THE MEASUREMENT OF COMPLEX RELATIVE PERMITTIVITY OF SOIL BY USING THE TOTAL REFLECTION TIME DOMAIN REFLECTOMETRY

by

Tilahun Mengesha Worku

A thesis
presented to the University of Manitoba
in fulfilment of the
thesis requirement for the degree of
Master of Science
in
Agricultural Engineering

Winnipeg, Manitoba, Canada 1995

ISBN 0-612-13574-8

Canada

Name _____

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Electronics and Electrical
SUBJECT TERM

0544
SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology ... 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology .. 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics ... 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

THE MEASUREMENT OF COMPLEX RELATIVE PERMITTIVITY OF SOIL BY USING THE TOTAL REFLECTION TIME DOMAIN REFLECTOMETRY

BY

TILAHUN MENGESHA WORKU

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Abstract

A method for the measurement of complex relative permittivity of soil by using the total reflection time domain reflectometry is developed. The basic theoretical and practical basis of time domain reflectometry is reviewed. The performance of the method is tested by carrying out measurements on five types of soils with a range of texture from sand to clay. The effects of soil temperature and soil electrical conductivity on the complex relative permittivity of soil are investigated. Empirical relationships which may be used to predict volumetric water content of a soil from the real part of the complex relative permittivity of the soil determined at 10, 50, 100, or 150 MHz are given. Both the real and the imaginary parts of the complex relative permittivity of all the five soils were observed to decrease with a decrease in volumetric water content and with an increase in frequency. The effects of soil temperature and soil electrical conductivity were observed to be higher on the imaginary part of the complex relative permittivity as compared with that on the real part of the complex relative permittivity. The results obtained agree very well with other experimenters' results. It was concluded that the method was successful in measuring the complex relative permittivity of soil.

Acknowledgments

I wish to express my gratitude to my thesis advisor Dr. R. Sri Ranjan for his guidance and assistance during the course of this study.

I would like to thank Dr. J. S. Townsend for reading this thesis and supplying constructive comments.

Special thanks to Dr. G. E. Bridges for his critical evaluation of my thesis, for providing the Tektronix 7854 mainframe oscilloscope for my experiments and for his co-operation in coordinating the use of the oscilloscope in his laboratory.

I thank Messrs Matt McDonald, Rob Ataman, and Jack Putnam for their excellent technical support.

Financial support of the Alemaya University of Agriculture, Ethiopia, for two academic years and the Department of Agricultural Engineering, University of Manitoba, for three months is greatly acknowledged.

I dedicated this thesis to my mother and to the rest of my family whose undying love and moral support placed me on the road to learing.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

It is a fundamental experimental result, first discovered by Faraday, that capacitance of a condenser increased when the space between the conductor is filled with a dielectric material. If C_o is the capacitance of the condenser with the region between the conductors evacuated and C its capacitance when this region is filled with a dielectric then the ratio $\frac{C}{C_o} = \epsilon$ is found to be independent of the shape or the dimensions of the conductors and is solely a characteristic of the particular dielectric medium used. The ratio ϵ is called the *relative permittivity* of the medium. Ever since Faraday's discovery, scientists have been interested in the study of the dielectric properties of materials.

The term relative permittivity is commonly abbreviated as permittivity and it is never less than one. For gases the relative permittivity is close to unity. The term dielectric constant is also sometimes used for relative permittivity and is somewhat misleading when it is applied to polar materials. The relative permittivity of polar materials, such as water, shows frequency dependency in alternating electric fields. When a polar material is subjected to an electric field, dipoles in the material align or orient themselves with the electric field. In an alternating electric field, the

orientation of the dipoles will tend to reverse as the polarity of the field changes. If the rate of change in the direction of the electric field is sufficiently high, molecular forces impeding the motion of dipoles dominates. In a certain frequency band, referred to as the *relaxation band*, a phase lag develops between the electric field and the dipole orientation. In this case, energy from the electric field is dissipated as heat. This relaxation phenomenon is described by a *complex relative permittivity*, $\epsilon(f)$

$$\epsilon(f) = \epsilon'(f) - j(\epsilon''(f) + \frac{\sigma_{dc}}{2\pi f \epsilon_o}) \tag{1.1}$$

