I=1,NR
      A=EHCMAG(I)*COS(EHCANG(I)*PI/180)
      B=EHCMAG(I)*SIN(EHCANG(I)*PI/180)
      F(I)=CMPLH(A,B)
      F(I)=F(I)*DELTAR(I)
   20 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE GAUSEL(NSIZE,MBAND,K,F,A)
C
C        THIS ROUTINE SOLUES THE LINEAR ALGEBRAIC SYSTEM OF
```

```
C      EQUATIONS K*A=F BY GAUSSIAN ELIMINATION WITHOUT PIVOTING.
C      THE MATRIX K IS SYMMETRIC, POSITIVE-DEFINITE, AND BANDED;
C      ROWS IN THE UPPER TRIANGLE ARE SHIFTED TO THE LEFT UNTIL
C      DIAGONAL TERMS ARE IN THE FIRST COLLUMN. THE SOLUTION IS RETURNED
C      IN VECTOR A.(ALL ENTRIES MUST BE IN RECTANGULAR FORM)
C
C  INPUT VARIABLES
C
C        NSIZE - ORDER OF MATRIX KK
C        MBAND - HALFBANDWIDTH OF KK (INCLUDING DIAGONAL)
C        K(NSIZE,MBAND) - COMPRESSED MATRIX
C        F(NSIZE) - EXCITATION VECTOR
C
C  OUTPUT VARIABLES
C
C        A(NSIZE) - SOLUTION VECTOR
C
      COMPLEX K(NSIZE,MBAND),A(NSIZE),F(NSIZE),C
C
      NSIZM1=NSIZE-1
C
C  FORWARD REDUCTION OF MATRIX
C
      DO 30 N=1,NSIZM1
      DO 20 L=2,MBAND
      I=N+L-1
      IF(K(N,L).EQ.(0.0,0.0).OR.I.GT.NSIZE) GO TO 20
      C=K(N,L)/K(N,1)
      J=0
      DO 10 M=L,MBAND
      J=J+1
      K(I,J)=K(I,J)-C*K(N,M)
   10 CONTINUE
      K(N,L)=C
   20 CONTINUE
   30 CONTINUE
C
C  SOLVE THE REDUCED MATRIX
C
      DO 60 N=1,NSIZM1
      DO 50 L=2,MBAND
      I=N+L-1
      IF(I.LE.NSIZE) F(I)=F(I)-K(N,L)*F(N)
   50 CONTINUE
      F(N)=F(N)/K(N,1)
   60 CONTINUE
      F(NSIZE)=F(NSIZE)/K(NSIZE,1)
C
C  BACK SUBSTITUTION
```

```
C
      A(NSIZE)=F(NSIZE)
      DO 90 N=1,NSIZM1
      J=NSIZE-N
      A(J)=F(J)
      DO 80 L=2,MBAND
      M=J+L-1
      IF(M.LE.NSIZE) A(J)=A(J)-K(J,L)*A(M)
   80 CONTINUE
   90 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE EFIELD(NR,NR1,NZ,NZ1,NRNZ,SIGJWE,WMU,
     *EXMAG,EXANG,EZMAG,EZANG,H,DELX,DELZ,EXCMAG,EXCANG,
     *EX,EZ,HH,EXAVG,EZAVG)
C
      COMPLEX H(NRNZ),EX(NR1,NZ1),EZ(NR1,NZ1),DEN1,DEN2,DEN3,DEN4
      COMPLEX HH(0:NR1,0:NZ1),SIGJWE(0:NR1,0:NZ1)
      COMPLEX EXAVG(NR,NZ), EZAVG(NR,NZ),CMPLX,EXCITE
      REAL EZMAG(NR,NZ),EZANG(NR,NZ),EXMAG(NR,NZ),EXANG(NR,NZ)
      REAL DELX(0:NR1),DELZ(0:NZ1),WMU(0:NR1,0:NZ1)
      REAL EXCMAG(NR),EXCANG(NR)


        PI=3.141592654
C
C  SET ALL H'S IN THE PHANTOM BLOCKS EQUAL TO ZERO
C
      DO 10 J=0,NZ1
      HH(0,J)=CMPLX(0.0,0.0)
   10 HH(NR1,J)=CMPLX(0.0,0.0)
      DO 20 I=1,NR
      HH(I,0)=CMPLX(0.0,0.0)
   20 HH(I,NZ1)=CMPLX(0.0,0.0)
C
C  CONVERT H(M) TO H(I,J)
C
      DO 30 M=1,NRNZ
      I=M-((M-1)/NR)*NR
      J=(M-1)/NR+1
      HH(I,J) = H(M)
   30 CONTINUE
C
C  CALCULATE THE E FIELDS FROM THE H'S
C
      DO 40 M=1,NRNZ
      I=M-((M-1)/NR)*NR
      J=(M-1)/NR+1
      DEN1=(SIGJWE(I,J)*DELX(I)+SIGJWE(I-1,J)*DELX(I-1))/2
      DEN2=(SIGJWE(I,J)*DELZ(J)+SIGJWE(I,J+1)*DELZ(J+1))/2
      DEN3=(SIGJWE(I,J)*DELX(I)+SIGJWE(I+1,J)*DELX(I+1))/2
      DEN4=(SIGJWE(I,J)*DELZ(J)+SIGJWE(I,J-1)*DELZ(J-1))/2
C
C THIS CODE CORRECTS THE TRANSVERSE E-FIELD NEAR THE EXCITATION
C i.e. THE APPLIED EXCITATION FIELD HAS TO BE SUPERIMPOSED OVER THE CALCULATED FIELD
C TO YIELD THE CORRECT ANSWER
C
      REXCIT=EXCMAG(I)*COS(PI/180*EXCANG(I))
      AIMXCT=EXCMAG(I)*SIN(PI/180*EXCANG(I))
      EXCITE=CMPLX(REXCIT,AIMXCT)
C
      EZ(I,J)=(HH(I,J)-HH(I-1,J))/DEN1
      EX(I,J+1)=(HH(I,J)-HH(I,J+1))/DEN2
      EZ(I+1,J)=-(HH(I,J)-HH(I+1,J))/DEN3
      EX(I,J)=-(HH(I,J)-HH(I,J-1))/DEN4
C
      IF(J.EQ.1) EX(I,J)=EXCITE
C
      EZAVG(I,J) = (EZ(I,J)+EZ(I+1,J))/2
      EXAVG(I,J) = (EX(I,J)+EX(I,J+1))/2
      EZMAG(I,J) = ABS(EZAVG(I,J))
      EZANG(I,J) = 180/PI*ATAN(AIMAG(EZAVG(I,J))/REAL(EZAVG(I,J)))
C
C IF RE(EZAVG)=0 THEN ASSIGN A DEFAULT VALUE OF ZERO DEGREES TO EZANG
      IF(REAL(EZAVG(I,J)).EQ.0.0) EZANG(I,J)=0.0
C
      EXMAG(I,J) = ABS(EXAVG(I,J))
      EXANG(I,J) = 180/PI*ATAN(AIMAG(EXAVG(I,J))/REAL(EXAVG(I,J)))
C
C IF RE(EXAVG)=0 THEN ASSIGN A DEFAULT VALUE OF ZERO DEGREES TO EXANG
      IF(REAL(EXAVG(I,J)).EQ.0.0) EXANG(I,J)=0.0

   40 CONTINUE
C
C PRINT OUT THE VALUES OF EZ AND EX AS CALCULATED
C
C     DO 77 II=1,NR
C     DO 88 JJ=1,NZ
C     WRITE(7,111) II,JJ,EZ(II,JJ)
C     WRITE(7,112) II,JJ,EX(II,JJ)
C  88 CONTINUE
C  77 CONTINUE
C 111 FORMAT(' ','EZ(',2I3,')=',2E12.5)
C 112 FORMAT(' ','EX(',2I3,')=',2E12.5)
C
```

```
      RETURN
      END
C
C
C
      SUBROUTINE POWER(NR,NR1,NZ,NZ1,EXMAG,EZMAG,SIGJWE,HTRATE)
C
      REAL EXMAG(NR,NZ),EZMAG(NR,NZ),HTRATE(NR,NZ)
      COMPLEX SIGJWE(0:NR1,0:NZ1)
C
      DO 20 I=1,NR
      DO 10 J=1,NZ
      ETSQRD = (EXMAG(I,J)**2)+(EZMAG(I,J)**2)
      HTRATE(I,J) = 0.5*REAL(SIGJWE(I,J))*ETSQRD
   10 CONTINUE
   20 CONTINUE
      RETURN
      END
```

Appendix 3.
Program Listings

This appendix contains fully documented listings of all the computer programs referred to in this thesis. They are listed in the order of mention in the main thesis body and are consecutively numbered as shown in the Table of Contents. All of the program listings in this appendix were written in either Fortran or Pascal and executed on a Macintosh SE/30 personal computer. Programs 1-5 and 7 were written in Pascal and executed using TurboPascal 1.00A (Borland International 1986, 1987, 1988). The remaining programs were written in Fortran and executed using MPW Fortran 3.1 (Apple Computer Inc. 1985-89).

Several graphing programs have been used to generate the figures presented in this thesis. The majority of the figures have been generated in two steps. The plots appearing in the figures were generated initially and then inserted into pagemaking programs after which the legends were added. A short mention of these programs and a listing of the figures for which they were used follows.

The most commonly used graphing program was Cricketgraph 1.3 (Cricket Software 1986, 1987, 1988). It was used to generate the plots in Figures 4, 6, 7, 8, 9, 10, 12, 13 and 19 and the field variation plots appearing in Figures 17, 18 and 20-24. The greyshade plots of Figures 15-18 and 20-24 were generated using SYSTAT 5.0 (Systat Inc. 1990).

The graphs of Figures 11 and 14 were generated using a specially developed program called Efields whose source code is program listing 7 in this appendix. The actual figures were created by inserting the plots into a drawing program called Superpaint 2.0 (Silicon Beach Software Inc. 1986, 1988, 1989) and then adding the legends.

```
{ Program 1. RepeatingMuller                                }

program Muller;

{-----------------------------------------------------------------------}
{                                               }
{                                               }
{-  This program is a modified version of the original copyrighted program -}
{-  'Muller' (TurboPascal Numerical Methods Toolbox (c) 1987, Borland     -}
{-  International). It uses Muller's method to generate the dominant mode  -}
{-  solution set of the transcendental system of equations describing the  -}
{-  propagation constants kx1, kx2 and kz within the parallel-plate        -}
{-  waveguide. The dominant mode solution set is generated incrementally   -}
{-  over a desired range of frequencies. The program generates the 3       -}
{-  required initial guesses internally. The small argument approximation  -}
{-  is used to generate the first guess after which 2 simple arithmetic    -}
{-  algorithms operate on the first guess to generate the remaining 2.     -}
{-                                              -}
{-  This program is the driver program for the Muller rootfinding          -}
{-  subroutine. The transcendental function, the variable declarations     -}
{-  and all the I/O statements are all taken care of here. The actual      -}
{-  Muller subroutine is called 'RootsOfEquat' and is listed as            -}
{-  program 5. All the information concerning the transcendental           -}
{-  equation and the system parameters is contained in the body of the     -}
{-  text of the accompanying thesis.                          -}
{-                                              -}
{-  Program input consists of physical system parameters which are located -}
{-  at the beginning of the program under the heading 'ENTRY OF SYSTEM     -}
{-  PARAMETERS.' 'freq'=frequency(Hz), 'sig1'=electrical conductivity of   -}
{-  region1 (S/m), 'sig2'=electrical conductivity of region2 (S/m),        -}
{-  'eps1'=permittivity of region1 (H/m), 'eps2'=permittivity of           -}
{-  region2 (H/m), 'mu'=magnetic permeability, 'a'=plate separation(m) and -}
{-  'd'=saturated region thickness(m).                        -}
{-                                              -}
{-  Program output is given to 3 files. The file called 'outfile' is       -}
{-  displayed on the screen and shows the frequency, the calculated roots  -}
{-  kx1, kx2 and kz and the value of the transcendental function calculated-}
{-  at the roots. The file called 'plotfile' contains frequency and the    -}
{-  3 propagation constants in rectangular form and formatted to be        -}
{-  directly read by plotting routines such as 'Cricketgraph.' Finally,    -}
{-  the file called 'smallfile' contains the same information as 'plotfile'-}
{-  however, the small argument propagation constants are given.          -}
{-                                              -}
{-  Copyright (c) 1990 Richard Maslowski                     -}
{-  All Rights Reserved                                -}
{-                                              -}
{-                                              -}
{-  Units used:  IOSelection      var OutFile : text;             -}
{-                        OutName : string;          -}
{-                        InFile  : text;         -}
{-                        InName  : string;          -}
{-                        IOerr   : boolean;        -}
{-                      procedure DisplayWarning        -}
{-                      procedure DisplayError        -}
{-                      procedure IOCheck            -}
{-                      procedure GetInputFile        -}
{-                      procedure GetOutputFile        -}
{-                                             -}
{-          RootsOfEquat     procedure Muller              -}
{-                                             -}
{-                                             -}
{-----------------------------------------------------------------------------}

{-----------------------------------------------------------------------------}

{$I-}       { Disable I/O error trapping }
{$R+}       { Enable range checking }
{$S+}       { Enable segmentation of code }

{$R IOSelection.rsrc} { Resource file for IOSelection unit }
{$U IOSelection}
{$U RootsOfEquat}

uses
 MemTypes, QuickDraw, OSIntf, ToolIntf, PackIntf, PasPrinter,
 RootsOfEquat,

{$S SecondSegment}

 IOSelection;

 {---------------------------------------------------------------------------}
 { ENTRY OF THE PHYSICAL SYSTEM PARAMETERS                        }
 {---------------------------------------------------------------------------}

const
{ enter the saturated region conductivity (S/m) }
  sig1 = 0;

{ enter the depleted region conductivity (S/m) }
  sig2 = 0;

{ enter the saturated region dielectric constant (F/m) }
  eps1 = 9.7394e-11; { normally eps1 = 9.7394e-11 }

{ enter the depleted region dielectric constant (F/m) }
  eps2 = 2.6562e-11;
```

```
{ enter the magnetic permeability for both regions (H/m)
  mu  = 12.5664e-7;

{ enter the parallel-plate separation (m) }
  a   = 15;

{ enter the saturated region thickness (m) }
  d   = 12.75;

{ VARIABLE DECLARATIONS }

var
  Guess : TNcomplex;          { Initial approximations }
  Tol : Extended;            { Tolerance in answer }
  Iter : integer;            { Number of iterations }
  Startfreq : Extended;       { Starting frequency }
  Endfreq : Extended          { Ending frequency }
  Stepfreq : Extended         { Frequency step }
  MaxIter : integer;          { Max number of iterations allowed }
  Answer, yAnswer : TNcomplex;   { Root and function evaluated at root }
  Error : byte;              { Flags an error }
  k1sqr,k2sqr: TNcomplex;
  dum,dum1,dum2: TNcomplex;
  Asqr,kx1sqr,kx2sqr,kx1,kx2: TNcomplex;
  freq,omega: Extended;
  PlotFile: text;
  SmallFile: text;
  E_File: text;
  a_over_d,smallnum,a_m_d_over_d,dumratio,multdumratio,one: TNcomplex;
  smalldenom,smallkzsqr,smallkz: TNcomplex;
  dcomp,tanarg,Etan,coef,Eratio: TNcomplex;
  Emag: extended;

{ HERE ARE SOME COMPLEX OPERATIONS }

procedure Conjugate(C1 : TNcomplex; var C2 : TNcomplex);
begin
  C2.Re := C1.Re;
  C2.Im := -C1.Im;
end; { procedure Conjugate }

procedure Real2Complex(realnumber: extended; var complexnumber: TNcomplex);
begin
  complexnumber.Re:= realnumber;
  complexnumber.Im:= 0;
end;  { procedure Real2Complex }

function Modulus(var C1 : TNcomplex) : Extended;
```

```
begin
 Modulus := Sqrt(Sqr(C1.Re) + Sqr(C1.Im));
end; { function Modulus }

procedure Add(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
 C3.Re := C1.Re + C2.Re;
 C3.Im := C1.Im + C2.Im;
end; { procedure Add }

procedure Sub(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
 C3.Re := C1.Re - C2.Re;
 C3.Im := C1.Im - C2.Im;
end; { procedure Sub }

procedure Mult(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
 C3.Re := C1.Re * C2.Re - C1.Im * C2.Im;
 C3.Im := C1.Im * C2.Re + C1.Re * C2.Im;
end; { procedure Mult }

procedure Divide(C1, C2 : TNcomplex; var C3 : TNcomplex);
var
 Dum1, Dum2 : TNcomplex;
 E : Extended;
begin
 Conjugate(C2, Dum1);
 Mult(C1, Dum1, Dum2);
 E := Sqr(Modulus(C2));
 C3.Re := Dum2.Re / E;
 C3.Im := Dum2.Im / E;
end; { procedure Divide }

procedure SquareRoot(C1 : TNcomplex; var C2 : TNcomplex);
var
 R, Theta : Extended;
begin
 R := Sqrt(Sqr(C1.Re) + Sqr(C1.Im));
 if ABS(C1.Re) < TNNearlyZero then
   begin
    if C1.Im < 0 then
      Theta := Pi / 2
    else
      Theta := -Pi / 2;
   end
 else
   if C1.Re < 0 then
    Theta := ArcTan(C1.Im / C1.Re) + Pi
```

```
  else
   Theta := ArcTan(C1.Im / C1.Re);
 C2.Re := Sqrt(R) * Cos(Theta / 2);
 C2.Im := Sqrt(R) * Sin(Theta / 2);
end; { procedure SquareRoot }

procedure ComplexSIN(C1: TNcomplex; var C2: TNcomplex);
begin
 C2.Re:=  SIN(C1.Re) * (EXP(-C1.Im) + EXP(C1.Im))/2;
 C2.Im:= -COS(C1.Re) * (EXP(-C1.Im) - EXP(C1.Im))/2
end;  { procedure ComplexSIN }

procedure ComplexCOS(C1: TNcomplex; var C2: TNcomplex);
begin
 C2.Re:= COS(C1.Re) * (EXP(-C1.Im) + EXP(C1.Im))/2;
 C2.Im:= SIN(C1.Re) * (EXP(-C1.Im) - EXP(C1.Im))/2
end;  { procedure ComplexCOS }

procedure ComplexTAN(C1: TNcomplex; var C2: TNcomplex);
var
 Dum1,Dum2: TNcomplex;
begin
 ComplexSIN(C1,Dum1);
 ComplexCOS(C1,Dum2);
 Divide(Dum1,Dum2,C2);
end;  { procedure ComplexTAN }

procedure ComplexEXP(C1: TNcomplex; var C2: TNcomplex);
begin
 C2.Re:= EXP(C1.Re) * COS(C1.Im);
 C2.Im:= EXP(C1.Re) * SIN(C1.Im)
end;  { procedure ComplexEXP }

procedure ComplexSqr(C1: TNcomplex; var C2: TNcomplex);
begin
 C2.Re:= SQR(C1.Re) - SQR(C1.Im);
 C2.Im:= 2 * C1.Re * C1.Im
end;  { procedure ComplexSqr }


{-------------------------------------------------------------------------}
{ HERE IS THE FUNCTION IN THE FORM F(KZ)=0                        }
{-------------------------------------------------------------------------}

procedure TNTargetF(kz : TNcomplex; var Y : TNcomplex);

var
 kzsqr: TNcomplex;
 cnst,cnstsqr,konst,konstsqr,C1,C2: TNcomplex;
```

```
  dcomplex,acomplex,a_minus_d,arg1,arg2: TNcomplex;
  tan1,tan2,term1,term2: TNcomplex;
  original,root1,rootfactor1,root2,rootfactor2,Yinter: TNcomplex;

begin

  complexSqr(kz,kzsqr);

  Sub(k1sqr,kzsqr,cnstsqr);
  SquareRoot(cnstsqr,cnst);

  Sub(k2sqr,kzsqr,konstsqr);
  SquareRoot(konstsqr,konst);

  Divide(cnst,dum1,C1);
  Divide(konst,dum2,C2);

  dcomplex.Re:= d; dcomplex.Im:= 0;
  acomplex.Re:= a; acomplex.Im:= 0;

  Sub(acomplex,dcomplex,a_minus_d);

  Mult(cnst,dcomplex,arg1);
  Mult(konst,a_minus_d,arg2);

  complexTAN(arg1,tan1); complexTAN(arg2,tan2);

  Mult(C1,tan1,term1); Mult(C2,tan2,term2);

  Add(term1,term2,Y);