in which $\epsilon'(f)$ is the real part and $(\epsilon''(f) + \frac{\sigma_{dc}}{2\pi f \epsilon_o})$ is the imaginary part. The real part of the complex relative permittivity indicates the extent of the polarization of the dielectric under the influence of the applied electric field, as explained in Section 2.3, and the imaginary part represents the dielectric loss and hence an absorption of energy. In the equation, σ_{dc} $[dS\ m^{-1}]$ accounts for the electrical conduction which arises not from the effect of polarization but from actual charge transport (for example, ionic conductivity in electrolytes). The frequency of the applied electric field is denoted by f $[Hz]$.

Water exhibits a special dielectric behavior which distinguishes it from most other materials. The static relative permittivity and the relaxation frequency of water are 78.3 and 19.7 GHz, respectively, (Grant et al., 1989) and are relatively high when compared with that of other materials. There are a number of applications which utilize this special dielectric behavior of water. In mircowave ovens, heat is generated by electromagnetic waves (microwaves) with a frequency which falls within the relaxation band of water. Another application which uses the special dielectric properties of water is the indirect measurement of volumetric water content of porous substances such as soil. The success of these dielectric measurement techniques lies in the fact that either the electromagnetic waves are used at the relaxation frequencies

of water or the measurements occur at frequencies where the permittivity of water is much higher than that of other material in the porous mixture.

The dielectric properties of soils are of considerable practical and theoretical interest. The main practical interest stems from the desire to use dielectric properties of soils as a sensitive and accurate indicator of soil water content (Campbell, 1990).

Soil water content is a fundamental variable in many processes and its measurement is important in many types of studies (Baker and Allmaras, 1992). The measurement of soil water content in the field environment generally depends on gravimetric sampling, installation of resistance blocks, or access tubes for the neutron probe. All of these methods have shortcomings which make them unsuitable for field use. The destructive nature of the sampling and the time delay for drying of the sample are the drawbacks of the gravimetric method. The requirement to calibrate resistance blocks for each site seriously limits their utility in varying environments. The need for an access tube for the measurement and a calibration curve for each situation, and the radiation hazard involved causes difficulty for many operators of the neutron method.

The measurement of volumetric water content of the soil by using Time Domain Reflectometry, TDR, is a relatively new technique. It was first introduced in the early 1980s and was shown to have several favorable attributes. Most of the experimental and theoretical advances in the use of this method were initiated by Topp et al. (1980). The determination of volumetric water content of a soil by TDR is, generally, based on the measurement of travel time of an electromagnetic pulse along a probe inserted in the soil by using time domain reflectometer. Theoretical analysis and experimental correlations show that the pulse travel time is proportional to the apparent dielectric constant, ϵ_a, of the soil in which the probe is inserted. Topp et al. (1980) measured this pulse travel time, t, in a probe of known

length, L, and correlated the calculatedvalues of the apparent dielectric constant of soil with the volumetric water content, Θ, and obtained:

$$\Theta = -0.530 \times 10^{-3} + 0.292 \times 10^{-3}\epsilon_a - 0.550 \times 10^{-5}{\epsilon_a}^2 + 0.430 \times 10^{-7}{\epsilon_a}^3 \quad (1.2)$$

with

$$\epsilon_a = \frac{Ct}{2L} \quad (1.3)$$

where C is the velocity of light in a vacuum. They showed this relationship to be similar for a wide range of mineral soils and independent of soil type, soil density, soil temperature, and soil salinity.

Recently several exceptions to this relationship have been reported by different researchers. Drungil et al. (1987) carried out calibrations for different types of soils and reported that the $\Theta(\epsilon_a)$ relationship was valid only for gravelly soils. Dalton et al., (1992), showed the dependency of the $\Theta(\epsilon_a)$ relationship on soil salinity. Moreover, Dobson et al., (1985) found a dependency of the $\Theta(\epsilon_a)$ relationship on soil texture. They showed that fine textured soils have significantly lower dielectric values when compared with coarse textured soils at the same water content, a difference that increased with water content. All these exceptions call for an alternative or new approach for the measurement of dielectric properties of soil.