  { FACTORIZATION OF CALCULATED ROOTS (disabled) }

  {root1.Re:= 1.639964e-2;
  root1.Im:= -2.518189e-3;

  Sub(kz,root1,rootfactor1);

  root2.Re:= -1.639964e-2;
  root2.Im:= 2.518189e-3;

  Sub(kz,root2,rootfactor2);

  Divide(original,rootfactor1,Yinter);

  Divide(Yinter,rootfactor2,Y);}

  end; { procedure TNTargetF }
```

```
{-------------------------------------------------------------------- ---}
{ INPUT TOLERANCE, MAXIMUM ITERATIONS, FREQUENCY ENDPOINTS AND STEPSIZE   }
{-------------------------------------------------------------------------}

procedure GetTolerance(var Tol : Extended);
begin
 Tol := 1E-8;
 Writeln;
 repeat
  Write('Tolerance (> 0): ');
  Readln(Tol);
  IOCheck;
  if Tol <= 0 then
  begin
   IOerr := true;
   Tol := 1E-8;
  end;
 until not IOerr;
end; { procedure GetTolerance }

procedure GetMaxIter(var MaxIter : integer);
begin
 MaxIter := 100;
 Writeln;
 repeat
  Write('Maximum number of iterations (> 0): ');
  Readln(MaxIter);
  IOCheck;
  if MaxIter < 0 then
  begin
   IOerr := true;
   MaxIter := 100;
  end;
 until not IOerr;
end; { procedure GetMaxIter }

procedure GetStartfreq(var Tol : Extended);
begin
 Startfreq := 1E+5;
 Writeln;
 repeat
  Write('Starting frequency (Hz): ');
  Readln(Startfreq);
  IOCheck;
  if Startfreq <= 0 then
  begin
   IOerr := true;
   Startfreq := 1E+5;
  end;
  until not IOerr;
end; { procedure GetStartfreq }

procedure GetEndfreq(var Tol : Extended);
begin
  Endfreq := 2E+6;
  Writeln;
  repeat
    Write('Ending frequency (Hz): ');
    Readln(Endfreq);
    IOCheck;
    if Endfreq <= 0 then
    begin
      IOerr := true;
      Endfreq := 2E+6;
    end;
  until not IOerr;
end; { procedure GetEndfreq }

procedure GetStepfreq(var Tol : Extended);
begin
  Stepfreq := 1E+4;
  Writeln;
  repeat
    Write('frequency stepsize  (+ or - in Hz): ');
    Readln(Stepfreq);
    IOCheck;
    if Stepfreq <= 0 then
    begin
      IOerr := true;
      Stepfreq := 1E+4;
    end;
  until not IOerr;
end; { procedure GetStepfreq }



{ CALCULATE KX1 AND KX2 FROM KZ }

procedure Calculatekx(inputvalue: TNcomplex);
begin
      ComplexSqr(inputvalue,Asqr);
      Sub(k1sqr,Asqr,kx1sqr);
      Sub(k2sqr,Asqr,kx2sqr);
      SquareRoot(kx1sqr,kx1);
      SquareRoot(kx2sqr,kx2);
      if kx1.Re < 0.0 then
        begin
          kx1.Re:= -kx1.Re;
      kx1.Im:= -kx1.Im
     end;
   if kx2.Re < 0.0 then
     begin
      kx2.Re:= -kx2.Re;
      kx2.Im:= -kx2.Im
     end;
end; { procedure Calculatekx }

{-----------------------------------------------------------------------------}
{ OUTPUT THE RESULTS TO THE DEVICE OUTFILE                                     }
{-----------------------------------------------------------------------------}

procedure Results(Guess   : TNcomplex;
                 Answer  : TNcomplex;
                 yAnswer : TNcomplex;
                 Tol     : Extended;
                 MaxIter : integer;
                 Iter    : integer;
                 Error   : byte);


begin
  Writeln(OutFile);
  Write(OutFile, 'frequency: ' : 30);
  Writeln(OutFile, freq/1000:7:1, ' kHz');
  Writeln(OutFile);
  if Error in [1, 2] then
    DisplayWarning;
  if Error >= 3 then
    DisplayError;

  case Error of
    1 : Writeln(OutFile, 'This will take more than ', MaxIter, ' iterations.');

    2 : begin
         Writeln(OutFile, 'A parabola which intersects the x-axis can not');
         Writeln(OutFile, 'be constructed through these three points.');
        end;

    3 : Writeln(OutFile, 'The tolerance must be greater than zero.');

    4 : Writeln(OutFile,
             'The maximum number of iterations must be greater than zero.');
  end; { case }

  if Error <= 2 then
  begin
    Calculatekx(Answer);
  Writeln(OutFile);
  Writeln(OutFile, 'Number of iterations: ' : 26, Iter:2);
  Write(OutFile, 'Calculated root (kz): ' : 26);
  Writeln(OutFile, Answer.Re:23, ' + ', Answer.Im:23, ' i');
   Write(OutFile, '    kx1      : ' : 26);
   Writeln(OutFile, kx1.Re:23, ' + ', kx1.Im:23, ' i');
   Write(OutFile, '    kx2      : ' : 26);
   Writeln(OutFile, kx2.Re:23, ' + ', kx2.Im:23, ' i');
  Writeln(OutFile, 'Value of the function  ' : 26);
  Write(OutFile, 'at the calculated root: ' : 26);
  Writeln(OutFile, yAnswer.Re:23, ' + ', yAnswer.Im:23, ' i');
  Writeln(OutFile);
   Writeln(PlotFile,freq/1000:7:1,Chr(9),Answer.Re:23,Chr(9),Answer.Im:23,
    Chr(9),kx1.Re:23,Chr(9),kx1.Im:23,Chr(9),kx2.Re:23,Chr(9),kx2.Im:23);
 end;
end; { procedure Results }

{---------------------------------------------------------------------------}
{ MAIN PROGRAM                                               }
{---------------------------------------------------------------------------}

begin { program Muller }
   GetTolerance(Tol);
   GetMaxIter(MaxIter);
   GetStartfreq(Startfreq);
   GetEndfreq(Endfreq);
   GetStepfreq(stepfreq);
   GetOutputFile(OutFile);
   GetOutputFile(PlotFile);
   GetOutputFile(SmallFile);

   Real2Complex(a/d,a_over_d);
   Real2Complex((a-d)/d,a_m_d_over_d);
   Real2Complex(1,one);

   freq:= Startfreq;

   repeat
     Writeln;Writeln;
     Writeln('now working on frequency = ', freq/1000:7:1 , ' kHz');

{ CALCULATE THE SMALL ARGUMENT APPROXIMATION CONSTANTS }

      omega:= 2*Pi*freq;

      dum1.Re:= sig1;      { for k1sqr }
      dum1.Im:= omega*eps1;

      dum2.Re:= sig2;      { for k2sqr }
      dum2.Im:= omega*eps2;

      dum.Re:= 0;        { for both k1sqr and k2sqr }
      dum.Im:= -omega*mu;

      Mult(dum1,dum,k1sqr);
      Mult(dum2,dum,k2sqr);

      Mult(k1sqr,a_over_d,smallnum);
      Divide(dum1,dum2,dumratio);
      Mult(dumratio,a_m_d_over_d,multdumratio);
      Add(one,multdumratio,smalldenom);
      Divide(smallnum,smalldenom,smallkzsqr);
      SquareRoot(smallkzsqr,smallkz);
       if smallkz.Re < 0 then
        begin
         smallkz.Re:= -smallkz.Re;
         smallkz.Im:= -smallkz.Im
       end;
      Calculatekx(smallkz);
   Writeln(SmallFile,freq/1000:7:1,Chr(9),smallkz.Re:23,Chr(9),smallkz.Im:23,
    Chr(9),kx1.Re:23,Chr(9),kx1.Im:23,Chr(9),kx2.Re:23,Chr(9),kx2.Im:23);

  Guess:= smallkz;

{ Use Muller's method with deflation to find a root }
  Muller(Guess, Tol, MaxIter, Answer, yAnswer, Iter, Error, @TNTargetF);

{ Output the results from the root finder }
  Results(Guess, Answer, yAnswer, Tol, MaxIter, Iter, Error);

     freq:= freq + Stepfreq;

   until freq > Endfreq;


  Close(OutFile);
  Close(PlotFile);
  Close(SmallFile);

  WaitReturnOrClick;
end. { program Muller }
```

```
{ Program 2. SingleStepMuller                              }

program Muller;

{-----------------------------------------------------------------------}
{                                            }
{                                            }
{- This program is a modified version of the original copyrighted program -}
{- 'Muller' (TurboPascal Numerical Methods Toolbox (c) 1987 Borland      -}
{- International). This program accepts one user-entered initial guess   -}
{- at a particular frequency and calculates the corresponding exact      -}
{- root of the transcendental function. An internal algorithm generates   -}
{- 2 required additional guesses from the user-given one.            -}
{-                                          -}
{- This program is the driver program for the Muller rootfinding        -}
{- subroutine. The transcendental function, the variable declarations    -}
{- and all the I/O statements are all taken care of here. The actual     -}
{- Muller subroutine is called 'RootsOfEquat' and is listed as program 5  -}
{- in this appendix. All the information concerning the transcendental    -}
{- equation and the system parameters is contained in the body of the     -}
{- text of the accompanying thesis.                       -}
{-                                          -}
{- Program input consists of physical system parameters which are located -}
{- at the beginning of the program under the heading 'ENTRY OF SYSTEM     -}
{- PARAMETERS.' 'freq'=frequency(Hz), 'sig1'=electrical conductivity of   -}
{- region1 (S/m), 'sig2'=electrical conductivity of region2 (S/m),       -}
{- 'eps1'=permittivity of region1 (H/m), 'eps2'=permittivity of          -}
{- region2 (H/m), 'mu'=magnetic permeability, 'a'=plate separation(m) and -}
{- 'd'=saturated region thickness(m).                      -}
{-                                          -}
{- Program output is displayed on the screen and consists of the frequency-}
{- the calculated roots kx1, kx2 and kz in rectangular form, and the value-}
{- of the transcendental function at the calculated root.            -}
{-                                          -}
{- Copyright (c) 1990 Richard Maslowski                      -}
{- All Rights Reserved                               -}
{-                                          -}
{- Units used:  IOSelection       var OutFile : text;           -}
{-                       OutName : string;         -}
{-                       InFile  : text;        -}
{-                       InName  : string;         -}
{-                       IOerr   : boolean;        -}
{-                     procedure DisplayWarning         -}
{-                     procedure DisplayError         -}
{-                     procedure IOCheck             -}
{-                     procedure GetInputFile          -}
{-                     procedure GetOutputFile          -}
{-                                          -}
{-            RootsOfEquat      procedure Muller              -}
{-                                              -}
{-                                              -}
{----------------------------------------------------------------------}

{$I-}       { Disable I/O error trapping }
{$R+}       { Enable range checking }
{$S+}       { Enable segmentation of code }

{$R IOSelection.rsrc} { Resource file for IOSelection unit }
{$U IOSelection}
{$U RootsOfEquat}

uses
  MemTypes, QuickDraw, OSIntf, ToolIntf, PackIntf, PasPrinter,
  RootsOfEquat,

{$S SecondSegment}

 IOSelection;

{ DECLARATION OF VARIABLES }

var
  Guess : TNcomplex;          { Initial approximations }
  Tol : Extended;            { Tolerance in answer }
  Iter : integer;            { Number of iterations }
  MaxIter : integer;           { Max number of iterations allowed }
  Answer, yAnswer : TNcomplex;   { Root and function evaluated at root }
  Error : byte;              { Flags an error }
  k1sqr,k2sqr: TNcomplex;
  Asqr,kx1sqr,kx2sqr,kx1,kx2: TNcomplex;


 { HERE ARE SOME NECESSARY COMPLEX PROCEDURES }


 procedure Conjugate(C1 : TNcomplex; var C2 : TNcomplex);
 begin
   C2.Re := C1.Re;
   C2.Im := -C1.Im;
 end; { procedure Conjugate }

 function Modulus(var C1 : TNcomplex) : Extended;
 begin
   Modulus := Sqrt(Sqr(C1.Re) + Sqr(C1.Im));
  end; { function Modulus }

 procedure Add(C1, C2 : TNcomplex; var C3 : TNcomplex);
```

```
begin
  C3.Re := C1.Re + C2.Re;
  C3.Im := C1.Im + C2.Im;
end; { procedure Add }

procedure Sub(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
  C3.Re := C1.Re - C2.Re;
  C3.Im := C1.Im - C2.Im;
end; { procedure Sub }

procedure Mult(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
  C3.Re := C1.Re * C2.Re - C1.Im * C2.Im;
  C3.Im := C1.Im * C2.Re + C1.Re * C2.Im;
end; { procedure Mult }

procedure Divide(C1, C2 : TNcomplex; var C3 : TNcomplex);
var
  Dum1, Dum2 : TNcomplex;
  E : Extended;
begin
  Conjugate(C2, Dum1);
  Mult(C1, Dum1, Dum2);
  E := Sqr(Modulus(C2));
  C3.Re := Dum2.Re / E;
  C3.Im := Dum2.Im / E;
end; { procedure Divide }

procedure SquareRoot(C1 : TNcomplex; var C2 : TNcomplex);
var
  R, Theta : Extended;
begin
  R := Sqrt(Sqr(C1.Re) + Sqr(C1.Im));
  if ABS(C1.Re) < TNNearlyZero then
    begin
      if C1.Im < 0 then
        Theta := Pi / 2
      else
        Theta := -Pi / 2;
    end
  else
    if C1.Re < 0 then
      Theta := ArcTan(C1.Im / C1.Re) + Pi
    else
      Theta := ArcTan(C1.Im / C1.Re);
  C2.Re := Sqrt(R) * Cos(Theta / 2);
  C2.Im := Sqrt(R) * Sin(Theta / 2);
end; { procedure SquareRoot }
```

```
procedure ComplexSIN(C1: TNcomplex; var C2: TNcomplex);
begin
  C2.Re:=  SIN(C1.Re) * (EXP(-C1.Im) + EXP(C1.Im))/2;
  C2.Im:= -COS(C1.Re) * (EXP(-C1.Im) - EXP(C1.Im))/2
end;  { procedure ComplexSIN }

procedure ComplexCOS(C1: TNcomplex; var C2: TNcomplex);
begin
  C2.Re:= COS(C1.Re) * (EXP(-C1.Im) + EXP(C1.Im))/2;
  C2.Im:= SIN(C1.Re) * (EXP(-C1.Im) - EXP(C1.Im))/2
end;  { procedure ComplexCOS }

procedure ComplexTAN(C1: TNcomplex; var C2: TNcomplex);
var
  Dum1,Dum2: TNcomplex;
begin
  ComplexSIN(C1,Dum1);
  ComplexCOS(C1,Dum2);
  Divide(Dum1,Dum2,C2);
end;  { procedure ComplexTAN }

procedure ComplexEXP(C1: TNcomplex; var C2: TNcomplex);
begin
  C2.Re:= EXP(C1.Re) * COS(C1.Im);
  C2.Im:= EXP(C1.Re) * SIN(C1.Im)
end;  { procedure ComplexEXP }

procedure ComplexSqr(C1: TNcomplex; var C2: TNcomplex);
begin
  C2.Re:= SQR(C1.Re) - SQR(C1.Im);
  C2.Im:= 2 * C1.Re * C1.Im
end;  { procedure ComplexSqr }


{---------------------------------------------------------------------}
{ HERE IS THE FUNCTION IN THE FORM F(kz)=0                      }
{---------------------------------------------------------------------}

{ ENTRY OF SYSTEM PARAMETERS }

procedure TNTargetF(kz : TNcomplex; var Y : TNcomplex);
const
  freq = 60.0;
  sig1 = 1.e-3;
  sig2 = 1.e-6;
  eps1 = 9.7394e-11;
  eps2 = 2.6562e-11;
  mu   = 12.5664e-7;
  a = 15.0;
  d = 14.9;
var
  omega: extended;
  dum,dum1,dum2,kzsqr: TNcomplex;
  cnst,cnstsqr,konst,konstsqr,C1,C2: TNcomplex;
  dcomplex,acomplex,a_minus_d,arg1,arg2: TNcomplex;
  tan1,tan2,term1,term2: TNcomplex;
  original,root1,rootfactor1,root2,rootfactor2,Yinter: TNcomplex;
begin
  omega:= 2*Pi*freq;

  dum1.Re:= sig1;       { for k1sqr }
  dum1.Im:= omega*eps1;

  dum2.Re:= sig2;       { for k2sqr }
  dum2.Im:= omega*eps2;

  dum.Re:= 0;          { for both k1sqr and k2sqr }
  dum.Im:= -omega*mu;

  Mult(dum1,dum,k1sqr);
  Mult(dum2,dum,k2sqr);

  complexSqr(kz,kzsqr);

  Sub(k1sqr,kzsqr,cnstsqr);
  SquareRoot(cnstsqr,cnst);

  Sub(k2sqr,kzsqr,konstsqr);
  SquareRoot(konstsqr,konst);

  Divide(cnst,dum1,C1);
  Divide(konst,dum2,C2);

  dcomplex.Re:= d; dcomplex.Im:= 0;
  acomplex.Re:= a; acomplex.Im:= 0;

  Sub(acomplex,dcomplex,a_minus_d);

  Mult(cnst,dcomplex,arg1);
  Mult(konst,a_minus_d,arg2);

  complexTAN(arg1,tan1); complexTAN(arg2,tan2);

  Mult(C1,tan1,term1); Mult(C2,tan2,term2);

  Add(term1,term2,Y);
```

```
{ FACTORIZATION OF CALCULATED ROOTS FROM THE FUNCTION (disabled) }

 {root1.Re:=1.625410e-1;
 root1.Im:= -4.399160e-2;

 Sub(kz,root1,rootfactor1);

 root2.Re:= -1.625410e-1;
 root2.Im:= 4.399160e-2;

 Sub(kz,root2,rootfactor2);

 Divide(original,rootfactor1,Yinter);

 Divide(Yinter,rootfactor2,Y);}

end; { procedure TNTargetF }

{------------------------------------------------------------------------}
{ ENTER THE USER INPUT VARIABLES                                         }
{------------------------------------------------------------------------}

procedure UserInput(var Guess   : TNcomplex;
              var Tol     : Extended;
              var MaxIter : integer);

{------------------------------------------------------}
{- Output: Guess, Tol, MaxIter                        -}
{-                                     -}
{- This procedure assigns values to the above variables. -}
{- The initial approximation of the guess (Guess) is   -}
{- enetered from keyboard input. Tolerance (Tol) is    -}
{- automatically set to 10**-8 and the maximum number   -}
{- of iterations (MaxIter) is set to 100              -}
{------------------------------------------------------}

procedure GetInitialGuess(var Guess : TNcomplex);
var
  Answer: TNcomplex;
begin
  Writeln;
  Writeln('Initial approximation to the root: ');
  repeat
    Write('Re(Approximation) = ');
    Readln(Guess.Re);
    IOCheck;
  until not IOerr;
  repeat
    Write('Im(Approximation) = ');
```

```
   Readln(Guess.Im);
   IOCheck;
  until not IOerr;

  TNTargetF(Guess,Answer);
  Writeln('f(initial guess) = ',Answer.Re:23,' + ',Answer.Im:23,'i');

 end; { procedure GetInitialGuess }

 procedure GetTolerance(var Tol : Extended);
 begin
  Tol := 1E-8;
 end; { procedure GetTolerance }

 procedure GetMaxIter(var MaxIter : integer);
 begin
  MaxIter := 100;
 end; { procedure GetMaxIter }

 begin { procedure UserInput }
  GetInitialGuess(Guess);
  GetTolerance(Tol);
  GetMaxIter(MaxIter);
  GetOutputFile(OutFile);
 end; { procedure UserInput }


{--------------------------------     ---------------------------------------}
{ OUTPUT THE RESULTS TO THE DEVICE OUTFILE                          }
{---------------------------------------------------------------------------}


procedure Results(Guess   : TNcomplex;
                  Answer  : TNcomplex;
                  yAnswer : TNcomplex;
                  Tol     : Extended;
                  MaxIter : integer;
                  Iter    : Integer;
                  Error   : byte);


 begin
   Writeln(OutFile);
   Write(OutFile, 'Initial approximation: ' : 30);
   Writeln(OutFile, Guess.Re:23, ' +', Guess.Im:23, 'i');
   Writeln(OutFile, 'Tolerance: ' : 30, Tol:23);
   Writeln(OutFile, 'Maximum number of iterations: ' : 30, MaxIter:2);
   Writeln(OutFile);
   if Error in [1, 2] then
  DisplayWarning;
if Error >= 3 then
  DisplayError;

case Error of
  1 : Writeln(OutFile, 'This will take more than ', MaxIter, ' iterations.');