Wet materials, such as soil, are generally dispersive, their permittivities are frequency dependent. In the travel–time approach the measurement is frequency independent and it is unknown to what frequency or frequency ranges such a determination of ϵ_a would correspond. In time domain reflectometry, the measurement of wave forms is conducted in the time domain. The analysis, however, will be simple and the result will be frequency dependent and broadband if the wave forms observed in time domain are first Fourier transformed and analyzed in the frequency domain.

The objectives of this thesis research are:

1. To develop a method for the measurement of complex relative permittivity of soil by using the total reflection time domain reflectometry and the frequency domain analysis,

2. To evaluate the performance of the method, and

3. To develop a calibration equation which may be used for predicting volumetric water content of a soil from the complex relative permittivity of the soil determined by total reflection TDR.

The measurement of complex relative permittivity of the soil, in this study, is based on measuring the forward scattering function, $S11(f)$, of a specially designed coaxial probe inserted into the soil. The $S11(f)$ function is computed from the ratio of a discrete Fourier transform of the total reflected wave form from the probe to the incident wave form into the probe.

This thesis consists of six chapters. In Chapter 2, a review of the theoretical basis of time domain reflectometry is presented. In Chapter 3, the sources of errors in the total reflection time domain reflectometry are identified and methods for their minimization or elimination are presented. The experimental apparatus used and the soil samples tested, the design of the probe constructed, and the experimental procedures implemented are described in Chapter 4. The results are given and discussed in Chapter 5. A summary of the work presented in this thesis is given and general conclusions are drawn in Chapter 6.

Chapter 2

Theory

2.1 Introduction

In time domain reflectometry a sample of the material to be tested is inserted into a section of a coaxial line system. The complex relative permittivity spectrum of the material is measured by analyzing the response of the material to a fast rise time step voltage. This is normally done by using Fourier analysis and transmission line theory.

In this chapter, the theoretical basis of TDR is discussed. This is done in order to provide a background for the error analysis given in Chapter 3 and the experimental methods presented Chapter 4. In Section 2.2, the reflection and transmission of a wave by a change in the characteristic impedance of a finite section is considered. The microscopic concept of polarization and frequency dependency of permittivity are discussed in Section 2.3 and Section 2.4, respectively.

The method selected for the purpose of measurement of complex relative permittivity of the soils in this study, as explained in Chapetr 4, requires forming the Fourier transform of the incident and the reflected wave forms. For the benefit of

the reader of this thesis, a brief introduction to the Fourier integral and the discrete Fourier transformation is presented in Section 2.6. Finally, the results of the previous discussions are used to describe the general principles of TDR, the single reflection TDR and the total reflection TDR (Section 2.7).

2.2 Multiple reflection within a coaxial line

If a wave propagating along a coaxial line encounters a change in the characteristic impedance of the line, part of the wave will be transmitted and part will be reflected as shown in Figure 2.1. Let the voltage of the incident wave, the reflected wave, and the transmitted wave at the discontinuity be denoted by V_i, V_r and V_t, respectively, and the corresponding currents be denoted by I_i, I_r and I_t, respectively. It then follows that

$$V_i + V_r = V_t \tag{2.1}$$

and

$$I_i + I_r = I_t \tag{2.2}$$

If the characteristic impedances of the two sections of the coaxial line are denoted by Z_1 and Z_2, as shown in Figure 2.1, the current and voltage are then related by,

$$V_i = Z_1 I_i \ , \qquad V_t = Z_2 I_t \qquad and \qquad V_r = -Z_1 I_r \tag{2.3}$$

Using these relationships, equation (2.2) can be rewritten in terms of the voltage as

$$\frac{V_i}{Z_1} - \frac{V_r}{Z_1} = \frac{V_t}{Z_2} \tag{2.4}$$

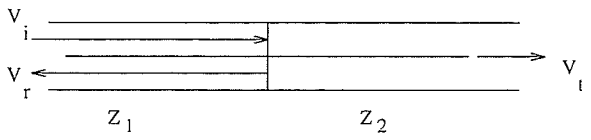

Figure 2.1: Reflection and transmission of a wave at a change in the characteristic impedance of a coaxial line.