  2 : begin
      Writeln(OutFile, 'A parabola which intersects the x-axis can not');
      Writeln(OutFile, 'be constructed through these three points.');
     end;

  3 : Writeln(OutFile, 'The tolerance must be greater than zero.');

  4 : Writeln(OutFile,
         'The maximum number of iterations must be greater than zero.');
end; { case }

{ CALCULATE KX1 AND KX2 FROM KZ }

if Error <= 2 then
begin
   ComplexSqr(Answer,Asqr);
   Sub(k1sqr,Asqr,kx1sqr);
   Sub(k2sqr,Asqr,kx2sqr);
   SquareRoot(kx1sqr,kx1);
   SquareRoot(kx2sqr,kx2);
   if kx1.Re < 0.0 then
     begin
       kx1.Re:= -kx1.Re;
       kx1.Im:= -kx1.Im
     end;
   if kx2.Re < 0.0 then
     begin
       kx2.Re:= -kx2.Re;
       kx2.Im:= -kx2.Im
     end;
  Writeln(OutFile);
  Writeln(OutFile, 'Number of iterations: ' : 26, Iter:2);
  Write(OutFile, 'Calculated root (kz): ' : 26);
  Writeln(OutFile, Answer.Re:23, ' + ', Answer.Im:23, ' i');
   Write(OutFile, '    kx1      : ' : 26);
   Writeln(OutFile, kx1.Re:23, ' + ', kx1.Im:23, ' i');
   Write(OutFile, '    kx2      : ' : 26);
   Writeln(OutFile, kx2.Re:23, ' + ', kx2.Im:23, ' i');
  Writeln(OutFile, 'Value of the function  ' : 26);
  Write(OutFile, 'at the calculated root: ' : 26);
  Writeln(OutFile, yAnswer.Re:23, ' + ', yAnswer.Im:23, ' i');
  Writeln(OutFile);
  end;
end; { procedure Results }

{-----------------------------------------------------------------------------}
{ MAIN PROGRAM                                           }
{-----------------------------------------------------------------------------}

begin { program Muller }

{ RETRIEVE THE INPUT DATA }
  UserInput(Guess, Tol, MaxIter);

{ USE MULLER'S METHOD WITH DEFLATION TO FIND A ROOT }
  Muller(Guess, Tol, MaxIter, Answer, yAnswer, Iter, Error, @TNTargetF);

{ OUTPUT THE RESULTS }
  Results(Guess, Answer, yAnswer, Tol, MaxIter, Iter, Error);


  Close(OutFile);
  WaitReturnOrClick;
 end. { program Muller }
```

```
{ Program 3. FillingMuller                              }


program Muller;

{-------------------------------------------------------------------------}
{                                                 }
{                                                 }
{-  This program is a modified version of the original copyrighted program -}
{-  'Muller' (TurboPascal Numerical Methods Toolbox (c) 1987 Borland       -}
{-  International). It calculates higher order mode branch solutions      -}
{-  of the transcendental function provided one solution belonging to the  -}
{-  higher order mode set is known. The solution subset at frequencies     -}
{-  above the known point are calculated by incrementing the frequency     -}
{-  and using initial guesses based on the exact roots found at previous   -}
{-  frequencies. Similarly, the solution sub-set for frequencies below     -}
{-  the known point are found by decrementing the frequency.               -}
{-                                                -}
{-  This program is the driver program for the Muller rootfinding        -}
{-  subroutine. The transcendental function, the variable declarations    -}
{-  and all the I/O statements are all taken care of here. The actual     -}
{-  Muller subroutine is called 'RootsOfEquat' and is listed as program   -}
{-  5 in this appendix. All the information concerning the transcendental -}
{-  equation and the system parameters is contained in the body of the     -}
{-  text of the accompanying thesis.                          -}
{-                                                -}
{-  Program input consists of the physical system parameters entered in    -}
{-  the section titled 'ENTRY OF THE PHYSICAL SYSTEM PARAMETERS.' The      -}
{-  known point (kz), its frequency (starting frequency), the frequency    -}
{-  increment and the ending frequency are all prompted for when the       -}
{-  program is run.                                     -}
{-                                                -}
{-  Program output is given in 2 files. The file called 'outfile' is      -}
{-  displayed on the screen and shows the frequency, the calculated       -}
{-  roots kx1, kx2 and kz and the value of the transcendental function     -}
{-  calculated at the root. The file called 'plotfile' contains frequency  -}
{-  and the 3 propagation constants in rectangular form and is formatted   -}
{-  to be directly read by plotting routines such as 'Cricketgraph.'      -}
{-                                                -}
{-  Copyright (c) 1990 Richard Maslowski                          -}
{-  All Rights Reserved                                  -}

{-                                                -}
{-  Units used:  IOSelection        var OutFile : text;              -}
{-                              OutName : string;           -}
{-                              InFile  : text;          -}
{-                              InName  : string;           -}
{-                              IOerr   : boolean;          -}
{-                            procedure DisplayWarning        -}
{-                      procedure DisplayError        -}
{-                      procedure IOCheck             -}
{-                      procedure GetInputFile        -}
{-                      procedure GetOutputFile       -}
{-                                          -}
{-          RootsOfEquat     procedure Muller            -}
{-                                          -}
{- Note: One condition statement in the main program may have to be altered-}
{- depending on whether frequency steps are + or -.              -}
{-                                          -}
{--------------------------------------------------------------------------}

{$I-}       { Disable I/O error trapping }
{$R+}       { Enable range checking }
{$S+}       { Enable segmentation of code }

{$R IOSelection.rsrc} { Resource file for IOSelection unit }
{$U IOSelection}
{$U RootsOfEquat}

uses
 MemTypes, QuickDraw, OSIntf, ToolIntf, PackIntf, PasPrinter,
 RootsOfEquat,

{$S SecondSegment}

 IOSelection;

 {-------------------------------------------------------------------------}
 { ENTRY OF THE PHYSICAL SYSTEM PARAMETERS                        }
 {-------------------------------------------------------------------------}

const
{ enter the saturated region conductivity (S/m) }
 sig1 = 0;

{ enter the depleted region conductivity (S/m) }
 sig2 = 0;

{ enter the saturated region dielectric constant (F/m) }
 eps1 = 9.7394e-11; { normally eps1 = 9.7394e-11 }

{ enter the depleted region dielectric constant (F/m) }
 eps2 = 2.6562e-11;

{ enter the magnetic permeability for both regions (H/m)
 mu  = 12.5664e-7;

{ enter the parallel-plate separation (m) }
```

```
 a   = 15;

{ enter the saturated region thickness (m) }
 d   = 12.75;

{ VARIABLE DECLARATIONS }

var
 Guess : TNcomplex;            { Initial approximations }
 Tol : Extended;              { Tolerance in answer }
 Iter : integer;              { Number of iterations }
 MaxIter : integer;             { Max number of iterations allowed }
 Endfreq : Extended             { Ending frequency }
 Stepfreq : Extended            { Frequency step }
 Answer, yAnswer : TNcomplex;      { Root and function evaluated at root }
 Error : byte;                { Flags an error }
 k1sqr,k2sqr: TNcomplex;
 dum,dum1,dum2: TNcomplex;
 Asqr,kx1sqr,kx2sqr,kx1,kx2: TNcomplex;
 freq,omega: Extended;
 PlotFile: text;
 SmallFile: text;
 E_File: text;
 a_over_d,smallnum,a_m_d_over_d,dumratio,multdumratio,one: TNcomplex;
 smalldenom,smallkzsqr,smallkz: TNcomplex;
 dcomp,tanarg,Etan,coef,Eratio: TNcomplex;
 Emag: extended;

{ HERE ARE SOME NECESSARY COMPLEX PROCEDURES }


procedure Conjugate(C1 : TNcomplex; var C2 : TNcomplex);
begin
 C2.Re := C1.Re;
 C2.Im := -C1.Im;
end; { procedure Conjugate }

procedure Real2Complex(realnumber: extended; var complexnumber: TNcomplex);
begin
 complexnumber.Re:= realnumber;
 complexnumber.Im:= 0;
end; { procedure Real2Complex }

function Modulus(var C1 : TNcomplex) : Extended;
begin
 Modulus := Sqrt(Sqr(C1.Re) + Sqr(C1.Im));
end; { function Modulus }

procedure Add(C1, C2 : TNcomplex; var C3 : TNcomplex);
```

```
begin
  C3.Re := C1.Re + C2.Re;
  C3.Im := C1.Im + C2.Im;
end; { procedure Add }

procedure Sub(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
  C3.Re := C1.Re - C2.Re;
  C3.Im := C1.Im - C2.Im;
end; { procedure Sub }

procedure Mult(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
  C3.Re := C1.Re * C2.Re - C1.Im * C2.Im;
  C3.Im := C1.Im * C2.Re + C1.Re * C2.Im;
end; { procedure Mult }

procedure Divide(C1, C2 : TNcomplex; var C3 : TNcomplex);
var
  Dum1, Dum2 : TNcomplex;
  E : Extended;
begin
  Conjugate(C2, Dum1);
  Mult(C1, Dum1, Dum2);
  E := Sqr(Modulus(C2));
  C3.Re := Dum2.Re / E;
  C3.Im := Dum2.Im / E;
end; { procedure Divide }

procedure SquareRoot(C1 : TNcomplex; var C2 : TNcomplex);
var
  R, Theta : Extended;
begin
  R := Sqrt(Sqr(C1.Re) + Sqr(C1.Im));
  if ABS(C1.Re) < TNNearlyZero then
    begin
      if C1.Im < 0 then
        Theta := Pi / 2
      else
        Theta := -Pi / 2;
    end
  else
    if C1.Re < 0 then
      Theta := ArcTan(C1.Im / C1.Re) + Pi
    else
      Theta := ArcTan(C1.Im / C1.Re);
  C2.Re := Sqrt(R) * Cos(Theta / 2);
  C2.Im := Sqrt(R) * Sin(Theta / 2);
end; { procedure SquareRoot }
```

```
procedure ComplexSIN(C1: TNcomplex; var C2: TNcomplex);
begin
 C2.Re:=  SIN(C1.Re) * (EXP(-C1.Im) + EXP(C1.Im))/2;
 C2.Im:= -COS(C1.Re) * (EXP(-C1.Im) - EXP(C1.Im))/2
end; { procedure ComplexSIN }

procedure ComplexCOS(C1: TNcomplex; var C2: TNcomplex);
begin
 C2.Re:= COS(C1.Re) * (EXP(-C1.Im) + EXP(C1.Im))/2;
 C2.Im:= SIN(C1.Re) * (EXP(-C1.Im) - EXP(C1.Im))/2
end; { procedure ComplexCOS }

procedure ComplexTAN(C1: TNcomplex; var C2: TNcomplex);
var
 Dum1,Dum2: TNcomplex;
begin
 ComplexSIN(C1,Dum1);
 ComplexCOS(C1,Dum2);
 Divide(Dum1,Dum2,C2);
end; { procedure ComplexTAN }

procedure ComplexEXP(C1: TNcomplex; var C2: TNcomplex);
begin
 C2.Re:= EXP(C1.Re) * COS(C1.Im);
 C2.Im:= EXP(C1.Re) * SIN(C1.Im)
end; { procedure ComplexEXP }

procedure ComplexSqr(C1: TNcomplex; var C2: TNcomplex);
begin
 C2.Re:= SQR(C1.Re) - SQR(C1.Im);
 C2.Im:= 2 * C1.Re * C1.Im
end; { procedure ComplexSqr }


{-----------------------------------------------------------------------------}
{ HERE IS THE FUNCTION IN THE FORM F(KZ)=0                             }
{-----------------------------------------------------------------------------}

procedure TNTargetF(kz : TNcomplex; var Y : TNcomplex);

var
 kzsqr: TNcomplex;
 cnst,cnstsqr,konst,konstsqr,C1,C2: TNcomplex;
 dcomplex,acomplex,a_minus_d,arg1,arg2: TNcomplex;
 tan1,tan2,term1,term2: TNcomplex;
 original,root1,rootfactor1,root2,rootfactor2,Yinter: TNcomplex;

begin
```

```
complexSqr(kz,kzsqr);

Sub(k1sqr,kzsqr,cnstsqr);
SquareRoot(cnstsqr,cnst);

Sub(k2sqr,kzsqr,konstsqr);
SquareRoot(konstsqr,konst);

Divide(cnst,dum1,C1);
Divide(konst,dum2,C2);

dcomplex.Re:= d; dcomplex.Im:= 0;
acomplex.Re:= a; acomplex.Im:= 0;

Sub(acomplex,dcomplex,a_minus_d);

Mult(cnst,dcomplex,arg1);
Mult(konst,a_minus_d,arg2);

complexTAN(arg1,tan1); complexTAN(arg2,tan2);

Mult(C1,tan1,term1); Mult(C2,tan2,term2);

Add(term1,term2,Y);

{ FACTORIZATION OF THE CALCULATED ROOTS (disabled) }

{root1.Re:= 1.639964e-2;
root1.Im:= -2.518189e-3;

Sub(kz,root1,rootfactor1);

root2.Re:= -1.639964e-2;
root2.Im:= 2.518189e-3;

Sub(kz,root2,rootfactor2);

Divide(original,rootfactor1,Yinter);

Divide(Yinter,rootfactor2,Y);}

end; { procedure TNTargetF }

{-----------------------------------------------------------------------}
{ ENTER THE USER INPUT VARIABLES                                        }
{-----------------------------------------------------------------------}

procedure GetTolerance(var Tol : Extended);
```

```
begin
 Tol := 1E-8;
 Writeln;
 repeat
   Write('Tolerance (> 0): ');
   Readln(Tol);
   IOCheck;
   if Tol <= 0 then
   begin
     IOerr := true;
     Tol := 1E-8;
   end;
 until not IOerr;
end; { procedure GetTolerance }

procedure GetMaxIter(var MaxIter : integer);
begin
 MaxIter := 100;
 Writeln;
 repeat
   Write('Maximum number of iterations (> 0): ');
   Readln(MaxIter);
   IOCheck;
   if MaxIter < 0 then
   begin
     IOerr := true;
     MaxIter := 100;
   end;
 until not IOerr;
end; { procedure GetMaxIter }

procedure GetEndfreq(var Tol : Extended);
begin
 Endfreq := 2E+6;
 Writeln;
 repeat
   Write('Ending frequency (Hz): ');
   Readln(Endfreq);
   IOCheck;
   if Endfreq <= 0 then
   begin
     IOerr := true;
     Endfreq := 2E+6;
   end;
 until not IOerr;
end; { procedure GetEndfreq }

procedure GetStepfreq(var Tol : Extended);
begin
```

```
 Stepfreq := 1E+4;
 Writeln;
 repeat
  Write('frequency stepsize  (+ or - in Hz): ');
  Readln(Stepfreq);
  IOCheck;
  if Stepfreq <= 0 then
  begin
   IOerr := true;
   Stepfreq := 1E+4;
  end;
 until not IOerr;
end; { procedure GetStepfreq }


{ CALCULATE KX1 AND KX2 FROM KZ }

procedure Calculatekx(inputvalue: TNcomplex);
begin
    ComplexSqr(inputvalue,Asqr);
    Sub(k1sqr,Asqr,kx1sqr);
    Sub(k2sqr,Asqr,kx2sqr);
    SquareRoot(kx1sqr,kx1);
    SquareRoot(kx2sqr,kx2);
    if kx1.Re < 0.0 then
      begin
        kx1.Re:= -kx1.Re;
        kx1.Im:= -kx1.Im
      end;
    if kx2.Re < 0.0 then
      begin
        kx2.Re:= -kx2.Re;
        kx2.Im:= -kx2.Im
      end;
end;  { procedure Calculatekx }

procedure UserInput(var Guess   : TNcomplex;
              var Tol    : Extended;
              var MaxIter : integer);

{-------------------------------------------------------}
{- Output: Guess, Tol, MaxIter                      -}
{-                                   -}
{- This procedure assigns values to the above variables  -}
{- from keyboard input.  The initial approximation of the -}
{- guess (Guess), tolerance (Tol), and maximum number of  -}
{- iterations (MaxIter) are all input here.          -}
{-------------------------------------------------------}
```

```
procedure GetInitialGuess(var Guess : TNcomplex);
var
  Answer: TNcomplex;
begin
  Writeln;
  Writeln('Initial approximation to the root:  ');
  repeat
    Write('Re(Approximation) = ');
    Readln(Guess.Re);
    IOCheck;
  until not IOerr;
  repeat
    Write('Im(Approximation) = ');
    Readln(Guess.Im);
    IOCheck;
  until not IOerr;

  TNTargetF(Guess,Answer);
  Writeln('f(initial guess) = ',Answer.Re:23,' + ',Answer.Im:23,'i');

end; { procedure GetInitialGuess }


begin { procedure UserInput }
  GetInitialGuess(Guess);
end; { procedure UserInput }

{--------------------------------------------------------------------------}
{  OUTPUT THE RESULTS TO THE DEVICE OUTFILE                              }
{--------------------------------------------------------------------------}

procedure Results(Guess   : TNcomplex;
                  Answer  : TNcomplex;
                  yAnswer : TNcomplex;
                  Tol     : Extended;
                  MaxIter : integer;
                  Iter    : integer;
                  Error   : byte);


begin
  Writeln(OutFile);
  Write(OutFile, 'frequency: ' : 30);
  Writeln(OutFile, freq/1000:7:2, ' kHz');
  Writeln(OutFile);
  if Error in [1, 2] then
    DisplayWarning;
  if Error >= 3 then
    DisplayError;
```

```
case Error of
  1 : Writeln(OutFile, 'This will take more than ', MaxIter, ' iterations.');

  2 : begin
      Writeln(OutFile, 'A parabola which intersects the x-axis can not');
      Writeln(OutFile, 'be constructed through these three points.');
    end;

  3 : Writeln(OutFile, 'The tolerance must be greater than zero.');

  4 : Writeln(OutFile,
          'The maximum number of iterations must be greater than zero.');
end; { case }


if Error <= 2 then
begin
  Calculatekx(Answer);
   {Real2Complex(d,dcomp);  calculate the ratio of Ez over Ex1
   Mult(kx1,dcomp,tanarg);
   ComplexTAN(tanarg,Etan);
   Divide(kx1,Answer,coef);
   Mult(coef,Etan,Eratio);
   Emag:= Modulus(Eratio);}
  Writeln(OutFile);
  Writeln(OutFile, 'Number of iterations: ' : 26, Iter:2);
  Write(OutFile, 'Calculated root (kz): ' : 26);
  Writeln(OutFile, Answer.Re:23, ' + ', Answer.Im:23, ' i');
   Write(OutFile, '    kx1     : ' : 26);
   Writeln(OutFile, kx1.Re:23, ' + ', kx1.Im:23, ' i');
   Write(OutFile, '    kx2     : ' : 26);
   Writeln(OutFile, kx2.Re:23, ' + ', kx2.Im:23, ' i');
  Writeln(OutFile, 'Value of the function  ' : 26);
  Write(OutFile, 'at the calculated root: ' : 26);
  Writeln(OutFile, yAnswer.Re:23, ' + ', yAnswer.Im:23, ' i');
  Writeln(OutFile);
   Writeln(PlotFile,freq/1000:7:2,Chr(9),Answer.Re:23,Chr(9),Answer.Im:23,
     Chr(9),kx1.Re:23,Chr(9),kx1.Im:23,Chr(9),kx2.Re:23,Chr(9),kx2.Im:23);
 end;
end; { procedure Results }

{-----------------------------------------------------------------------}
{ MAIN PROGRAM                                           }
{-----------------------------------------------------------------------}

begin { program Muller }
  GetTolerance(Tol);
  GetMaxIter(MaxIter);
 GetOutputFile(OutFile);
 GetOutputFile(PlotFile);
 GetEndfreq(Endfreq);
 GetStepfreq(Stepfreq);

 repeat
   Writeln;Writeln;
   Write('Starting frequency (Hz) = ');
   Readln(freq);
   IOCheck;
 until not IOerr;

 repeat
   Writeln;Writeln;
   Writeln('Starting guess value');
   Write('    real part = ');
   Readln(Answer.Re);
   Write('    imaginary part = ');
   Readln(Answer.Im);
   IOCheck;
 until not IOerr;

 repeat
   Writeln;Writeln;
   Writeln('now working on frequency = ', freq/1000:7:1 , ' kHz');

{ CALCULATE THE SMALL ARGUMENT PROPAGATION CONSTANTS }

     omega:= 2*Pi*freq;

     dum1.Re:= sig1;      { for k1sqr }
     dum1.Im:= omega*eps1;

     dum2.Re:= sig2;      { for k2sqr }
     dum2.Im:= omega*eps2;

     dum.Re:= 0;          { for both k1sqr and k2sqr }
     dum.Im:= -omega*mu;

     Mult(dum1,dum,k1sqr);
     Mult(dum2,dum,k2sqr);

     {Mult(k1sqr,a_over_d,smallnum);  the following calculates small kz
     Divide(dum1,dum2,dumratio);
     Mult(dumratio,a_m_d_over_d,multdumratio);
     Add(one,multdumratio,smalldenom);
     Divide(smallnum,smalldenom,smallkzsqr);
     SquareRoot(smallkzsqr,smallkz);
      if smallkz.Re < 0 then
      begin
       smallkz.Re:= -smallkz.Re;
       smallkz.Im:= -smallkz.Im
     end;
    Calculatekx(smallkz);
  Writeln(SmallFile,freq/1000:7:1,Chr(9),smallkz.Re:23,Chr(9),smallkz.Im:23,
   Chr(9),kx1.Re:23,Chr(9),kx1.Im:23,Chr(9),kx2.Re:23,Chr(9),kx2.Im:23);}

 Guess:= Answer;

 { USE MULLER'S METHOD WITH DEFLATION TO FIND A ROOT }
 Muller(Guess, Tol, MaxIter, Answer, yAnswer, Iter, Error, @TNTargetF);

 { OUTPUT THE RESULTS }
 Results(Guess, Answer, yAnswer, Tol, MaxIter, Iter, Error);

   freq:= freq + Stepfreq;

  until freq > Endfreq;  { > or < depending on whether Stepfreq = + or - }


 Close(OutFile);
 Close(PlotFile);
 WaitReturnOrClick;
end. { program Muller }
```

```
{ Program 4. ERatioMuller                              }


program Muller;

{-----------------------------------------------------------------------}
{-                                             -}
{-                                             -}
{-  This program is a modified version of the original copyrighted program -}
{-  'Muller' (Turbo Pascal Numerical Methods Toolbox (c) 1987, Borland    -}
{-  International) It calculates roots of the transcendental equation    -}
{-  describing the propagation constants for the parallel-plate waveguide  -}
{-  under study.This program is essentially similar to FillingMuller      -}
{-  except that the ratio of axial to transverse electric field magnitudes,-}
{-  as measured at the media interface in the saturated region, are       -}
{-  calculated at each frequency after calculation of the propagation     -}
{-  constants.                                       -}
{-                                             -}
{-  This program is the driver program for the Muller rootfinding         -}
{-  subroutine. The transcendental function, the variable declarations    -}
{-  and all the I/O statements are all taken care of here. The actual     -}
{-  Muller subroutine is called 'RootsOfEquat' and is listed as program 5  -}
{-  in this appendix. All the information concerning the transcendental   -}
{-  equation and the system parameters is contained in the body of the    -}
{-  text of the accompanying thesis.                         -}
{-                                             -}
{-  Program input consists of the physical system parameters entered in   -}
{-  the section titled 'ENTRY OF THE PHYSICAL SYSTEM PARAMETERS.' The      -}
{-  known point (kz), its frequency (starting frequency), the frequency    -}
{-  increment and the ending frequency are all prompted for when the      -}
{-  program is run.                                   -}
{-                                             -}
{-  Program output is given in 2 files. The file called 'outfile' is      -}
{-  displayed on the screen and shows the frequency, the calculated       -}
{-  roots kx1, kx2 and kz and the value of the transcendental function     -}
{-  calculated at the root. The file called 'plotfile' contains frequency, -}
{-  the transverse coordinate x, and the ratio of the E-field magnitudes   -}
{-  at that coordinate formatted to be directly read by plotting routines  -}
{-  such as 'Cricketgraph.'                          -}                       -}
{-                                             -}
{-  Copyright (c) 1990 Richard Maslowski                      -}
{-  All Rights Reserved                               -}
{-                                             -}
{-  Units used:  IOSelection        var OutFile : text;            -}
{-                           OutName : string;          -}
{-                           InFile  : text;        -}
{-                           InName  : string;         -}
{-                           IOerr   : boolean;        -}
{-                         procedure DisplayWarning         -}
{-                    procedure DisplayError        -}
{-                    procedure IOCheck             -}
{-                    procedure GetInputFile        -}
{-                    procedure GetOutputFile       -}
{-                                          -}
{-          RootsOfEquat    procedure Muller            -}
{-                                          -}
{-                                          -}
{----------------------------------------------------------------------------}

{$I-}       { Disable I/O error trapping }
{$R+}       { Enable range checking }
{$S+}       { Enable segmentation of code }

{$R IOSelection.rsrc} { Resource file for IOSelection unit }
{$U IOSelection}
{$U RootsOfEquat}

uses
 MemTypes, QuickDraw, OSIntf, ToolIntf, PackIntf, PasPrinter,
 RootsOfEquat,

{$S SecondSegment}

 IOSelection;


{ VARIABLE DECLARATIONS }

var
 Guess : TNcomplex;            { Initial approximations }
 Tol : Extended;             { Tolerance in answer }
 Iter : integer;            { Number of iterations }
 MaxIter : integer;             { Max number of iterations allowed }
 Endfreq : Extended;           { Ending frequency }
 Stepfreq : Extended;           { Frequency stepsize }
 Answer, yAnswer : TNcomplex;    { Root and function evaluated at root }
 Error : byte;               { Flags an error }
 k1sqr,k2sqr: TNcomplex;
 dum,dum1,dum2: TNcomplex;
 Asqr,kx1sqr,kx2sqr,kx1,kx2: TNcomplex;
 freq,omega: Extended;
 PlotFile: text;
 Ex1,Ez1,Ex2,Ez2: TNcomplex;

{  SOME NECESSARY COMPLEX PROCEDURES  }

procedure Conjugate(C1 : TNcomplex; var C2 : TNcomplex);
begin
```

```
  C2.Re := C1.Re;
  C2.Im := -C1.Im;
end; { procedure Conjugate }

procedure CMPLX(realpart,imagpart: extended; var cmplxnum: TNcomplex);
begin
  cmplxnum.Re:= realpart;
  cmplxnum.Im:= imagpart;
end;  { procedure CMPLX }

function Modulus(var C1 : TNcomplex) : Extended;
begin
  Modulus := Sqrt(Sqr(C1.Re) + Sqr(C1.Im));
end; { function Modulus }

function Phase(var C1: TNcomplex): Extended;
begin
  if C1.Re = 0 then Phase:= -90 else
    Phase:= arctan(C1.Im/C1.Re)*180/pi;
end;  { function Phase }

procedure Add(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
  C3.Re := C1.Re + C2.Re;
  C3.Im := C1.Im + C2.Im;
end; { procedure Add }

procedure Sub(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
  C3.Re := C1.Re - C2.Re;
  C3.Im := C1.Im - C2.Im;
end; { procedure Sub }

procedure Mult(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
  C3.Re := C1.Re * C2.Re - C1.Im * C2.Im;
  C3.Im := C1.Im * C2.Re + C1.Re * C2.Im;
end; { procedure Mult }

procedure Divide(C1, C2 : TNcomplex; var C3 : TNcomplex);
var
  Dum1, Dum2 : TNcomplex;
  E : Extended;
begin
  Conjugate(C2, Dum1);
  Mult(C1, Dum1, Dum2);
  E := Sqr(Modulus(C2));
  C3.Re := Dum2.Re / E;
  C3.Im := Dum2.Im / E;
```

```
end; { procedure Divide }

procedure SquareRoot(C1 : TNcomplex; var C2 : TNcomplex);
var
 R, Theta : Extended;
begin
  R := Sqrt(Sqr(C1.Re) + Sqr(C1.Im));
  if ABS(C1.Re) < TNNearlyZero then
    begin
      if C1.Im < 0 then
        Theta := Pi / 2
      else
        Theta := -Pi / 2;
    end
  else
    if C1.Re < 0 then
      Theta := ArcTan(C1.Im / C1.Re) + Pi
    else
      Theta := ArcTan(C1.Im / C1.Re);
  C2.Re := Sqrt(R) * Cos(Theta / 2);
  C2.Im := Sqrt(R) * Sin(Theta / 2);
end; { procedure SquareRoot }

procedure ComplexSIN(C1: TNcomplex; var C2: TNcomplex);
begin
  C2.Re:=  SIN(C1.Re) * (EXP(-C1.Im) + EXP(C1.Im))/2;
  C2.Im:= -COS(C1.Re) * (EXP(-C1.Im) - EXP(C1.Im))/2
end;  { procedure ComplexSIN }

procedure ComplexCOS(C1: TNcomplex; var C2: TNcomplex);
begin
  C2.Re:= COS(C1.Re) * (EXP(-C1.Im) + EXP(C1.Im))/2;
  C2.Im:= SIN(C1.Re) * (EXP(-C1.Im) - EXP(C1.Im))/2
end;  { procedure ComplexCOS }

procedure ComplexTAN(C1: TNcomplex; var C2: TNcomplex);
var
  Dum1,Dum2: TNcomplex;
begin
  ComplexSIN(C1,Dum1);
  ComplexCOS(C1,Dum2);
  Divide(Dum1,Dum2,C2);
end;  { procedure ComplexTAN }

procedure ComplexEXP(C1: TNcomplex; var C2: TNcomplex);
begin
  C2.Re:= EXP(C1.Re) * COS(C1.Im);
  C2.Im:= EXP(C1.Re) * SIN(C1.Im)
end;  { procedure ComplexEXP }
```

```
procedure ComplexSqr(C1: TNcomplex; var C2: TNcomplex);
begin
 C2.Re:= SQR(C1.Re) - SQR(C1.Im);
 C2.Im:= 2 * C1.Re * C1.Im
end; { procedure ComplexSqr }


{------------------------------------------------------------------------------}
{ HERE IS THE FUNCTION F(kz)=0                                }
{------------------------------------------------------------------------------}

procedure TNTargetF(kz : TNcomplex; var Y : TNcomplex);

 {---------------------------------------------------------------------------}
 { ENTRY OF THE PHYSICAL SYSTEM PARAMETERS                          }
 {---------------------------------------------------------------------------}

 const
 { enter the saturated region conductivity (S/m) }
  sig1 = 1.e-3;

 { enter the depleted region conductivity (S/m) }
  sig2 = 1.e-6;

 { enter the saturated region dielectric constant (F/m) }
  eps1 = 9.7394e-11; { normally eps1 = 9.7394e-11 }

 { enter the depleted region dielectric constant (F/m) }
  eps2 = 2.6562e-11;

 { enter the magnetic permeability for both regions (H/m)
  mu = 12.5664e-7;

 { enter the parallel-plate separation (m) }
  a = 15.0;

 { enter the saturated region thickness (m) }
  d = 15.0;

 var
  kzsqr: TNcomplex;
  cnst,cnstsqr,konst,konstsqr,C1,C2: TNcomplex;
  dcomplex,acomplex,a_minus_d,arg1,arg2: TNcomplex;
  tan1,tan2,term1,term2: TNcomplex;
  original,root1,rootfactor1,root2,rootfactor2,Yinter: TNcomplex;

 begin
 complexSqr(kz,kzsqr);

 Sub(k1sqr,kzsqr,cnstsqr);
 SquareRoot(cnstsqr,cnst);

 Sub(k2sqr,kzsqr,konstsqr);
 SquareRoot(konstsqr,konst);

 Divide(cnst,dum1,C1);
 Divide(konst,dum2,C2);

 dcomplex.Re:= d; dcomplex.Im:= 0;
 acomplex.Re:= a; acomplex.Im:= 0;

 Sub(acomplex,dcomplex,a_minus_d);

 Mult(cnst,dcomplex,arg1);
 Mult(konst,a_minus_d,arg2);

 complexTAN(arg1,tan1); complexTAN(arg2,tan2);

 Mult(C1,tan1,term1); Mult(C2,tan2,term2);

 Add(term1,term2,Y);

{ FACTORIZATION OF CALCULATED ROOTS FROM THE FUNCTION (disabled) }

 {root1.Re:[illegible]e-2;
 root1.Im:= -2.[illegible];

 Sub(kz,root1,rootfactor1);

 root2.Re:= -1.639964e-2;
 root2.Im:= 2.518189e-3;

 Sub(kz,root2,rootfactor2);

 Divide(original,rootfactor1,Yinter);

 Divide(Yinter,rootfactor2,Y);}

end; { procedure TNTargetF }

{-----------------------------------------------------------------------------}
{ INPUT TOLERANCE, MAX. ITERATIONS, ENDING FREQUENCY AND STEPSIZE          }
{-----------------------------------------------------------------------------}

procedure GetTolerance(var Tol : Extended);
begin
```

```
 Tol := 1E-8;
 Writeln;
 repeat
  Write('Tolerance (> 0): ');
  Readln(Tol);
  IOCheck;
  if Tol <= 0 then
  begin
   IOerr := true;
   Tol := 1E-8;
  end;
 until not IOerr;
end; { procedure GetTolerance }

procedure GetMaxIter(var MaxIter : integer);
begin
 MaxIter := 100;
 Writeln;
 repeat
  Write('Maximum number of iterations (> 0): ');
  Readln(MaxIter);
  IOCheck;
  if MaxIter < 0 then
  begin
   IOerr := true;
   MaxIter := 100;
  end;
 until not IOerr;
end; { procedure GetMaxIter }

procedure GetEndfreq(var Tol : Extended);
begin
 Endfreq := 2E+6;
 Writeln;
 repeat
  Write('Ending frequency (Hz): ');
  Readln(Endfreq);
  IOCheck;
  if Endfreq <= 0 then
  begin
   IOerr := true;
   Endfreq := 2E+6;
  end;
 until not IOerr;
end; { procedure GetEndfreq }

procedure GetStepfreq(var Tol : Extended);
begin
 Stepfreq := 1E+4;
 Writeln;
 repeat
  Write('frequency stepsize  (+ or - in Hz): ');
  Readln(Stepfreq);
  IOCheck;
  if Stepfreq <= 0 then
  begin
   IOerr := true;
   Stepfreq := 1E+4;
  end;
 until not IOerr;
end; { procedure GetStepfreq }



{ CALCULATE KX1 AND KX2 FROM KZ }


procedure Calculatekx(inputvalue: TNcomplex);
begin
    ComplexSqr(inputvalue,Asqr);
    Sub(k1sqr,Asqr,kx1sqr);
    Sub(k2sqr,Asqr,kx2sqr);
    SquareRoot(kx1sqr,kx1);
    SquareRoot(kx2sqr,kx2);
    if kx1.Re < 0.0 then
      begin
       kx1.Re:= -kx1.Re;
       kx1.Im:= -kx1.Im
      end;
    if kx2.Re < 0.0 then
      begin
       kx2.Re:= -kx2.Re;
       kx2.Im:= -kx2.Im
      end;
end;  { procedure Calculatekx }

{ CALCULATION OF EX IN REGION 1 }

procedure CalculateEx1(x: extended);
var
  xcomplex,Robert,Mike: TNcomplex;
begin
  CMPLX(x,0,xcomplex);
  Mult(xcomplex,kx1,Robert);
  ComplexCOS(Robert,Mike);
  Mult(Mike,Answer,Robert);
  Mult(Robert,Answer,Mike);
  Divide(Mike,dum1,Ex1);
```

```
end; { procedure CalculateEx1}

{ CALCULATION OF EZ IN REGION 1 }

procedure CalculateEz1(x: extended);
var
  xcomplex,Robert,Mike,imunity: TNcomplex;
begin
  CMPLX(x,0,xcomplex);
  Mult(xcomplex,kx1,Robert);
  ComplexSIN(Robert,Mike);
  Mult(Mike,Answer,Robert);
  Mult(Robert,kx1,Mike);
  Divide(Mike,dum1,Robert);
  CMPLX(0,1,imunity);
  Mult(Robert,imunity,Ez1);
end; { procedure CalculateEz1 }

{ CALCULATION OF EX IN REGION 2 }

procedure CalculateEx2(x: extended);
var
  xcomplex,Robert,Mike,dcomplex,C2num,amdcomplex: TNcomplex;
  argdenom,C2denom: TNcomplex;
  a_minus_x,a_minus_d: extended;
begin
  a_minus_x:= a - x;
  CMPLX(a_minus_x,0,xcomplex);
  Mult(xcomplex,kx2,Robert);
  ComplexCOS(Robert,Mike);
  Mult(Mike,Answer,Robert);
  Mult(Robert,Answer,Mike);
  Divide(Mike,dum2,Robert);

  CMPLX(d,0,dcomplex);
  Mult(dcomplex,kx1,Mike);
  ComplexCOS(Mike,C2num);
  Mult(C2num,Robert,Mike);

  a_minus_d:= a - d;
  CMPLX(a_minus_d,0,amdcomplex);
  Mult(kx2,amdcomplex,argdenom);
  ComplexCOS(argdenom,C2denom);
  Divide(Mike,C2denom,Ex2)
end; { procedure CalculateEx2 }

{ CALCULATION OF EZ IN REGION 2 }

procedure CalculateEz2(x: extended);
```

```
var
 xcomplex,Robert,Mike,imunity,dcomplex,C2num,amdcomplex: TNcomplex;
 argdenom,C2denom: TNcomplex;
 a_minus_x,a_minus_d: extended;
begin
 a_minus_x:= a - x;
 CMPLX(a_minus_x,0,xcomplex);
 Mult(xcomplex,kx2,Robert);
 ComplexSIN(Robert,Mike);
 Mult(Mike,Answer,Robert);
 Mult(Robert,kx2,Mike);
 Divide(Mike,dum2,Robert);
 CMPLX(0,-1,imunity);
 Mult(Robert,imunity,Mike);

 CMPLX(d,0,dcomplex);
 Mult(dcomplex,kx1,Robert);
 ComplexCOS(Robert,C2num);
 Mult(C2num,Mike,Robert);

 a_minus_d:= a - d;
 CMPLX(a_minus_d,0,amdcomplex);
 Mult(kx2,amdcomplex,argdenom);
 ComplexCOS(argdenom,C2denom);
 Divide(Robert,C2denom,Ez2)
end; { procedure CalculateEz2 }



procedure UserInput(var Guess   : TNcomplex;
             var Tol    : Extended;
             var MaxIter : integer);

{-------------------------------------------------------}
{- Output: Guess, Tol, MaxIter                        -}
{-                                -}
{- This procedure assigns values to the above variables  -}
{- from keyboard input. The initial approximation of the -}
{- guess (Guess), tolerance (Tol), and maximum number of  -}
{- iterations (MaxIter) are all input here.           -}
{-------------------------------------------------------}

procedure GetInitialGuess(var Guess : TNcomplex);
var
 Answer: TNcomplex;
begin
 Writeln;
 Writeln('Initial approximation to the root: ');
 repeat
```

```
            Write('Re(Approximation) = ');
            Readln(Guess.Re);
            IOCheck;
           until not IOerr;
           repeat
            Write('Im(Approximation) = ');
            Readln(Guess.Im);
            IOCheck;
           until not IOerr;

           TNTargetF(Guess,Answer);
           Writeln('f(initial guess) = ',Answer.Re:23,' + ',Answer.Im:23,'i');

          end; { procedure GetInitialGuess }


          begin { procedure UserInput }
           GetInitialGuess(Guess);
          end; { procedure UserInput }

          {---------------------------------------------------------------------------}
          { OUTPUT THE RESULTS TO THE DEVICE OUTFILE                                   }
          {---------------------------------------------------------------------------}


          procedure Results(Guess   : TNcomplex;
                            Answer  : TNcomplex;
                            yAnswer : TNcomplex;
                            Tol     : Extended;
                            MaxIter : integer;
                            Iter    : integer;
                            Error   : byte);

          var
           x: extended;
           Ex1mag,Ex1ph,Ez1mag,Ez1ph,Ex2mag,Ex2ph,Ez2mag,Ez2ph,E1,E2,Eratio1,Eratio2:
extended;

          begin
           Writeln(OutFile);
           Write(OutFile, 'frequency: ' : 30);
           Writeln(OutFile, freq/1000:7:1, ' kHz');
           Writeln(OutFile);
           if Error in [1, 2] then
             DisplayWarning;
           if Error >= 3 then
             DisplayError;

           case Error of
```

```
  1 : Writeln(OutFile, 'This will take more than ', MaxIter, ' iterations.');

  2 : begin
       Writeln(OutFile, 'A parabola which intersects the x-axis can not');
       Writeln(OutFile, 'be constructed through these three points.');
     end;

  3 : Writeln(OutFile, 'The tolerance must be greater than zero.');