Equations (2.1) and (2.4) can then be solved for V_r and V_t in terms of the characteristic impedances. This gives

$$V_t = \frac{2Z_2}{Z_2 + Z_1} V_i \tag{2.5}$$

and

$$V_r = \frac{Z_2 - Z_1}{Z_2 + Z_1} V_i \tag{2.6}$$

The reflection coefficient ρ_{12} and the transmission coefficient τ_{12} for the discontinuity are defined to be equal to $\frac{V_r}{V_i}$ and $\frac{V_t}{V_i}$, respectively. Using equations (2.5) and (2.6), ρ_{12} and τ_{12} can be expressed in terms of the characteristic impedance as

$$\rho_{12} = \frac{Z_2 - Z_1}{Z_2 + Z_1} \tag{2.7}$$

and

$$\tau_{12} = \frac{2Z_2}{Z_2 + Z_1} \tag{2.8}$$

For a wave propagating along the coaxial line in the opposite direction, the reflection coefficient ρ_{21} and the transmission coefficient τ_{21} can be expressed in terms of ρ_{12} and τ_{12} by

$$\rho_{21} = -\rho_{12}\,; \qquad \tau_{12}\tau_{21} = 1 - {\rho_{12}}^2 \tag{2.9}$$

If the characteristic impedance of a finite section of a coaxial line is changed, the incident wave will undergo multiple reflections from the two impedance discontinuities

as shown in Figure 2.2. If the voltage of the incident wave and the total reflected wave at the first discontinuity (section A–A) are denoted, respectively, by V_i and V_{tr}, and the voltage of the totally transmitted wave at the second discontinuity (section B–B) is denoted by V_{tl} then it is customary to write

$$V_{tr} = S11V_i \tag{2.10}$$

and

$$V_{tl} = S21V_i \tag{2.11}$$

where $S11$ and $S21$ are referred to as the *forward scattering coefficient* and the *reverse scattering coefficient*, respectively.

From Figure 2.2, $S11(f)$ is given to n terms by,

$$S11(f) = \rho_{12}+\tau_{12}\tau_{21}\rho_{21}e^{-2\gamma l}+\tau_{12}\tau_{21}\rho_{21}{}^{3}e^{-4\gamma l}+\cdots+\tau_{12}\tau_{21}\rho_{21}{}^{2n-3}e^{-2(n-1)\gamma l}+\cdots \tag{2.12}$$

where l is the length of the section of the coaxial line with propagation factor γ and γ is equal to $2\pi jf\sqrt{\epsilon(f)}/C$ where f is the frequency, $\epsilon(f)$ is the complex relative permittivity, C is speed of light in vacuum, and j is $\sqrt{-1}$. Substituting $\rho_{21} = -\rho_{21}$ and $\tau_{12}\tau_{21} = 1-\rho^2$ and summing equation (2.12) as $n \to \infty$ gives (Clark et al., 1974)

$$S11(f) = \rho\frac{1-e^{-2\gamma l}}{1-\rho^2e^{-2\gamma l}} \tag{2.13}$$

in which

$$\rho = \rho_{12} = \frac{1-\sqrt{\epsilon(f)}}{1+\sqrt{\epsilon(f)}} \tag{2.14}$$

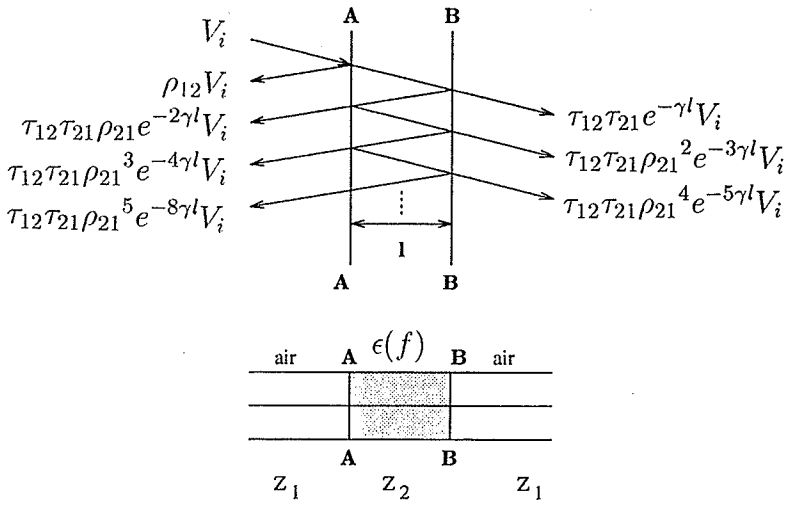

Figure 2.2: Multiple reflection produced by a change in the characteristic impedance of a finite section of a coaxial line.