  4 : Writeln(OutFile,
          'The maximum number of iterations must be greater than zero.');
 end; { case }

 if Error <= 2 then
 begin
  Calculatekx(Answer);  { kx1 and kx2 are calculated from kz }

  Writeln(OutFile);
  Writeln(OutFile, 'Number of iterations: ' : 26, Iter:2);
  Write(OutFile, 'Calculated root (kz): ' : 26);
  Writeln(OutFile, Answer.Re:23, ' + ', Answer.Im:23, ' i');
   Write(OutFile, '    kx1      : ' : 26);
   Writeln(OutFile, kx1.Re:23, ' + ', kx1.Im:23, ' i');
   Write(OutFile, '    kx2      : ' : 26);
   Writeln(OutFile, kx2.Re:23, ' + ', kx2.Im:23, ' i');
  Writeln(OutFile, 'Value of the function  ' : 26);
  Write(OutFile, 'at the calculated root: ' : 26);
  Writeln(OutFile, yAnswer.Re:23, ' + ', yAnswer.Im:23, ' i');
  Writeln(OutFile);

{ =========== EVALUATION OF THE FIELDS ========= }
    x:= 0;
    while (x <= a ) do
    begin
     if x <= d then
      begin
       CalculateEz1(x);
       CalculateEx1(x);
        Ez1mag:= Modulus(Ez1);
         Ez1ph:= Phase(Ez1);
        Ex1mag:= Modulus(Ex1);
         Ex1ph:= Phase(Ex1);
        E1:= sqr(Ez1mag) + sqr(Ex1mag);
        Eratio1:= Ez1mag/Ex1mag;

Writeln(PlotFile,freq/1000:7:1,Chr(9),x:4:1,Chr(9),Ex1mag:23,Chr(9),Ex1ph:23,
            Chr(9),Ez1mag:23,Chr(9),Ez1ph:23,Chr(9),E1:23,Chr(9),Eratio1:23);
      end
      else
      begin
       CalculateEz2(x);
       CalculateEx2(x);
        Ez2mag:= Modulus(Ez2);
         Ez2ph:= Phase(Ez2);
        Ex2mag:= Modulus(Ex2);
         Ex2ph:= Phase(Ex2);
        E2:= sqr(Ez2mag) + sqr(Ex2mag);
        Eratio2:= Ez2mag/Ex2mag;

Writeln(PlotFile,freq/1000:7:1,Chr(9),x:4:1,Chr(9),Ex2mag:23,Chr(9),Ex2ph:23,
          Chr(9),Ez2mag:23,Chr(9),Ez2ph:23,Chr(9),E2:23,Chr(9),Eratio2:23);
      end;
    x:= x + 1
    end;
  end;
end; { procedure Results }

{--------------------------------------------------------------------------------}
{ MAIN PROGRAM                                         }
{--------------------------------------------------------------------------------}

begin { program Muller }
  GetTolerance(Tol);
  GetMaxIter(MaxIter);
  GetEndfreq(Endfreq);
  GetStepfreq(stepfreq);
  GetOutputFile(OutFile);
  GetOutputFile(PlotFile);


  repeat
   Writeln;Writeln;
   Write('Starting frequency(Hz) = ');
   Readln(freq);
   IOCheck;
  until not IOerr;

  repeat
   Writeln;Writeln;
   Writeln('Starting guess value');
   Write('    real part = ');
   Readln(Answer.Re);
   Write('    imaginary part = ');
   Readln(Answer.Im);
   IOCheck;
  until not IOerr;

  repeat
```

```
   Writeln;Writeln;
   Writeln('now working on frequency = ', freq/1000:7:1 , ' kHz');

{ CALCULATION OF THE SMALL ARGUMENT PROPAGATION CONSTANTS }

    omega:= 2*Pi*freq;

    dum1.Re:= sig1;      { for k1sqr }
    dum1.Im:= omega*eps1;

    dum2.Re:= sig2;      { for k2sqr }
    dum2.Im:= omega*eps2;

    dum.Re:= 0;         { for both k1sqr and k2sqr }
    dum.Im:= -omega*mu;  {*** omega*mu}

    Mult(dum1,dum,k1sqr);
    Mult(dum2,dum,k2sqr);

 {UserInput(Guess, Tol, MaxIter);}

 Guess:= Answer;

{ USE MULLER'S METHOD WITH DEFLATION TO FIND A ROOT}
 Muller(Guess, Tol, MaxIter, Answer, yAnswer, Iter, Error, @TNTargetf );

{ OUTPUT THE RESULTS }
 Results(Guess, Answer, yAnswer, Tol, MaxIter, Iter, Error);

   freq:= freq + Stepfreq;

  until freq > Endfreq;  { > or < depending on whether Stepfreq = + or - }


 Close(OutFile);
 Close(PlotFile);
WaitReturnOrClick;
end. { program Muller }
```

```
{ Program 5. Subroutine RootsOfEquat                    }


unit RootsOfEquat(2000);

{---------------------------------------------------------}
{-                                          -}
{-  This subroutine is a modified version of the RootsOfEquat  -}
{-  subroutine (Turbo Pascal Numerical Methods Toolbox        -}
{-  (c) Borland International). It is common to the programs  -}
{-  RepeatingMuller, SingleStepMuller, FillingMuller and      -}
{-  ERatioMuller. This subroutine implements the Muller       -}
{-  method for solving for complex roots of complex functions  -}
{-  and is documented throughout for easy understanding.      -}
{---------------------------------------------------------}

{$R+} { Enable range checking }

interface

uses
 MemTypes;

const
 TNNearlyZero = 1E-015;     { Close to zero }
 TNArraySize  = 30;         { Maximum size of vectors }

type
 TNvector    = array[0..TNArraySize] of Extended;

 TNIntVector  = array[0..TNArraySize] of integer;

 TNcomplex   = record
           Re, Im : Extended;
          end;

 TNCompVector = array[0..TNArraySize] of TNcomplex;


procedure Muller(Guess   : TNcomplex;
          Tol    : Extended;
          MaxIter : integer;
        var Answer  : TNcomplex;
        var yAnswer : TNcomplex;
        var Iter    : integer;
        var Error   : byte;
          FuncPtr : ProcPtr);
{-----------------------------------------------------------------------}
{-                                                   -}
{-          Input: Guess, Tol, MaxIter, FuncPtr                 -}
{-         Output: Answer, yAnswer, Iter, Error                  -}
{-                                                    -}
{-        Purpose: This program uses Muller's method to find a root   -}
{-              of a user defined function Y=TNTargetF given an     -}
{-              initial approximation.  The root may be complex.    -}
{-                                                    -}
{-                                                    -}
{-  User-Defined                                           -}
{-      Procedures: TNTargetF(X : TNcomplex; VAR Y : TNcomplex);        -}
{-                                                    -}
{-  Pre-Defined Types: TNcomplex = record                        -}
{-                        Re, Im : Extended;                 -}
{-                       end;                        -}
{-                                                    -}
{-       Variables: Guess   : Extended;       initial guess          -}
{-              Tol     : Extended;       tolerance in the        -}
{-                                answer              -}
{-              MaxIter : integer;        maximum number of       -}
{-                                iterations          -}
{-              Answer  : TNcomplex;      a root of the           -}
{-                                polynomial          -}
{-              yAnswer : TNcomplex;      value of the            -}
{-                                polynomial at the      -}
{-                                root (close to zero)   -}
{-              Iter    : integer;        number of iterations    -}
{-                                it took to find root    -}
{-              Error   : byte;           flags an error          -}
{-                                                    -}
{-         Errors: 0: No errors                            -}
{-              1: Iter > MaxIter                          -}
{-              2: parabola could not                        -}
{-                be formed                              -}
{-              3: Tol <= 0                              -}
{-              4: MaxIter < 0                             -}
{-                                                    -}
{----------------------------------------------------------------------}


implementation

{----------------------------------------------------------------------}
{- The following inline procedure and function are used to call the user  -}
{- defined procedures and functions pointed to by the ProcAddr parameter.  -}
{----------------------------------------------------------------------}

function UserFunction(X : Extended; ProcAddr : ProcPtr) : Extended;
inline
  $205F,  { MOVE.L (A7)+, A0 }
```

```
 $4E90; { JSR   (A0)    }

procedure UserProcedure(X : TNcomplex; var Y : TNcomplex; ProcAddr : ProcPtr);
inline
 $205F, { MOVE.L (A7)+, A0 }
 $4E90; { JSR   (A0)    }



procedure Muller((Guess  : TNcomplex;
            Tol    : Extended;
            MaxIter : integer;
         var Answer  : TNcomplex;
         var yAnswer : TNcomplex;
         var Iter    : integer;
         var Error   : byte;
            FuncPtr : ProcPtr));

type
 TNquadratic = record
             A, B, C : TNcomplex;
            end;

var
 X0, X1, OldApprox,
 NewApprox, yNewApprox : TNcomplex;   { Iteration variables }
 Factor : TNquadratic;             { Factor of polynomial }
 Found : boolean;                 { Flags that a factor }
                              { has been found }

{----------- HERE ARE SOME COMPLEX OPERATIONS ------------}

procedure Conjugate(C1 : TNcomplex; var C2 : TNcomplex);
begin
 C2.Re := C1.Re;
 C2.Im := -C1.Im;
end; { procedure Conjugate }

function Modulus(var C1 : TNcomplex) : Extended;
begin
 Modulus := Sqrt(Sqr(C1.Re) + Sqr(C1.Im));
end; { function Modulus }

procedure Add(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
 C3.Re := C1.Re + C2.Re;
 C3.Im := C1.Im + C2.Im;
end; { procedure Add }
```

```
procedure Sub(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
  C3.Re := C1.Re - C2.Re;
  C3.Im := C1.Im - C2.Im;
end; { procedure Sub }

procedure Mult(C1, C2 : TNcomplex; var C3 : TNcomplex);
begin
  C3.Re := C1.Re * C2.Re - C1.Im * C2.Im;
  C3.Im := C1.Im * C2.Re + C1.Re * C2.Im;
end; { procedure Mult }

procedure Divide(C1, C2 : TNcomplex; var C3 : TNcomplex);
var
  Dum1, Dum2 : TNcomplex;
  E : Extended;
begin
  Conjugate(C2, Dum1);
  Mult(C1, Dum1, Dum2);
  E := Sqr(Modulus(C2));
  C3.Re := Dum2.Re / E;
  C3.Im := Dum2.Im / E;
end; { procedure Divide }

procedure SquareRoot(C1 : TNcomplex; var C2 : TNcomplex);
var
  R, Theta : Extended;
begin
  R := Sqrt(Sqr(C1.Re) + Sqr(C1.Im));
  if ABS(C1.Re) < TNNearlyZero then
    begin
      if C1.Im < 0 then
        Theta := Pi / 2
      else
        Theta := -Pi / 2;
    end
  else
    if C1.Re < 0 then
      Theta := ArcTan(C1.Im / C1.Re) + Pi
    else
      Theta := ArcTan(C1.Im / C1.Re);
  C2.Re := Sqrt(R) * Cos(Theta / 2);
  C2.Im := Sqrt(R) * Sin(Theta / 2);
end; { procedure SquareRoot }


procedure MakeParabola(X0     : TNcomplex;
                       X1     : TNcomplex;
                       Approx : TNcomplex;
```

```
          var Factor : TNquadratic;
          var Error  : byte);

{--------------------------------------------------------}
{- Input:  X0, X1, Approx                   -}
{- Output: Factor, Error                    -}
{-                                 -}
{- This procedure constructs a parabola to fit the     -}
{- three points X0, X1, Approx.  The intersection of   -}
{- this parabola with the x-axis will yield the next   -}
{- approximation.  If the parabola is a horizontal line -}
{- then Error = 2 since a horizontal line will not     -}
{- intersect the x-axis.                    -}
{--------------------------------------------------------}

var
 H1, H2, H3, H : TNcomplex;
 Delta1, Delta2 : TNcomplex;
 Dum1, Dum2, Dum3, Dum4 : TNcomplex;     { Dummy variables }

begin
 with Factor do
 begin
  Sub(X0, Approx, H1);
  Sub(X1, Approx, H2);
  Sub(X0, X1, H3);
  Mult(H2, H3, Dum1);
  Mult(H1, Dum1, H);
  if Modulus(H) < TNNearlyZero then
   Error := 2;              { Can't fit a quadratic to these points }
  UserProcedure(X1, Dum1, FuncPtr);
  Sub(Dum1, C, Delta1);          { C was passed in }
  UserProcedure(X0, Dum1, FuncPtr);
  Sub(Dum1, C, Delta2);

  if Error = 0 then            { Calculate coefficients of quadratic }
  begin
   Mult(H1, H1, Dum1);          { Calculate B }
   Mult(Dum1, Delta1, Dum2);
   Mult(H2, H2, Dum1);
   Mult(Dum1, Delta2, Dum3);
   Sub(Dum2, Dum3, Dum4);
   Divide(Dum4, H, B);

   Mult(H2, Delta2, Dum1);        { Calculate A }
   Mult(H1, Delta1, Dum2);
   Sub(Dum1, Dum2, Dum3);
   Divide(Dum3, H, A);
  end;
  if (Modulus(A) <= TNNearlyZero) and (Modulus(B) <= TNNearlyZero) then
    Error := 2;          { Test if parabola is actually a constant }
 end; { with }
end; { procedure MakeParabola }

procedure Initial(Guess     : TNcomplex;
              Tol       : Extended;
              MaxIter   : integer;
          var Error     : byte;
          var Iter      : integer;
          var Found     : boolean;
          var NewApprox  : TNcomplex;
          var yNewApprox : TNcomplex;
          var X0        : TNcomplex;
          var X1        : TNcomplex;
          var OldApprox  : TNcomplex;
          var Factor    : TNquadratic);

{----------------------------------------------------------------}
{- Input:  Guess, Tol, MaxIter                          -}
{- Output: Error, Iter, Found, NewApprox, yNewApprox,          -}
{-        X0, X1, OldApprox, Factor                    -}
{-                                          -}
{- This procedure initializes all the above variables. It        -}
{- sets OldApprox equal to Guess, X0 and X1 are set close to      -}
{- Guess.  The procedure also checks the tolerance (Tol) and      -}
{- maximum number of iterations (MaxIter) for errors.           -}
{----------------------------------------------------------------}

var
 cZero, yOldApprox : TNcomplex;

begin
 cZero.Re := 0;   { Complex zero }
 cZero.Im := 0;   { Complex zero }
 Error := 0;
 Found := false;
 Iter := 0;
 NewApprox := cZero;
 yNewApprox := cZero;
     { X0 and X1 are points which are close to Guess }
 X0.Re := Guess.Re + 0.005;    X0.Im := Guess.Im;
 X1.Re := Guess.Re - 0.005;    X1.Im := Guess.Im;
 OldApprox := Guess;
 UserProcedure(OldApprox, yOldApprox, FuncPtr); { Evaluate the function at OldApprox }
 Factor.A := cZero;
 Factor.B := cZero;
 Factor.C := yOldApprox;
```

```
  MakeParabola(X0, X1, OldApprox, Factor, Error);
  if Tol <= 0 then
    Error := 3;
  if MaxIter < 0 then
    Error := 4;
end; { procedure Initial }

procedure QuadraticFormula(Factor    : TNquadratic;
                           OldApprox : TNcomplex;
                       var NewApprox : TNcomplex);

{-----------------------------------------------------------}
{- Input:  Factor, OldApprox                              -}
{- Output: NewApprox                                      -}
{-                                                        -}
{- This procedure applies the complex quadratic formula   -}
{- to the quadratic Factor to determine where the parabola -}
{- represented by Factor intersects the x-axis. The solution -}
{- of the quadratic formula is subtracted from OldApprox to -}
{- yield NewApprox.                                       -}
{-----------------------------------------------------------}

var
  Discrim, Difference, Dum1, Dum2, Dum3 : TNcomplex;

begin
  with Factor do
  begin
    Mult(B, B, Dum1);                   { B^2 }
    Mult(A, C, Dum2);
    Discrim.Re := Dum1.Re - 4 * Dum2.Re;  { B^2 - 4AC }
    Discrim.Im := Dum1.Im - 4 * Dum2.Im;
    SquareRoot(Discrim, Discrim);
    Sub(B, Discrim, Dum1);              { B +/- sqrt(B^2 - 4AC) }
    Add(B, Discrim, Dum2);
    { Choose the root with B +/- Discrim greatest }
    if Modulus(Dum1) < Modulus(Dum2) then
      Dum1 := Dum2;
    Add(C, C, Dum3);
    if Modulus(Dum1) < TNNearlyZero then  { if B +/- sqrt(B^2 - 4AC) = 0 }
      begin
        NewApprox.Re := 0;
        NewApprox.Im := 0;
      end
    else
      begin                         { 2C/[B +/- sqrt(B^2 - 4AC)] }
        Divide(Dum3, Dum1, Difference);
        { Calculate NewApprox }
        Sub(OldApprox, Difference, NewApprox);
    end;
  end; { with }
end; { procedure QuadraticFormula }

function TestForRoot(X    : TNcomplex;
                     OldX : TNcomplex;
                     Y    : TNcomplex;
                     Tol  : Extended) : boolean;

var
  Dif, FracDif : TNcomplex;

{-------------------------------------------------------------}
{- These are the stopping criteria.  Four different ones are    -}
{- provided.  If you wish to change the active criteria, simply  -}
{- comment off the current criteria (including the preceding OR) -}
{- and remove the comment brackets from the criteria (including  -}
{- the following OR) you wish to be active.                 -}
{-------------------------------------------------------------}

begin
  Sub(X, OldX, Dif);
  FracDif.Re := X.Re * Tol;
  FracDif.Im := X.Im * Tol;

  TestForRoot :=                    {---------------------------}
    (Modulus(Y) <= TNNearlyZero)      {- Y = 0                  -}
                                    {-                        -}
         or                          {-                        -}
                                    {-                        -}
    (Modulus(Dif) < Modulus(FracDif)) {- Relative change in X    -}
                                    {-                        -}
                                    {-                        -}
(*       or                        *) {-                        -}
(*                                 *) {-                        -}
(*  (Modulus(Dif) < Tol)           *) {- Absolute change in X    -}
(*                                 *) {-                        -}
(*       or                        *) {-                        -}
(*                                 *) {-                        -}
(*  (Modulus(Y) <= Tol)            *) {- Absolute change in Y    -}
                                    {---------------------------}

{-------------------------------------------------------------------}
{- The first criteria simply checks to see if the value of the        -}
{- function is zero.  You should probably always keep this criteria    -}
{- active.                                                -}
{-                                                        -}
{- The second criteria checks the relative error in x. This criteria  -}
{- evaluates the fractional change in x between interations. Note     -}
{- that x has been multiplied through the inequality to avoid Divide  -}
{- by zero errors.                                       -}
{-                                                   -}
{- The third criteria checks the absolute difference in x between      -}
{- iterations.                                          -}
{-                                                   -}
{- The fourth criteria checks the absolute difference between         -}
{- the value of the function and zero.                         -}
{-----------------------------------------------------------------------}

end; { function TestForRoot }

begin { procedure  Muller }
  Initial(Guess, Tol, MaxIter, Error, Iter, Found, NewApprox, yNewApprox,
       X0, X1, OldApprox, Factor);

  while not Found and (Error = 0) and (Iter < MaxIter) do
  begin
    Iter := Succ(Iter);
    QuadraticFormula(Factor, OldApprox, NewApprox);
    UserProcedure(NewApprox, yNewApprox, FuncPtr);   { Calculate a new yNewApprox }
    Found := TestForRoot(NewApprox, OldApprox, yNewApprox, Tol);
    X0 := X1;
    X1 := OldApprox;
    OldApprox := NewApprox;
    Factor.C := yNewApprox;
    MakeParabola(X0, X1, OldApprox, Factor, Error);
  end;
  Answer := NewApprox;
  yAnswer := yNewApprox;
  if Found then
    Error := 0
  else
    if (Error = 0) and (Iter >= MaxIter) then
      Error := 1;
end; { procedure Muller }


begin
end.  { RootsOfEquat }
```

```
C Program 6. ForwardE
C
C
C This 2-dimensional program calculates the time-dependent forwara-
C travelling electric fields over a grid in the x-z cross-section of a
C parallel-plate waveguide filled with either a homogeneous or 2-layered
C medium and terminated with a matched load. z is directed along the guide
C axis from source to load; x is directed from bottom to top plate; y is
C directed across the waveguide following a RH coordinate system.
C The origin of coordinates is located at the lower right hand corner
C of the problem domain.
C Note: The layers of media must be parallel to the parallel plates.
C
C Data input consists of the system parameters which are clearly
C identified in the program section: 'Entry of System Parameters.'
C Note: For more information see Figure 3. in the thesis text.
C
C Program output is written to output unit 8 and into a file called
C 'ELFIELD.' Output consists of the z and x coordinate and the
C corresponding axial and transverse electric field magnitudes according
C to the following collumn format:
C
C           z(m)   x(m)   Ez(V/m)   Ex(V/m)
C
C The output is formatted to be directly read by the graphing program
C Efields whose source code appears as program 7 in this appendix.
C
C ____________________________________________________________________
C DECLARATION STATEMENTS AND INPUT OF PARAMETERS
C ____________________________________________________________________
C
   INTEGER ZDIV,XDIV
      REAL F,A,D,REEL,FAKE,C1REAL,C1IMAG,C2REAL,C2IMAG
      REAL SIG1,SIG2,EPS1R,EPS2R,L,MU,REX1,IMEX1,T,EMAX
      COMPLEX CMPLX,EX1
      INTEGER M
C
   COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
C ____________________________________________________________________
C ENTRY OF SYSTEM PARAMETERS
C ____________________________________________________________________
C
C INPUT THE NUMBER OF DIVISIONS DESIRED FOR THE PROBLEM
C DOMAIN IN BOTH THE X AND Z DIRECTIONS. ZDIV IS THE NUMBER
C OF GRID DIVISIONS ALONG THE Z-AXIS. XDIV IS THE NUMBER OF
C GRID DIVISIONS ALONG THE X-AXIS.
C
   PARAMETER(ZDIV=25,XDIV=25)
```