In a similar way, for a coaxial terminated by an open circuit at section B-B, $S11(f)$ is given by (Clarkson et al., 1977)

$$S11(f) = \frac{\rho + e^{-2\gamma l}}{1 + \rho e^{-2\gamma l}} \tag{2.15}$$

with ρ as defined above.

Figure 2.3 shows a coaxial cable having two sections with different dimensions. Let the characteristic impedance of section 1 be Z_1 and that of section 2 be Z_2. In this case the expression for $S11(f)$ takes the following form (Clarkson, et al., 1977)

$$S11(f) = \frac{\rho + e^{-2\gamma l}}{1 + \rho e^{-2\gamma l}} \tag{2.16}$$

with

$$\rho = \frac{1 - z\sqrt{\epsilon(f)}}{1 + z\sqrt{\epsilon(f)}} \tag{2.17}$$

and

$$z = \frac{Z_1}{Z_2} \tag{2.18}$$

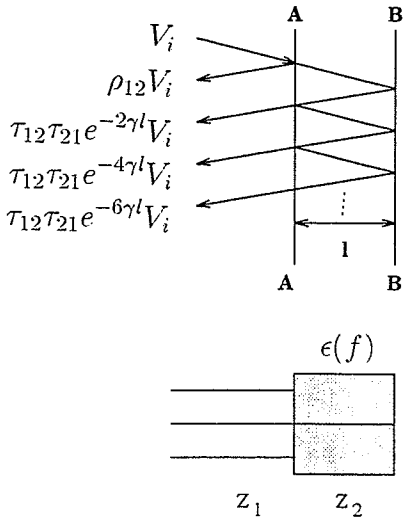

Figure 2.3: Multiple reflection produced by a change in the characteristic impedance resulted due to the change in geometry and the dielectric of a finite section at the open end of a coaxial line.

2.3 The microscopic concept of polarization

When a dielectric is subjected to an external electric field, an interaction, called *polarization*, takes place between the electric field and the molecules, atoms, and ions of the dielectric. The overall result of this interaction which occurs on a microscopic scale is expressed quantitatively by the permittivity of the dielectric. There are four types of polarizations (Smyth, 1955). In this section, these four types of polarizations are briefly explained.

The first type of polarization is electronic polarization. An atom is composed of a positively charged nucleus surrounded by symmetrically distributed electron clouds. When an isolated neutral atom is subjected to an electric field, the electron clouds are displaced slightly with respect to the positive nucleus causing the atom to acquire a dipole moment. When this happens, the atom is said to be *polarized* and the

polarization is known as *electronic polarization.* The electronic polarization of the atom P_e is proportional to the applied electric field strength (E) and is given by,

$$P_e = \alpha_e E \tag{2.19}$$

where the proportionality constant, α_e, is the *electronic polarizability* of the atom. In most organic solids, this type of polarization leads to a relative permittivity of about 1.8 to 4.0 (Pohl, 1916).

The second type of polarization is *atomic polarization* which arises from the shift of differently charged atoms with respect to each other. It is observed only in ionic molecules and a well known example is that of halite, $NaCl$, whose molecules are formed from atoms having excess charges of opposite polarities. When a halite molecule is subjected to an external electric field, the electric field shifts the positive sodium ion relative to the negative chloride ion thus inducing a dipole moment. The atomic polarization is give by,

$$P_a = \alpha_a E \tag{2.20}$$

where α_a is the *atomic polarizability of the molecule.* In halite, the atomic polarization contibutes about 15% to the total polarization and hence to the relative permittivity.