```
      REAL EX(0:XDIV,0:ZDIV),EZ(0:XDIV,0:ZDIV)
C
C ENTER THE FREQUENCY OF OPERATION (HERTZ).
C
    PARAMETER(F=340.E3)
C
C ENTER THE PLATE SEPARATION (A) AND THE SATURATED
C REGION THICKNESS (D) (METRES)
C
    PARAMETER(A=15.0,D=12.75)
C
C ENTER THE WAVEGUIDE LENGTH (METRES)
C
    PARAMETER(L=400.0)
C
C ENTER THE TIME OF FIELD OBSERVATION (SECONDS)...NOTE: T=1/f=2*pi/w
C
    PARAMETER(T=0./(24*150E+3))
C
C ENTER THE REAL(REEL) AND IMAGINARY(FAKE) PARTS
C OF THE Z-PROPAGATION CONSTANT KZ=(BETA-J*ALPHA)=(REEL+J*FAKE)
C
    PARAMETER(REEL=2.898E-2,FAKE=-5.567E-3)
C
C ENTER THE CONDUCTIVITIES OF REGION1 (SATURATED REGION)
C AND REGION2 (DEPLETED REGION) (SIEMENS/METRE)
C
    DATA SIG1,SIG2 /1.E-3,1.E-6/
C
C ENTER THE RELATIVE DIELECTRIC CONSTANTS OF REGION1
C AND REGION2 (DIMENSIONLESS)
C
    DATA EPS1R,EPS2R /11.0,3.0/

C
C ENTER THE APPLIED EXCITATION VOLTAGE BETWEEN THE
C PARALLEL-PLATES (MAGNITUDE AND PHASE(DEGREES))
C
    PARAMETER(EX1MAG=1000.0,EX1ANG=0.0)
C
C ENTER THE MAGNETIC PERMEABILITY OF BOTH REGIONS (H/M)
C
      PARAMETER(PI=3.141592654)
    MU = 4*PI*1.E-7
C
C______________________________________________________________
C MAIN PROGRAM
C______________________________________________________________
C
```

```
      OPEN(UNIT=8,FILE='ELFIELD',STATUS='NEW')
C
C CALCULATE THE EXCITATION VECTOR IN RECTANGULAR FORM
C
         REX1=EX1MAG*COS(EX1ANG*PI/180)
         IMEX1=EX1MAG*SIN(EX1ANG*PI/180)
         EX1=CMPLX(REX1,IMEX1)
C
         W=2*PI*F
         CALL FORWARD(W,A,D,L,T,XDIV,ZDIV,EX1,REEL,FAKE,EX,EZ)
C
C PRINT THE NUMBER OF GRIDBLOCKS IN THE WAVEGUIDE
C
      WRITE(8,*) (XDIV+1)*(ZDIV+1)
C
C FIND EMAX, THE MAXIMUM ELECTRIC FIELD FOR SCALING PURPOSES
C
      EMAX=0.0
         DO 110 I = 0,XDIV
         DO 100 J = 0,ZDIV
         IF(EX(I,J).GT.EMAX) EMAX=EX(I,J)
         IF(EZ(I,J).GT.EMAX) EMAX=EZ(I,J)
   100 CONTINUE
   110 CONTINUE
C
C PRINT THE ELECTRIC FIELDS FOR PLOTTING
C
      DO 20 I=0,XDIV
         Z=0.
         DO 15 J=0,ZDIV
         WRITE(8,90) Z,X,EZ(I,J)/EMAX,EX(I,J)/EMAX
    90 FORMAT(' ',F9.3,3X,F9.3,5X,G15.7,3X,G15.7)
    15 Z=Z-L/ZDIV
    20 X=X+A/XDIV
C
C PRINT THE RATIO OF HEIGHT TO WIDTH AND THE WINDOW LIMITS ON THE PLOT
C
      WRITE(8,*) -L,0.0,0.0,A
C
      STOP
         END
C
C ____________________________________________________________________
C SUBROUTINE FOR THE CALCULATION OF THE FORWARD TRAVELLING WAVE
C ELECTRIC FIELDS IN BETWEEN THE PLATES FOR A MATCHED WAVEGUIDE
C ____________________________________________________________________
C
      SUBROUTINE FORWARD(W,A,D,L,T,XDIV,ZDIV,EX1,REEL,FAKE,EX,EZ)
C
```

```
C VARIABLE DECLARATIONS
C
      INTEGER ZDIV,XDIV
      REAL L,MU,EX(0:XDIV,0:ZDIV),EZ(0:XDIV,0:ZDIV),MAGE2
      REAL SIG1,SIG2,EP1,EPS1R,EP2,EPS2R,T,X,Z,REEL,FAKE
      COMPLEX CMPLX,CCOS,CSIN,CSQRT,K1,K2,KZ,C1,C2,CEXP
      COMPLEX MED1,MED2,KZ2,KX1,KX2,EX1,JJ,TIME,ZVAR
      COMPLEX TERM1,TERM2
C
      COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
C PRELIMINARY CALCULATIONS
C
      EP1=EPS1R*8.8542E-12
      EP2=EPS2R*8.8542E-12
      K1=CMPLX(0.,-W*MU)*CMPLX(SIG1,W*EP1)
      K2=CMPLX(0.,-W*MU)*CMPLX(SIG2,W*EP2)
      MED1=CMPLX(SIG1,W*EP1)
      MED2=CMPLX(SIG2,W*EP2)
      KZ=CMPLX(REEL,FAKE)
      KZ2=KZ*KZ
      KX1=CSQRT(K1-KZ2)
      KX2=CSQRT(K2-KZ2)
      JJ=CMPLX(0.0,1.0)
      TIME=CMPLX(COS(W*T),SIN(W*T))

C
C CALCULATION OF C1 AND C2 FROM THE EXCITATION FIELD
C
    TERM1=CSIN(KX1*D)/(MED1*KX1)
      TERM2=(CCOS(KX1*D)*CSIN(KX2*(A-D)))
      TERM2=TERM2/(CCOS(KX2*(A-D))*MED2*KX2)
      C1=EX1/(2*KZ2*CCOS(KZ*(-L))*(TERM1+TERM2)
      C2=C1*CCOS(KX1*D)/CCOS(KX2*(A-D))
C
C FIELD CALCULATIONS
C
      X=0.
      DO 110 I=0,XDIV
      Z=0.
      DO 100 J=0,ZDIV
      ZVAR=CEXP(-JJ*KZ*Z)
      IF(X.GT.D) GO TO 10
    EX(I,J)=REAL(C1*KZ2*CCOS(KX1*X)*ZVAR*TIME/MED1)
       EZ(I,J)=REAL(JJ*C1*KX1*KZ*CSIN(KX1*X)*ZVAR*TIME/MED1)
      GO TO 20
   10 EX(I,J)=REAL((C2*KZ2*CCOS(KX2*(A-X))*ZVAR*TIME/MED2)/10)
    EZ(I,J)=REAL((-JJ*C2*KX2*KZ*CSIN(KX2*(A-X))*ZVAR*TIME/MED2)/10)
   20 IF(I.EQ.0) EZ(I,J)=0.0
```

```
           IF(I.EQ.XDIV) EZ(I,J)=0.0
         100 Z=Z-L/ZDIV
         110 X=X+A/XDIV
           RETURN
             END
C
```

```
{ Program 7. Efields                                         }

{ This program generates the electric field line plots inside the parallel-  }
{ plate waveguide as shown in Figures 11 and 14 in this thesis. The          }
{ waveguide x-z cross-section is divided into grid blocks with the total     }
{ electric field magnitude and direction at the center of each grid-block    }
{ computed from information provided to the program through a data input     }
{ file. The line segment length in each grid-block is proportional to the    }
{ electric field magnitude there and the orientation of the segment          }
{ corresponds to the electric field direction at that grid-block.           }
{                                                        }
{ This program consists of the main program shown below and 3 subroutines    }
{ whose source codes are named UInput, UDisplayRoutines and UGlobals. The    }
{ subroutine UInput reads the data file ELFIELD and also prompts the user    }
{ for certain parameters when the program is run. The subroutine             }
{ UDisplayRoutines generates the electric field line plot from the input     }
{ data. The subroutine UGlobals is responsible for refreshing the plot       }
{ output on the screen and for monitoring and identifying keystrokes.        }
{                                                        }
{ Program input consists of an input data file whose name is prompted for    }
{ when the program is run. This data file is generated by the program        }
{ ForwardE or TimeStandingE and is called 'ELFIELD.' The data file consists  }
{ of the axial and transverse coordinates and the corresponding axial and    }
{ transverse electric field magnitudes in collumn format as illustrated below}
{                                                        }
{            z(m)   x(m)   Ez(V/m)   Ex(V/m)                       }
{                                                        }
{ Program output consists of the field line plot displayed on the screen.    }
{ The plot can be copied directly to a file which is readable by the drawing }
{ program Superpaint 2.0. Pressing the keys <shift-command-3> will save the  }
{ plot.                                                  }




{ © Edmund Sumbar, Applied Electromagnetics Lab, U of A, 1990 }

program Efields;

    uses
        Globals, Input, DisplayRoutines;

begin
    new(X);
    new(Y);
    new(Ex_over_Emax);
    new(Ey_over_Emax);

    PrepareTextWindow;
    ReadData;
    DrawMesh;
    HoldIt;

    PrintMesh;

    dispose(X);
    dispose(Y);
    dispose(Ex_over_Emax);
    dispose(Ey_over_Emax);
end.
```

```
{ Subroutine Uinput}

{ This subroutine reads the data from the input file ELFIELD and also       }
{ prompts the user for graph preferences such as scaling and line segment   }
{ appearance.                                                   }

{ © Edmund Sumbar, AEL, University of Alberta, 1990 }

unit Input;

interface

    uses
        Globals;

    procedure ReadData;

implementation

    var
        dataFile: text;
        fileName: string;

    procedure GetInput;
    forward;
    procedure GetParameters;
    forward;

    procedure ReadData;
    begin
        fileName := OldFileName('Select an existing data file');
        { This function is THINK-specific.   }
        reset(dataFile, fileName);
        Writeln('Reading input file...');
        GetInput;
        Writeln('Done.');
        GetParameters;
        close(dataFile);
    end;

    procedure GetInput;
        var
            i: integer;
    begin
        Readln(dataFile, n);
        for i := 1 to n do
            Readln(dataFile, X^[i], Y^[i], Ex_over_Emax^[i], Ey_over_Emax^[i]);
    end;
    procedure GetParameters;
    begin
        Writeln;
        Write('      X-coordinate scale: ');
        Readln(Xscale);
        Write('      Y-coordinate scale: ');
        Readln(Yscale);
        Write('           field scale: ');
        Readln(Escale);
        Write('Display 1-lines or 2-arrows: ');
        Readln(plotChoice);
    end;

end.
```

```
{ Subroutine UDisplayRoutines     }

{ This subroutine generates the electric field line plot from the input data }

{ © Edmund Sumbar, AEL, University of Alberta, 1990 }

unit DisplayRoutines;

interface

     uses
          Globals, PrintTraps;

     procedure PrepareTextWindow;
     procedure DrawMesh;
     procedure PrintMesh;

implementation

     var
          displayRect: Rect;
          xPoint, yPoint, ExPoint, EyPoint: array[1..MAX_N] of integer;

     procedure SetCoordinates;
     forward;
     procedure GetScaledMesh;
     forward;
     procedure DoTheDrawingWithLineSegments;
     forward;
     procedure DoTheDrawingWithArrows;
     forward;

     procedure PrepareTextWindow;
     begin
          displayRect := screenBits.bounds;
          displayRect.top := displayRect.top + 36;
          InsetRect(displayRect, 4, 4);
          SetTextRect(displayRect);     { These are THINK-specific procedures. }
          ShowText;
     end;      { end of PrepareTextWindow }

     procedure DrawMesh;
     begin
{ Some of these are THINK-specific procedures. }
          SetDrawingRect(displayRect);
{ DisplayRect is set during the call to PrepareTextWindow.      }
          ShowDrawing;
          SetCoordinates;
          GetScaledMesh;
```

```
        case plotChoice of
            LINES:
                DoTheDrawingWithLineSegments;
            ARROWS:
                DoTheDrawingWithArrows;
        end;
    end;        { end of DrawMesh }

    procedure PrintMesh;
        var
            printRecordH: THPrint;
            printPort: TPPrPort;
            statusRec: TPrStatus;
    begin
{ Set up the variables for printing. }
        printRecordH := THPrint(NewHandle(sizeof(TPrint)));
        PrOpen;
        PrintDefault(printRecordH);
{ Display the printer SetUp and Print dialog boxes. }
        if not (PrStlDialog(printRecordH)) then
            Exit(PrintMesh);
        if not (PrJobDialog(printRecordH)) then
            Exit(PrintMesh);
        printPort := PrOpenDoc(printRecordH, nil, nil);
        PrOpenPage(printPort, nil);

{ Do the drawing of the mesh on the printer GrafPort. }
        SetCoordinates;
        GetScaledMesh;
        case plotChoice of
            LINES:
                DoTheDrawingWithLineSegments;
            ARROWS:
                DoTheDrawingWithArrows;
        end;

{ Finished printing. Now clean up. }
        PrClosePage(printPort);
        PrCloseDoc(printPort);
        PrPicFile(printRecordH, nil, nil, nil, statusRec);
        PrClose;
    end;        { end of PrintMesh }

    procedure SetCoordinates;
{ Adjusts the coordinate system so that 0,0 is in the lower left part of the}
{ window (not at the corner!). Positive x goes to the right and neg. y goes up.}
{ The clipping region is inset from the edges of the port rectangle by 3.}
{ Arrange for the minimum x and y values to be translated to zero. }
{ Calculate the coords. of the mesh as offsets from the minimums and use      }
{ a negative y for plotting  }
          var
               newV: integer;
     begin
          newV := -(thePort^.portRect.bottom - 36);
{ The hor. scroll bar 's 16 pixels tall. }
          SetOrigin(-20, newV);
          displayRect := thePort^.portRect;
{ The clipping region moves with SetOrigin so change it.     }
          InsetRect(displayRect, 3, 3);
          displayRect.right := displayRect.right - 16;
{ Allow for the scroll bars with a factor of 16. }
          displayRect.bottom := displayRect.bottom - 16;
          ClipRect(displayRect);
     end;      { end of SetCoordinates }

     procedure GetScaledMesh;
          var
               windowAspect: extended;
               xWidth, yHeight: extended;
               windowWidth, windowHeight: integer;
               xOffset, yOffset: integer;
               scale: extended;
               i: integer;
     begin

          xMax := 0.0;
          yMax := 0.0;
          xMin := 1e7;
          yMin := 1e7;
          for i := 1 to n do
               begin
                    if X^[i] > xMax then
                         xMax := X^[i]
                    else if X^[i] < xMin then
                         xMin := X^[i];

                    if Y^[i] > yMax then
                         yMax := Y^[i]
                    else if Y^[i] < yMin then
                         yMin := Y^[i];
               end;

          xWidth := (xMax - xMin) * Xscale;
{ Note assignment compatibility, p.246 manual. }
          yHeight := (yMax - yMin) * Yscale;
          gridAspect := yHeight / xWidth;

          with  isplayRect do
```

```
            begin
                windowWidth := (right - left - 40);
{ Use a slightly smaller window size. }
                windowHeight := (bottom - top - 40);
            end;
        windowAspect := windowHeight / windowWidth;

{ Evaluate the scale factor depending on the difference in aspect ratios.}
{ Calculate offsets so as to center the mesh.            }
        if gridAspect <= windowAspect then
            begin
                scale := windowWidth / xWidth;
                xOffset := 0;
                yOffset := trunc((windowHeight - yHeight * scale) / 2);
            end
        else if gridAspect > windowAspect then
            begin
                scale := windowHeight / yHeight;
                xOffset := trunc((windowWidth - xWidth * scale) / 2);
                yOffset := 0;
            end;

{ Fill integer arrays with the displaced and scaled grid points. }
        for i := 1 to n do
            begin
                xPoint[i] := trunc((X^[i] - xMin) * scale * Xscale) + xOffset;
                yPoint[i] := -trunc((Y^[i] - yMin) * scale * Yscale) - yOffset;
                ExPoint[i] := trunc(Escale * Ex_over_Emax^[i]);
                EyPoint[i] := -trunc(Escale * Ey_over_Emax^[i]);
            end;
    end;        { end of GetScaledMesh }

    procedure DoTheDrawingWithLineSegments;
        var
            i: integer;
            pointRect: Rect;
    begin
        for i := 1 to n do
            begin
                with pointRect do
                    begin
                        top := yPoint[i] - 2;
                        left := xPoint[i] - 2;
                        bottom := yPoint[i] + 2;
                        right := xPoint[i] + 2;
                    end;
                MoveTo(xPoint[i] - 2, yPoint[i] - 2);
                PaintOval(pointRect);
                Move(2, 2);
            Line(ExPoint[i], EyPoint[i]);
        end;
    end;        { end of DoTheDrawingWithLineSegments }

    procedure DoTheDrawingWithArrows;
    const
        ARROW_UNIT = 1;
    var
        i: integer;
        theArrow: PolyHandle;
        newX, newY: integer;
        theta, angle: extended;
        X1, Y1, X2, Y2, X3, Y3, X4, Y4, X6, Y6, X7, Y7: integer;
        arrowScale: integer;
        pi: extended;
        startPoint: point;

    function XRotate (x, y: extended): integer;
    begin
        XRotate := trunc(x * cos(theta) + y * sin(theta));
    end;
    function YRotate (x, y: extended): integer;
    begin
        YRotate := trunc(-x * sin(theta) + y * cos(theta));
    end;
    function GetAngle: extended;
    begin
        if (Ex_over_Emax^[i] = 0) and (Ey_over_Emax^[i] > 0) then
            GetAngle := 0;
        if (Ex_over_Emax^[i] = 0) and (Ey_over_Emax^[i] < 0) then
            GetAngle := pi;

        if (Ex_over_Emax^[i] > 0) and (Ey_over_Emax^[i] > 0) then
            GetAngle := arctan(abs(Ey_over_Emax^[i] /
             Ex_over_Emax^[i])) - pi / 2;
        if (Ex_over_Emax^[i] < 0) and (Ey_over_Emax^[i] > 0) then
            GetAngle := pi / 2 - arctan(abs(Ey_over_Emax^[i] /
             Ex_over_Emax^[i]));
        if (Ex_over_Emax^[i] > 0) and (Ey_over_Emax^[i] < 0) then
            GetAngle := -(pi / 2 + arctan(abs(Ey_over_Emax^[i] /
             Ex_over_Emax^[i])));
        if (Ex_over_Emax^[i] < 0) and (Ey_over_Emax^[i] < 0) then
            GetAngle := pi / 2 + arctan(abs(Ey_over_Emax^[i] /
            Ex_over_Emax^[i]));
        end;

begin
    pi := 4 * arctan(1);
    for i := 1 to n do
```

```
               begin
{ Set the angle here! }
                    theta := GetAngle;  { in radians. }