The third type of polarization is *orientational polarization* and is associated with polar molecules such as H_2O, HF, NO, *etc.* A polar molecule is one which has a dipole moment in the absence of an applied electric field. In the presence of an external electric field the dipolar molecules experience a torque tending to orient them in the direction of the electric field. The orientational polarization is given by,

$$P_o = \alpha_o E \tag{2.21}$$

where α_o is the *orientational polarizability* of the molecule. This type of polarization plays an important part in low–, medium–, and high–frequency polarization and hence the permittivity of materials containing polar molecules. Liquid water, for example,

has a relative permittivity of 80 at room temperature mainly due to orientational polarization (Pohl, 1916)

The last type of polarization is *interfacial polarization* and is associated with heterogeneous substances. A typical example is moist soil. Interfacial polarization arises only when two or more phases differ from each other in relative permittivity and conductivity. Due to the difference in electrical conductivity, charges move through the better conducting phases and build up on the surface that separates the phases from the resistive phase. As a result of this, each conducting phase becomes polarized. The interfacial polarization is given by,

$$P_i = \alpha_i E \tag{2.22}$$

where α_i is *interfacial polarizability.*

In conclusion, the total polarization is the sum of the polarizations due to the four types of polarizations. The relative permittivity ($\epsilon(f)$), which in this case is equal to $\epsilon'(f)$ as expalined in Section 1.1, is a function of the total polarizability ($\alpha_T = \alpha_e + \alpha_a + \alpha_o + \alpha_i$) and is given by,

$$\epsilon(f) = 1 + \frac{4\pi N \alpha_T F}{kE} \tag{2.23}$$

where k is a constant whose value depends on the system of units, E is the applied electric field, F is the average electric field acting on a molecule, and N is the number of molecule per unit volume (Hill, 1969). This equation will be used in the later parts of this thesis in the interpretation of results obtained in the tests conducted to investigate frequency dependencey of complex relative permittivity of soil.

2.4 Frequency dependency of permittivity

Each of the above four polarization types is a function of the frequency of the

applied electric field. This is because of the difference in the time taken by the different types of polarization to reach an equilibrium state.

In a dielectric material subjected to a static electric field, the polarization is in equilibrium with the field. But in an alternating electric field, the polarization will tend to change every time the polarity of the field changes. When the frequency of the applied field is sufficiently low, all types of polarization can reach the value they would have had in a static electric field. As the frequency is raised in an alternating electric field, the polarization will no longer have time to reach its static value. The first to be affected will be interfacial polarization because this type of polarization requires time in the order of seconds or minutes to reach equilibrium. When the frequency is further raised polar molecules, if present in the dielectric, will be unable to completely follow the field and the contribution to the total polarization from orienational polarization ceases. As a result, the total polarizability falls from $\alpha_T = \alpha_e + \alpha_a + \alpha_o$ to $\alpha_T = \alpha_e + \alpha_a$. This usually occurs in the radio frequency range of the electromagnetic spectrum.

As the frequency is further increased to the infrared range, the relatively heavy positive and negative ions cannot follow the field variation so that the contribution to the permittivity from the atomic polarization ceases and only the electronic polarization remains.

The above effects lead to a fall in the total polarizability and, as a result, in the permittivity with increasing frequency, a phenomenon which is usually referred to as *dielectric dispersion.* Dispersions arising during the transition from full orientational polarization at zero frequency to negligible orientational polarization in radio frequencies is referred to as *dielectric relaxation.*

Various models for frequency dependency of relative permittivity for polar substances (such as water) have been developed. One such model is the modification

of Debye's relaxation equation due to Cole and Cole (1941)

$$\epsilon(f) = \epsilon_\infty + \frac{(\epsilon_s - \epsilon_\infty)}{[1 + (j\frac{f}{f_{rx}})^{1-\alpha}]} - \frac{j\sigma_{dc}}{2\pi f \epsilon_o} \tag{2.24}$$

where ϵ_∞ is the high frequency permittivity, ϵ_s is the static permittivity, f_{rx} is the relaxation frequency of the material, ϵ_o is permittivity of free space, σ_{dc} is low frequency conductivity of the material, α is a parameter indicating the distribution of the relaxation frequency of the material, and j is $\sqrt{-1}$. For materials with a simple relaxation time such as water $\alpha = 0$, and the above equation reduces to the form given by Debye (1929). The effect of $\alpha > 0$ is to make the dispersion of the complex relative permittivity more broadband (Hoekstra and Delaney, 1974). Figure 2.4 shows the complex relative permittivity of ethanol as a function of frequency in the frequency. The complex relative permittivity of ethanol was computed by using equation 2.24 and the Cole-Cole parameters of ethanol given in Section 4.4.

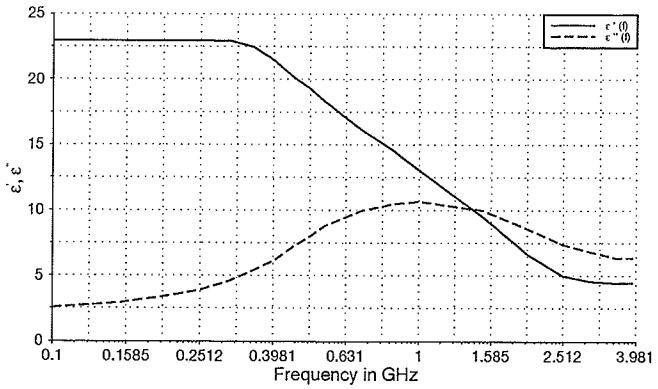

Figure 2.4: Complex relative permittivity of ethanol as a function of frequency.

2.5 Development of time domain reflectometry

Dielectric relaxation has long been recognized as a powerful method for studying the physical state of a system containing polar molecules (Suggett et al., 1970). The phenomenon of dielectric relaxation is studied conventionally by measuring the complex relative permittivity of each material. A wide variety of methods exist for the measurement of complex relative permittivity, the particular method adopted being determined by the nature of the specimen, the frequency range in which the measurement is made, and the permittivity data to be expected. At the present time, methods for the measurement of complex permittivity may be divided into two categories, namely, time domain and frequency domain. In this section, a brief survey of the historical development of one of the time domain methods, called *time domain reflectometry* will be presented.

Before focusing on time domain reflectometry, it is worth considering some of the frequency domain methods. The frequency range which can be covered by dielectric spectroscopy is extremely wide, approximately up to $100\,GHz$. At frequencies up to $100\,MHz$, the complex relative permittivity is normally determined by measuring the change in the impedance of a capacitor produced by inserting a sample of the material between the electrods. In the frequency range between $100\,MHz$ and $1\,GHz$, transmission line techniques based on standing-wave measurements are generally employed (Sheppard, 1972). For complex relative permittivity measurements at frequencies above $1\,GHz$, cavity resonator techniques are usually employed.

In all of the frequency domain methods, the complex relative permittivity is measured at each frequency separately. As a result of this, the determination of the complex relative permittivity over a wide frequency range can be lengthy and tedious. In addition, for measurement at microwave frequencies this approach requires a large investment in expensive equipment. One can measure the complex

relative permittivity over a wide frequency range in a relatively short time by making the measurements not in frequency domain but in time domain, using a steplike pulse that simultaniously contain all the frequencies of interest (Iskander, 1972). A brief comparesion between time domain and frequency domain methods in the frequency range of 10 MHz and 10 GHz is given in Table 2.1.

Frequency domain methods	*Time domain methods*
Expensive equipment (For example $HP8510A$, 45 MHz to 110 GHz Network Analayzer about $\$55,742$ US in 1994	Relatively inexpensive equipment (For example $Tektronix$ $1503B$, TDR system about $\$9,449$ US in 1994
$\pm 1\%$ uncertainity in the measurement of the individual values of the real and the imaginary parts of the complex relative permittivity	$\pm 5\%$ uncertainity in the measurement of the individual values of the real and the imaginary parts of the complex relative permittivity
Time consuming experiments (usually several hours for microwave frequencies)	Measurements are quick (Usually less than 5 min)

Table 2.1: Comparison between time domain and frequency domain methods of measurement of complex relative permittivity.

In the field of dielectric studies, the potential of time domain reflectometry was first realized by Fellner–Feldegg (1969). In TDR a sample of the material, whose

relative permittivity is to be determined, is inserted into a coaxial line and the multiple reflected signals from the sample and the step signal applied to the sample are measured by means of a sampling oscilloscope. The complex relative permittivity of the sample is obtained by Fourier analysis of the two signals. As an alternative to measuring the multiple reflected signals, it is also possible to measure just the first reflected signal. This is accomplished by making the sample length long enough to delay the arrival of the second reflected signal until after the first reflected signal has been completely measured. As a result of these different alternatives, a number of separate *single reflection* and *multiple reflection* TDR techniques have been developed.