{ Set the arrow scale here! The basic arrow unit is 1. Use a maximum scale of about }
{ 10. Don't forget that the field is scaled when the plotting points are calculated. }
                    arrowScale := trunc(Escale * sqrt(sqr(Ex_over_Emax^[i]) +
                     sqr(Ey_over_Emax^[i])));

{ Rotate the corners of the arrow from its initial vertical orientation. }
{ Positive angles go CCW. }
                    X1 := XRotate(-2 * ARROW_UNIT * arrowScale, 0);
                    Y1 := YRotate(-2 * ARROW_UNIT * arrowScale, 0);
                    X2 := XRotate(-ARROW_UN:T * arrowScale, 0);
                    Y2 := YRotate(-ARROW_UNIT * arrowScale, 0);
                    X3 := XRotate(3 * ARROW_UNIT * arrowScaie, -2 *
                    ARROW_UNIT * arrowScale);
                    Y3 := YRotate(3 * ARROW_UNIT * arrowScale, -2 *
                    ARROW_UNIT * arrowScale);
                    X4 := XRotate(3 * ARROW_UNIT * arrowScale, 2 *
                    ARROW_UNIT * arrowScale);
                    Y4 := YRotate(3 * ARROW_UNIT * arrowScale, 2 *
                    ARROW_UNIT * arrowScale);
                    X6 := XRotate(0, 2 * ARROW_UNIT * arrowScale);
                    Y6 := YRotate(0, 2 * ARROW_UNIT * arrowScale);
                    X7 := XRotate(-4 * ARROW_UNIT * arrowScale, 0);
                    Y7 := YRotate(-4 * ARROW_UNIT * arrowScale, 0);
                    theArrow := OpenPoly;
                    MoveTo(xPoint[i], yPoint[i]);
                    Move(X1, Y1);
                    GetPen(startPoint);
                    Line(X2, Y2);
                    Line(X3, Y3);
                    Line(X4, Y4);
                    Line(X2, Y2);
                    Line(X6, Y6);
                    Line(X7, Y7);
                    LineTo(startPoint.h, startPoint.v);
                    ClosePoly;
                    FillPoly(theArrow, white);
                    FramePoly(theArrow);
                    KillPoly(theArrow);
               end;
     end;      { end of DoTheDrawingWithArrows }

end.
```

```
{ Subroutine UGlobals }

{ This subroutine refreshes the screen output and monitors and identifies     }
{ keystrokes .                    }

{ © Edmund Sumbar, AEL, University of Alberta, 1990 }

unit Globals;

interface

     const
          MAX_N = 26 * 26;
          LINES = 1;
          ARROWS = 2;

     type
          CoordinateArray = array[1..MAX_N] of extended;

     var
          n: integer;
          X, Y, Ex_over_Emax, Ey_over_Emax: ^CoordinateArray;
          gridAspect, xMin, xMax, yMin, yMax: extended;
          Xscale, Yscale, Escale: extended;
          plotChoice: integer;

     procedure HoldIt;

implementation

     procedure HoldIt;
{ Puts the program into a loop. The knit applications needs this to retain        }
{ the drawing window on screen before program termination.                        }
          const
               CR = $0D;        { Character code (in Hex) for the carriage-return key. }
               ETX = $03;       { Character code (in Hex) for the enter key. }
          var
               Done: boolean;                    {Flag used in the mini-event loop.}
               myEvent: EventRecord;
          function Get_Return: boolean;
     {Determines whether or not keystroke is the return-key or enter-key.}
          begin
               Get_Return := false;
               if BitAnd(myEvent.message, charCodeMask) = CR then
                    Get_Return := true;
               if BitAnd(myEvent.message, charCodeMask) = ETX then
                    Get_Return := true;
          end;
     begin
```

```
        Done := false;
{The following is a mini-event loop which waits for a mouse click or carriage return.}
        repeat
            SystemTask;
            if GetNextEvent(everyEvent, myEvent) then
                case myEvent.what of
                    mouseDown:
                        Done := true;
                    keyDown:
                        if Get_Return then
                            Done := true;
                    otherwise
                        ;    {Do nothing in the case of other events.}
                end;
        until Done;
    end;    { end of HoldIt }

end.
```

```
C Program 8. StandingE
C
C
C This 2-dimensional program calculates the phasor standing-wave electric
C fields over a grid of an x-z cross-section of a parallel-plate waveguide
C filled with either a homogeneous or 2-layered medium and terminated with
C an open or short-circuit. z is directed along the waveguide axis
C from source to load; x is directed from bottom to top plate; y is
C directed across the waveguide following a RH coordinate system.
C The origin of coordinates is at the right hand bottom of the
C 2-D problem domain.
C Note: The layers of media must be parallel to the parallel plates.
C
C Data input consists of the system parameters which are clearly
C identified in the program section: 'Entry of System Parameters.'
C Note: For more information, see Figure 3. in the thesis text.
C
C Program output is written to output unit 8 and into a file called
C 'ELFIELD.' Output consists of the z and x coordinate and the
C corresponding axial and transverse phasor standing-wave electric field
C magnitudes according to the following collumn format:
C
C            z(m)   x(m)   Ez(V/m)   Ex(V/m)
C
C The output is formatted to be directly read by the graphing program
C Cricketgraph.
C
C _______________________________________________________________
C DECLARATION STATEMENTS AND INPUT OF PARAMETERS
C _______________________________________________________________
C
   INTEGER ZDIV,XDIV
      REAL F,A,D,REEL,FAKE,C1REAL,C1IMAG,C2REAL,C2IMAG
      REAL SIG1,SIG2,EPS1R,EPS2R,L,MU,REX1,IMEX1,T,EMAX
      COMPLEX CMPLX,EX1
      INTEGER M
      CHARACTER*1 TAB
C
   COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
C
C _______________________________________________________________
C ENTRY OF SYSTEM PARAMETERS
C _______________________________________________________________
C
C INPUT THE NUMBER OF DIVISIONS DESIRED FOR THE PROBLEM
C DOMAIN IN BOTH THE X AND Z DIRECTIONS. ZDIV IS THE NUMBER
C OF GRID DIVISIONS ALONG THE Z-AXIS. XDIV IS THE NUMBER OF
C GRID DIVISIONS ALONG THE X-AXIS.
```

```
C
      PARAMETER(ZDIV=25,XDIV=25)
         REAL EX(0:XDIV,0:ZDIV),EZ(0:XDIV,0:ZDIV)
C
C ENTER THE FREQUENCY OF OPERATION (HERTZ).
C
      PARAMETER(F=250000.0)
C
C ENTER THE PLATE SEPARATION (A) AND THE SATURATED
C REGION THICKNESS (D) (METRES)
C
      PARAMETER(A=15.0,D=12.75)
C
C ENTER THE WAVEGUIDE LENGTH (METRES)
C
      PARAMETER(L=400.0)
C
C ENTER THE PHASE(REEL) AND ATTENUATION(FAKE) COMPONENTS
C OF THE Z-PROPAGATION CONSTANT KZ=(BETA-J*ALPHA)=(REEL+J*FAKE)
C
      PARAMETER(REEL=2.208E-2,FAKE=-3.473E-3)
C
C INDICATE WHETHER THE WAVEGUIDE TERMINATION IS AN
C OPEN CIRCUIT(0) OR A SHORT CIRCUIT(1)
C
      PARAMETER(M=0)
C
C ENTER THE CONDUCTIVITIES OF REGION1 (SATURATED REGION)
C AND REGION2 (DEPLETED REGION) (SIEMENS/METRE)
C
      DATA SIG1,SIG2 /1.E-3,1.E-6/
C
C ENTER THE RELATIVE DIELECTRIC CONSTANTS OF REGION1
C AND REGION2 (DIMENSIONLESS)
C
      DATA EPS1R,EPS2R /11.0,3.0/

C
C ENTER THE APPLIED EXCITATION VOLTAGE BETWEEN THE
C PARALLEL-PLATES (MAGNITUDE AND PHASE(DEGREES))
C
      PARAMETER(EX1MAG=1000.0,EX1ANG=0.0)
C
C ENTER THE MAGNETIC PERMEABILITY OF BOTH REGIONS (H/M)
C
         PARAMETER(PI=3.141592654)
      MU = 4*PI*1.E-7
C
C______________________________________________________________
```

```
C MAIN PROGRAM
C______________________________________________________________
C
   OPEN(UNIT=8,FILE='ELFIELD',STATUS='NEW')
C
C CALCULATE THE EXCITATION VECTOR IN RECTANGULAR FORM
C
      REX1=EX1MAG*COS(EX1ANG*PI/180)
      IMEX1=EX1MAG*SIN(EX1ANG*PI/180)
      EX1=CMPLX(REX1,IMEX1)
C
C CALL THE APPROPRIATE SUBROUTINES DEPENDING ON WHETHER THE LOAD
C IS OPEN OR SHORT CIRCUITED
C
      W=2*PI*F
      IF(M.EQ.1) GO TO 10
      CALL OPENN(W,A,D,L,T,XDIV,ZDIV,EX1,REEL,FAKE,EX,EZ)
      GOTO 11
 10 CALL SHORT(W,A,D,L,T,XDIV,ZDIV,EX1,REEL,FAKE,EX,EZ)
 11 X=0.
C
C FIND EMAX, THE MAXIMUM ELECTRIC FIELD FOR SCALING PURPOSES
C
   EMAX=0.0
      DO 110 I = 0,XDIV
      DO 100 J = 0,ZDIV
      IF(EX(I,J).GT.EMAX) EMAX=EX(I,J)
      IF(EZ(I,J).GT.EMAX) EMAX=EZ(I,J)
 100 CONTINUE
 110 CONTINUE
C
C PRINT THE ELECTRIC FIELDS FOR PLOTTING
C
   TAB=CHAR(9)
C
   DO 20 I=0,XDIV
      Z=0.
      DO 15 J=0,ZDIV
      WRITE(8,90) Z,TAB,X,TAB,EZ(I,J),TAB,EX(I,J)
 90 FORMAT(' ',F9.3,A,F9.3,A,G15.7,A,G15.7)
 15 Z=Z-L/ZDIV
 20 X=X+A/XDIV
C
   STOP
      END
C
C ______________________________________________________________
C SUBROUTINE FOR THE CALCULATION OF THE OHMIC HEATING RATE
C IN BETWEEN THE PLATES FOR A SHORT-CIRCUITED WAVEGUIDE
```

```
C ______________________________________________________________
C
    SUBROUTINE SHORT(W,A,D,L,T,XDIV,ZDIV,EX1,REEL,FAKE,EX,EZ)
C
C VARIHBLE DECLARATIONS
C
      INTEGER ZDIV,XDIV
      REAL L,MU,EX(0:XDIV,0:ZDIV),EZ(0:XDIV,0:ZDIV),MAGE2
      REAL SIG1,SIG2,EP1,EPS1R,EP2,EPS2R,T
      COMPLEX CMPLX,CCOS,CSIN,CSQRT,K1,K2,KZ,C1,C2
      COMPLEX MED1,MED2,KZ2,KX1,KX2,EX1,JJ,TERM1,TERM2
C
      COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
C PRELIMINARY CALCULATIONS
C
      EP1=EPS1R*8.8542E-12
      EP2=EPS2R*8.8542E-12
      K1=CMPLX(0.,-W*MU)*CMPLX(SIG1,W*EP1)
      K2=CMPLX(0.,-W*MU)*CMPLX(SIG2,W*EP2)
      MED1=CMPLX(SIG1,W*EP1)
      MED2=CMPLX(SIG2,W*EP2)
      KZ=CMPLX(REEL,FAKE)
      KZ2=KZ*KZ
      KX1=CSQRT(K1-KZ2)
      KX2=CSQRT(K2-KZ2)
      JJ=CMPLX(0.0,1.0)

C
C CALCULATION OF C1 AND C2 FROM THE EXCITATION FIELD
C
    TERM1=CSIN(KX1*D)/(MED1*KX1)
      TERM2=(CCOS(KX1*D)*CSIN(KX2*(A-D)))
      TERM2=TERM2/(CCOS(KX2*(A-D))*MED2*KX2)
      C1=EX1/(2*KZ2*CCOS(KZ*(-L))*(TERM1+TERM2))
      C2=C1*CCOS(KX1*D)/CCOS(KX2*(A-D))
C
C FIELD CALCULATIONS
C
      X=0.
      DO 110 I=0,XDIV
      Z=0.
      DO 100 J=0,ZDIV
      IF(X.GT.D) GO TO 10
    EX(I,J)=CABS(-JJ*2*C1*KZ2*CCOS(KX1*X)*CSIN(KZ*Z)/MED1)
      EZ(I,J)=CABS(JJ*2*C1*KX1*KZ*CSIN(KX1*X)*CCOS(KZ*Z)/MED1)
      GO TO 20
  10 EX(I,J)=CABS(-JJ*2*C2*KZ2*CCOS(KX2*(A-X))*CSIN(KZ*Z)/MED2)
    EZ(I,J)=CABS(-JJ*2*C2*KX2*KZ*CSIN(KX2*(A-X))*CCOS(KZ*Z)/MED2)
  20 IF(I.EQ.0) EZ(I,J)=0.0
     IF(I.EQ.XDIV) EZ(I,J)=0.0
        IF(J.EQ.0) EX(I,J)=0.0
 100 Z=Z-L/ZDIV
 110 X=X+A/XDIV
     RETURN
        END
C
C ________________________________________________________________________
C SUBROUTINE FOR THE CALCULATION OF THE OHMIC HEATING RATE
C BETWEEN THE PLATES FOR AN OPEN-CIRCUITED WAVEGUIDE
C ________________________________________________________________________
C
     SUBROUTINE OPENN(W,A,D,L,T,XDIV,ZDIV,EX1,REEL,FAKE,EX,EZ)
C
C DECLARATIONS
C
        INTEGER ZDIV,XDIV
        REAL L,MU,EX(0:XDIV,0:ZDIV),EZ(0:XDIV,0:ZDIV),MAGE2
        REAL SIG1,SIG2,EP1,EPS1R,EP2,EPS2R
        COMPLEX CMPLX,CCOS,CSIN,CSQRT,K1,K2,KZ,C1,C2
        COMPLEX MED1,MED2,KZ2,KX1,KX2,EX1,TERM1,TERM2
C
        COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
C PRELIMINARY CALCULATIONS
C
        EP1=EPS1R*8.8542E-12
        EP2=EPS2R*8.8542E-12
        K1=CMPLX(0.,-W*MU)*CMPLX(SIG1,W*EP1)
        K2=CMPLX(0.,-W*MU)*CMPLX(SIG2,W*EP2)
        MED1=CMPLX(SIG1,W*EP1)
        MED2=CMPLX(SIG2,W*EP2)
        KZ=CMPLX(REEL,FAKE)
        KZ2=KZ*KZ
        KX1=CSQRT(K1-KZ2)
        KX2=CSQRT(K2-KZ2)
C
C CALCULATION OF C1 AND C2 FROM THE EXCITATION FIELD
C
     TERM1=CSIN(KX1*D)/(MED1*KX1)
        TERM2=(CCOS(KX1*D)*CSIN(KX2*(A-D)))
        TERM2=TERM2/(CCOS(KX2*(A-D))*MED2*KX2)
        C1=EX1/(2*KZ2*CCOS(KZ*(-L))*(TERM1+TERM2))
        C2=C1*CCOS(KX1*D)/CCOS(KX2*(A-D))
C
C FIELD CALCULATIONS
C
        X=0.
```

```
      DO 110 I=0,XDIV
      Z=0.
      DO 100 J=0,ZDIV
      IF(X.GT.D) GO TO 10
      EX(I,J)=CABS(2*C1*KZ2*CCOS(KX1*X)*CCOS(KZ*Z)/MED1)
      EZ(I,J)=CABS(2*C1*KX1*KZ*CSIN(KX1*X)*CSIN(KZ*Z)/MED1)
      GO TO 20
 10 EX(I,J)=CABS(2*C2*KZ2*CCOS(KX2*(A-X))*CCOS(KZ*Z)/MED2)
   EZ(I,J)=CABS(-2*C2*KX2*KZ*CSIN(KX2*(A-X))*CSIN(KZ*Z)/MED2)
 20 IF(I.EQ.0) EZ(I,J)=0.0
   IF(I.EQ.XDIV) EZ(I,J)=0.0
100 Z=Z-L/ZDIV
110 X=X+A/XDIV
   RETURN
      END
```

```
C Program 9. TimeStandingE
C
C
C This 2-dimensional program calculates the time-dependent electric
C fields over the x-z cross-sectional grid of a parallel-plate waveguide
C filled with either a homogeneous or 2-layered medium and terminated with
C an open or short-circuit. z is directed along the waveguide axis
C from source to load; x is directed from bottom to top plate; y is
C directed across the waveguide following a RH coordinate system.
C The origin of coordinates is located at the lower right hand
C corner of the problem domain.
C Note: The layers of media must be parallel to the parallel plates.
C
C Data input consists of the system parameters which are clearly
C identified in the program section: 'Entry of System Parameters'
C
C Program output is written to output unit 8 and into a file called
C 'ELFIELD.' Output consists of the z and x coordinate and the
C corresponding axial and transverse time-dependent standing-wave electric
C field magnitudes according to the following collumn format:
C
C            z(m)  x(m)   Ez(V/m)   Ex(V/m)
C
C The output is formatted to be directly read by the graphing program
C SYSTAT.
C
C ______________________________________________________________________
C DECLARATION STATEMENTS AND INPUT OF PARAMETERS
C ______________________________________________________________________
C
      INTEGER ZDIV,XDIV
        REAL F,A,D,REEL,FAKE,C1REAL,C1IMAG,C2REAL,C2IMAG
        REAL SIG1,SIG2,EPS1R,EPS2R,L,MU,REX1,IMEX1,T,EMAX
        COMPLEX CMPLX,EX1
        INTEGER M
C
      COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
C ______________________________________________________________________
C ENTRY OF SYSTEM PARAMETERS
C ______________________________________________________________________
C
C INPUT THE NUMBER OF DIVISIONS DESIRED FOR THE PROBLEM
C DOMAIN IN BOTH THE X AND Z DIRECTIONS. ZDIV IS THE NUMBER
C OF GRID DIVISIONS ALONG THE Z-AXIS. XDIV IS THE NUMBER OF
C GRID DIVISIONS ALONG THE X-AXIS.
C
      PARAMETER(ZDIV=25,XDIV=25)
        REAL EX(0:XDIV,0:ZDIV),EZ(0:XDIV,0:ZDIV)
```

```
C
C ENTER THE FREQUENCY OF OPERATION (HERTZ).
C
     PARAMETER(F=150.E3)
C
C ENTER THE PLATE SEPARATION (A) AND THE SATURATED
C REGION THICKNESS (D) (METRES)
C
     PARAMETER(A=15.0,D=13.5)
C
C ENTER THE WAVEGUIDE LENGTH (METRES)
C
     PARAMETER(L=400.0)
C
C ENTER THE TIME OF FIELD OBSERVATION (SECONDS)...NOTE: T=1/f=2*pi/w
C
     PARAMETER(T=3/(24*150E+3))
C
C ENTER THE REAL(REEL) AND IMAGINARY(FAKE) PARTS
C OF THE Z-PROPAGATION CONSTANT KZ=(BETA-J*ALPHA)=(REEL+J*FAKE)
C
     PARAMETER(REEL=1.6399E-2,FAKE=-2.518E-3)
C
C INDICATE WHETHER THE WAVEGUIDE TERMINATION IS AN
C OPEN CIRCUIT(0) OR A SHORT CIRCUIT(1)
C
     PARAMETER(M=1)
C
C ENTER THE CONDUCTIVITIES OF REGION1 (SATURATED REGION)
C AND REGION2 (DEPLETED REGION) (SIEMENS/METRE)
C
     DATA SIG1,SIG2 /1.E-3,1.E-6/
C
C ENTER THE RELATIVE DIELECTRIC CONSTANTS OF REGION1
C AND REGION2 (DIMENSIONLESS)
C
     DATA EPS1R,EPS2R /11.0,3.0/

C
C ENTER THE APPLIED EXCITATION VOLTAGE BETWEEN THE
C PARALLEL-PLATES (MAGNITUDE AND PHASE(DEGREES))
C
     PARAMETER(EX1MAG=1000.0,EX1ANG=0.0)
C
C ENTER THE MAGNETIC PERMEABILITY OF BOTH REGIONS (H/M)
C
        PARAMETER(PI=3.141592654)
     MU = 4*PI*1.E-7
C
```

```
C____________________________________________________________________
C MAIN PROGRAM
C____________________________________________________________________
C
    OPEN(UNIT=8,FILE='ELFIELD',STATUS='NEW')
C
C CALCULATE THE EXCITATION VECTOR IN RECTANGULAR FORM
C
      REX1=EX1MAG*COS(EX1ANG*PI/180)
      IMEX1=EX1MAG*SIN(EX1ANG*PI/180)
      EX1=CMPLX(REX1,IMEX1)
C
C CALL THE APPROPRIATE SUBROUTINES DEPENDING ON WHETHER THE LOAD
C IS OPEN OR SHORT CIRCUITED
C
      W=2*PI*F
      IF(M.EQ.1) GO TO 10
      CALL OPENN(W,A,D,L,T,XDIV,ZDIV,EX1,REEL,FAKE,EX,EZ)
      GOTO 11
  10 CALL SHORT(W,A,D,L,T,XDIV,ZDIV,EX1,REEL,FAKE,EX,EZ)
  11 X=0.
C
C PRINT THE NUMBER OF GRIDBLOCKS IN THE WAVEGUIDE
C
    WRITE(8,*) (XDIV+1)*(ZDIV+1)
C
C FIND EMAX, THE MAXIMUM ELECTRIC FIELD FOR SCALING PURPOSES
C
    EMAX=0.0
      DO 110 I = 0,XDIV
      DO 100 J = 0,ZDIV
      IF(EX(I,J).GT.EMAX) EMAX=EX(I,J)
      IF(EZ(I,J).GT.EMAX) EMAX=EZ(I,J)
 100 CONTINUE
 110 CONTINUE
C
C PRINT THE ELECTRIC FIELDS FOR PLOTTING
C
    DO 20 I=0,XDIV
      Z=0.
      DO 15 J=0,ZDIV
      WRITE(8,90) Z,X,EZ(I,J)/EMAX,EX(I,J)/EMAX
  90 FORMAT(' ',F9.3,3X,F9.3,5X,G15.7,3X,G15.7)
  15 Z=Z-L/ZDIV
  20 X=X+A/XDIV
C
C PRINT THE RATIO OF HEIGHT TO WIDTH AND THE WINDOW LIMITS ON THE PLOT
C
    WRITE(8,*) -L,0.0,0.0,A
```

```
C
     STOP
       END
C
C ___________________________________________________________________
C SUBROUTINE FOR THE CALCULATION OF THE OHMIC HEATING RATE
C IN BETWEEN THE PLATES FOR A SHORT-CIRCUITED WAVEGUIDE
C ___________________________________________________________________
C
     SUBROUTINE SHORT(W,A,D,L,T,XDIV,ZDIV,EX1,REEL,FAKE,EX,EZ)
C
C VARIABLE DECLARATIONS
C
       INTEGER ZDIV,XDIV
       REAL L,MU,EX(0:XDIV,0:ZDIV),EZ(0:XDIV,0:ZDIV),MAGE2
       REAL SIG1,SIG2,EP1,EPS1R,EP2,EPS2R,T
       COMPLEX CMPLX,CCOS,CSIN,CSQRT,K1,K2,KZ,C1,C2
       COMPLEX MED1,MED2,KZ2,KX1,KX2,EX1,JJ,TIME
       COMPLEX TERM1,TERM2
C
       COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
C PRELIMINARY CALCULATIONS
C
       EP1=EPS1R*8.8542E-12
       EP2=EPS2R*8.8542E-12
       K1=CMPLX(0.,-W*MU)*CMPLX(SIG1,W*EP1)
       K2=CMPLX(0.,-W*MU)*CMPLX(SIG2,W*EP2)
       MED1=CMPLX(SIG1,W*EP1)
       MED2=CMPLX(SIG2,W*EP2)
       KZ=CMPLX(REEL,FAKE)
       KZ2=KZ*KZ
       KX1=CSQRT(K1-KZ2)
       KX2=CSQRT(K2-KZ2)
       JJ=CMPLX(0.0,1.0)
       TIME=CMPLX(COS(W*T),SIN(W*T))

C
C CALCULATION OF C1 AND C2 FROM THE EXCITATION FIELD
C
     TERM1=CSIN(KX1*D)/(MED1*KX1)
       TERM2=(CCOS(KX1*D)*CSIN(KX2*(A-D)))
       TERM2=TERM2/(CCOS(KX2*(A-D))*MED2*KX2)
       C1=EX1/(2*KZ2*CCOS(KZ*(-L))*(TERM1+TERM2)
       C2=C1*CCOS(KX1*D)/CCOS(KX2*(A-D))
C
C FIELD CALCULATIONS
C
       X=0.
```

```
      DO 110 I=0,XDIV
      Z=0.
      DO 100 J=0,ZDIV
      IF(X.GT.D) GO TO 10
   EX(I,J)=REAL(-JJ*2*C1*KZ2*CCOS(KX1*X)*CSIN(KZ*Z)*TIME/MED1)
      EZ(I,J)=REAL(JJ*2*C1*KX1*KZ*CSIN(KX1*X)*CCOS(KZ*Z)*TIME/MED1)
      GO TO 20
 10 EX(I,J)=REAL((-JJ*2*C2*KZ2*CCOS(KX2*(A-X))*CSIN(KZ*Z)*TIME/MED2)/30)
   EZ(I,J)=REAL((-JJ*2*C2*KX2*KZ*CSIN(KX2*(A-X))*CCOS(KZ*Z)*TIME/MED2)/30)
 20 IF(I.EQ.0) EZ(I,J)=0.0
   IF(I.EQ.XDIV) EZ(I,J)=0.0
      IF(J.EQ.0) EX(I,J)=0.0
100 Z=Z-L/ZDIV
110 X=X+A/XDIV
   RETURN
      END
C
C ____________________________________________________________________
C SUBROUTINE FOR THE CALCULATION OF THE OHMIC HEATING RATE
C BETWEEN THE PLATES FOR AN OPEN-CIRCUITED WAVEGUIDE
C ____________________________________________________________________
C
   SUBROUTINE OPENN(W,A,D,L,T,XDIV,ZDIV,EX1,REEL,FAKE,EX,EZ)
C
C DECLARATIONS
C
      INTEGER ZDIV,XDIV
      REAL L,MU,EX(0:XDIV,0:ZDIV),EZ(0:XDIV,0:ZDIV),MAGE2
      REAL SIG1,SIG2,EP1,EPS1R,EP2,EPS2R
      COMPLEX CMPLX,CCOS,CSIN,CSQRT,K1,K2,KZ,C1,C2
      COMPLEX MED1,MED2,KZ2,KX1,KX2,EX1,TIME,TERM1,TERM2
C
      COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
C PRELIMINARY CALCULATIONS
C
      EP1=EPS1R*8.8542E-12
      EP2=EPS2R*8.8542E-12
      K1=CMPLX(0.,-W*MU)*CMPLX(SIG1,W*EP1)
      K2=CMPLX(0.,-W*MU)*CMPLX(SIG2,W*EP2)
      MED1=CMPLX(SIG1,W*EP1)
      MED2=CMPLX(SIG2,W*EP2)
      KZ=CMPLX(REEL,FAKE)
      KZ2=KZ*KZ
      KX1=CSQRT(K1-KZ2)
      KX2=CSQRT(K2-KZ2)
      TIME=CMPLX(COS(W*T),SIN(W*T))
C
C CALCULATION OF C1 AND C2 FROM THE EXCITATION FIELD
```

```
C
   TERM1=CSIN(KX1*D)/(MED1*KX1)
     TERM2=(CCOS(KX1*D)*CSIN(KX2*(A-D)))
     TERM2=TERM2/(CCOS(KX2*(A-D))*MED2*KX2)
     C1=EX1/(2*KZ2*CCOS(KZ*(-L))*(TERM1+TERM2)
     C2=C1*CCOS(KX1*D)/CCOS(KX2*(A-D))
C
C FIELD CALCULATIONS
C
     X=0.
     DO 110 I=0,XDIV
     Z=0.
     DO 100 J=0,ZDIV
     IF(X.GT.D) GO TO 10
     EX(I,J)=REAL(2*C1*KZ2*CCOS(KX1*X)*CCOS(KZ*Z)*TIME/MED1)
     EZ(I,J)=REAL(2*C1*KX1*KZ*CSIN(KX1*X)*CSIN(KZ*Z)*TIME/MED1)
     GO TO 20
 10 EX(I,J)=REAL(2*C2*KZ2*CCOS(KX2*(A-X))*CCOS(KZ*Z)*TIME/MED2)
   EZ(I,J)=REAL(-2*C2*KX2*KZ*CSIN(KX2*(A-X))*CSIN(KZ*Z)*TIME/MED2)
 20 IF(I.EQ.0) EZ(I,J)=0.0
   IF(I.EQ.XDIV) EZ(I,J)=0.0
100 Z=Z-1/ZDIV
110 X=X+A/XDIV
   RETURN
     END
```

```
C Program 10. Heater
C
C
C This 2-dimensional program calculates the time-averged resistive heating
C rate over a grid in the x-z cross-section of a parallel-plate
C waveguide filled with either a homogeneous or 2-layered medium and
C terminated with an open or short-circuit. z is directed along the guide
C axis from source to load; x is directed from bottom to top plate; y is
C directed across the waveguide following a RH coordinate system.
C The origin of coordinates is located at the bottom right hand corner
C of the problem domain.
C Note: The layers of media must be parallel to the parallel plates.
C
C Data input consists of the system parameters which are clearly
C identified in the program section: 'Entry of System Parameters.'
C Note: For more information, see Figure 3. in the thesis text.
C
C Program output is written to output unit 8 and into a file called
C 'DPOWER.' Output consists of the z and x coordinate and the
C corresponding volumetric heating rate in Watts per cubic metre
C according to the following collumn format:
C
C             z(m)   x(m)   Power(W/m**3)
C
C The output is formatted to be directly read by the graphing program
C SYSTAT.
C
C ______________________________________________________________
C DECLARATION STATEMENTS AND INPUT OF PARAMETERS
C ______________________________________________________________
C
      INTEGER ZDIV,XDIV
        REAL F,A,D,REEL,FAKE,C1REAL,C1IMAG,C2REAL,C2IMAG
        REAL SIG1,SIG2,EPS1R,EPS2R,L,MU,REX1,IMEX1
        COMPLEX CMPLX,EX1
        INTEGER M
C
      COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
C ______________________________________________________________
C ENTRY OF SYSTEM PARAMETERS
C ______________________________________________________________
C
C INPUT THE NUMBER OF DIVISIONS DESIRED FOR THE PROBLEM
C DOMAIN IN BOTH THE X AND Z DIRECTIONS. ZDIV IS THE NUMBER
C OF GRID DIVISIONS ALONG THE Z-AXIS. XDIV IS THE NUMBER OF
C GRID DIVISIONS ALONG THE X-AXIS.
C
      PARAMETER(ZDIV=25,XDIV=25)
```

```
        REAL HEAT(0:XDIV,0:ZDIV)
C
C ENTER THE FREQUENCY OF OPERATION (HERTZ).
C
     PARAMETER(F=250000.0)
C
C ENTER THE PLATE SEPARATION (A) AND THE SATURATED
C REGION THICKNESS (D) (METRES)
C
     PARAMETER(A=15.0,D=12.75)
C
C ENTER THE WAVEGUIDE LENGTH (METRES)
C
     PARAMETER(L=400.0)
C
C ENTER THE REAL(REEL) AND IMAGINARY(FAKE) PARTS
C OF THE Z-PROPAGATION CONSTANT KZ=(BETA-J*ALPHA)=(REEL+J*FAKE)
C
     PARAMETER(REEL=2.208E-2,FAKE=-3.473E-3)
C
C INDICATE WHETHER THE WAVEGUIDE TERMINATION IS AN
C OPEN CIRCUIT(0) OR A SHORT CIRCUIT(1)
C
     PARAMETER(M=0)
C
C ENTER THE CONDUCTIVITIES OF REGION1 (SATURATED REGION)
C AND REGION2 (DEPLETED REGION) (SIEMENS/METRE)
C
     DATA SIG1,SIG2 /1.E-3,1.E-6/
C
C ENTER THE RELATIVE DIELECTRIC CONSTANTS OF REGION1
C AND REGION2 (DIMENSIONLESS)
C
     DATA EPS1R,EPS2R /11.0,3.0/

C
C ENTER THE APPLIED EXCITATION VOLTAGE BETWEEN THE
C PARALLEL-PLATES (MAGNITUDE AND PHASE(DEGREES))
C
     PARAMETER(EX1MAG=1000.0,EX1ANG=0.0)
C
C ENTER THE MAGNETIC PERMEABILITY OF BOTH REGIONS (H/M)
C
        PARAMETER(PI=3.141592654)
     MU = 4*PI*1.E-7
C
C________________________________________________________________
C MAIN PROGRAM BODY
C________________________________________________________________
```

```
C
    OPEN(UNIT=8,FILE='DPOWER',STATUS='NEW')
C
C CLACULATE THE EXCITATION VECTOR IN RECTANGULAR FORM
C
      REX1=EX1MAG*COS(EX1ANG*PI/180)
      IMEX1=EX1MAG*SIN(EX1ANG*PI/180)
      EX1=CMPLX(REX1,IMEX1)
C
C CALL THE APPROPRIATE SUBROUTINES DEPENDING ON WHETHER THE LOAD
C IS OPEN OR SHORT CIRCUITED
C
      W=2*PI*F
      IF(M.EQ.1) GO TO 10
      CALL OPENN(W,A,D,L,XDIV,ZDIV,EX1,REEL,FAKE,HEAT)
      GOTO 11
  10 CALL SHORT(W,A,D,L,XDIV,ZDIV,EX1,REEL,FAKE,HEAT)
  11 X=0.
C
C PRINT THE RESULTS
C
    DO 20 I=0,XDIV
      Z=0.
      DO 15 J=0,ZDIV
      WRITE(8,90) Z,X,HEAT(I,J)
  90 FORMAT(' ',F9.3,3X,F9.3,5X,G15.7)
  15 Z=Z-L/ZDIV
  20 X=X+A/XDIV
    STOP
      END
C
C ____________________________________________________________________
C SUBROUTINE FOR THE CALCULATION OF THE OHMIC HEATING RATE
C IN BETWEEN THE PLATES FOR A SHORT-CIRCUITED WAVEGUIDE
C ____________________________________________________________________
C
    SUBROUTINE SHORT(W,A,D,L,XDIV,ZDIV,EX1,REEL,FAKE,HEAT)
C
C VARIABLE DECLARATIONS
C
      INTEGER ZDIV,XDIV
      REAL L,MU,HEAT(0:XDIV,0:ZDIV),MAGE2
      REAL SIG1,SIG2,EP1,EPS1R,EP2,EPS2R
      COMPLEX CMPLX,CCOS,CSIN,CSQRT,K1,K2,KZ,C1,C2
      COMPLEX MED1,MED2,KZ2,EX,EZ,KX1,KX2,EX1,JJ
      COMPLEX TERM1,TERM2
C
      COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
```

```
C PRELIMINARY CALCULATIONS
C
     EP1=EPS1R*8.8542E-12
     EP2=EPS2R*8.8542E-12
     K1=CMPLX(0.,-W*MU)*CMPLX(SIG1,W*EP1)
     K2=CMPLX(0.,-W*MU)*CMPLX(SIG2,W*EP2)
     MED1=CMPLX(SIG1,W*EP1)
     MED2=CMPLX(SIG2,W*EP2)
     KZ=CMPLX(REEL,FAKE)
     KZ2=KZ*KZ
     KX1=CSQRT(K1-KZ2)
     KX2=CSQRT(K2-KZ2)
     JJ=CMPLX(0.0,1.0)
C
C CALCULATION OF C1 AND C2 FROM THE EXCITATION FIELD
C
   TERM1=CSIN(KX1*D)/(MED1*KX1)
     TERM2=(CCOS(KX1*D)*CSIN(KX2*(A-D)))
     TERM2=TERM2/(CCOS(KX2*(A-D))*MED2*KX2)
     C1=EX1/(2*KZ2*CCOS(KZ*(-L))*(TERM1+TERM2))
     C2=C1*CCOS(KX1*D)/CCOS(KX2*(A-D))
C
C FIELD CALCULATIONS
C
     X=0.
     DO 110 I=0,XDIV
     Z=0.
     DO 100 J=0,ZDIV
     IF(X.GT.D) GO TO 10
     EX=-JJ*2*C1*KZ2*CCOS(KX1*X)*CSIN(KZ*Z)/MED1
     EZ=JJ*2*C1*KX1*KZ*CSIN(KX1*X)*CCOS(KZ*Z)/MED1
     GO TO 20
 10 EX=-JJ*2*C2*KZ2*CCOS(KX2*(A-X))*CSIN(KZ*Z)/MED2
   EZ=-JJ*2*C2*KX2*KZ*CSIN(KX2*(A-X))*CCOS(KZ*Z)/MED2
 20 IF(I.EQ.0) EZ=CMPLX(0.,0.)
   IF(I.EQ.XDIV) EZ=CMPLX(0.,0.)
     IF(J.EQ.0) EX=CMPLX(0.,0.)
C
C POWER DISSIPATION CALCULATIONS
C
   MAGE2=(CABS(EX)**2)+(CABS(EZ)**2)
     IF(X.GT.D) GOTO 50
     HEAT(I,J)=SIG1*MAGE2
     GOTO 100
 50 HEAT(I,J)=SIG2*MAGE2
100 Z=Z-L/ZDIV
110 X=X+A/XDIV
   RETURN
     END
```

```
C
C ------------------------------------------------------------------------
C SUBROUTINE FOR THE CALCULATION OF THE OHMIC HEATING RATE
C BETWEEN THE PLATES FOR AN OPEN-CIRCUITED WAVEGUIDE
C ------------------------------------------------------------------------
C
   SUBROUTINE OPENN(W,A,D,L,XDIV,ZDIV,EX1,REEL,FAKE,HEAT)
C
C DECLARATIONS
C
      INTEGER ZDIV,XDIV
      REAL L,MU,HEAT(0:XDIV,0:ZDIV),MAGE2
      REAL SIG1,SIG2,EP1,EPS1R,EP2,EPS2R
      COMPLEX CMPLX,CCOS,CSIN,CSQRT,K1,K2,KZ,C1,C2
      COMPLEX MED1,MED2,KZ2,EX,EZ,KX1,KX2,EX1
      COMPLEX TERM1,TERM2
C
      COMMON SIG1,SIG2,EPS1R,EPS2R,MU
C
C PRELIMINARY CALCULATIONS
C
      EP1=EPS1R*8.8542E-12
      EP2=EPS2R*8.8542E-12
      K1=CMPLX(0.,-W*MU)*CMPLX(SIG1,W*EP1)
      K2=CMPLX(0.,-W*MU)*CMPLX(SIG2,W*EP2)
      MED1=CMPLX(SIG1,W*EP1)
      MED2=CMPLX(SIG2,W*EP2)
      KZ=CMPLX(REEL,FAKE)
      KZ2=KZ*KZ
      KX1=CSQRT(K1-KZ2)
      KX2=CSQRT(K2-KZ2)
C
C CALCULATION OF C1 AND C2 FROM THE EXCITATION FIELD
C
   TERM1=CSIN(KX1*D)/(MED1*KX1)
      TERM2=(CCOS(KX1*D)*CSIN(KX2*(A-D)))
      TERM2=TERM2/(CCOS(KX2*(A-D))*MED2*KX2)
      C1=EX1/(2*KZ2*CCOS(KZ*(-L))*(TERM1+TERM2))
      C2=C1*CCOS(KX1*D)/CCOS(KX2*(A-D))
C
C FIELD CALCULATIONS
C
      X=0.
      DO 110 I=0,XDIV
      Z=0.
      DO 100 J=0,ZDIV
      IF(X.GT.D) GO TO 10
      EX=2*C1*KZ2*CCOS(KX1*X)*CCOS(KZ*Z)/MED1
      EZ=2*C1*KX1*KZ*CSIN(KX1*X)*CSIN(KZ*Z)/MED1
      GO TO 20
 10 EX=2*C2*KZ2*CCOS(KX2*(A-X))*CCOS(KZ*Z)/MED2
  EZ=-2*C2*KX2*KZ*CSIN(KX2*(A-X))*CSIN(KZ*Z)/MED2
 20 IF(I.EQ.0) EZ=CMPLX(0.,0.)
  IF(I.EQ.XDIV) EZ=CMPLX(0.,0.)
C
C POWER DISSIPATION CALCULATIONS
C
      MAGE2=(CABS(EX)**2)+(CABS(EZ)**2)
      IF(X.GT.D) GOTO 50
      HEAT(I,J)=SIG1*MAGE2
      GOTO 100
 50 HEAT(I,J)=SIG2*MAGE2
 100 Z=Z-L/ZDIV
 110 X=X+A/XDIV
  RETURN
      END
```