The first single reflection technique was developed by Fellner–Feldegg (1969). Fellner–Feldegg used this technique to determine the static permittivity, the high frequency permittivity and the relaxation time of a number of alkyl alcohols. The accuracy of this technique was later improved by Sugget et al. (1970), and Loeb et al. (1971).

The first multiple reflection technique was developed by Nicolson and Ross (1970). In this technique two response signals from the sample were considered, the multiple reflected and the multiple transmitted signals. Nicolson and Ross showed that after Fourier transformation of these two signals and the incident signal, the complex relative permittivity of the sample could be determined directly from the calculated values of the total reflection coefficient, $S11(f)$, and the total transmission coefficient, $S21(f)$. In this technique, in contrast with the single reflection technique in which it was necessary to use a relatively long sample, there were no restrictions on the sample length that could be used.

An exact approach to multiple reflection TDR was developed by De Loor et al. (1973). In this technique, which they referred to as *Total Reflection TDR*, the multiple reflected signal from the sample terminated by a short circuit was considered

and the Newton–Raphson procedure was used to determine the value of the complex relative permittivity from the calculated values of $S11(f)$.

A number of papers on the experimental aspect of TDR has been presented. Nicolson (1973) reported a method for the minimization of truncation error in discrete Fourier transformation of TDR wave forms. Claason and van Gemeret (1975) derived a number of expressions to enable the determination of complex relative permittivity directly, eliminating the need for numerical Fourier transformation. Giese and Tiemann (1975) presented a correction technique for reducing the contribution from unwanted reflections. Suggett (1975) devised a number of methods for improving the high frequency performance of the single reflection method. Clarksen et al. (1977) discussed the experimental factors involved in TDR. Gestblom and Noreland (1977) considered the single transmission and multiple transmission method and discussed the advantage of these method over the corresponding reflection methods. Stuchly and Matuszewski (1978) presented combined reflection/transmission methods. Kaatze and Giese (1980) presented a review article on the relative merits and limitations of the different techniques of measurement of complex relative permittivity. Gabriel et al. (1986) discussed the use of TDR for the measurement of complex relative permittivity of human skin. Arcone and Wills (1986) presented the effect of incorrect time alignment and truncation error in single reflection TDR measurements.

2.6 Fourier analysis

2.6.1 Fourier's integral theorem

If the integral

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-j\omega t}dt \tag{2.25}$$

of the function f(t) of the real variable t exists, it defines the *Fourier transform of* $f(t)$. The basis of Fourier analysis is Fourier's integral theorem. This states that under certain general conditions, the function $f(t)$ can be written as the integral

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{j\omega t} d\omega \tag{2.26}$$

where $F(\omega)$ is as defined by equation (2.25). This equation is known as the *inverse Fourier transform.* Equation (2.25) and equation (2.26) are referred to as *the Fourier transform pair.*

A physical process in a linear system can be described either in time domain, by values of some quantity f as a function of time t, $f(t)$, or else in frequency domain, where the process is specified by giving its amplitude F as a function of frequency, $F(\omega)$. One goes back and forth between time domain and frequency domain by means of the Fourier transform pair (Press et al., 1988). Equation (2.25) transforms a function $f(t)$ in time domain into its equivalent function $F(\omega)$ in frequency domain, and equation (2.26) reverses the process. In other words, equation (2.25) analyzes the time function into the frequency spectrum, and equation (2.26) synthesizes the frequency spectrum to regain the time function.

2.6.2 Discrete Fourier transform

The practical use of the Fourier transform is based on the sampling theorem. The sampling theorem states that *if a continuous bandwidth limited signal* $f(t)$ *contains no frequency component higher than* ω_c *radians per second, then f(t) can be completely determined by its values sampled at uniform time interval less than* $T = \frac{\pi}{\omega_c}$ *seconds apart* . It can be shown that, the Fourier transform $F(\omega)$ of $f(t)$ is related to the equally spaced values of $f(t)$ by (Chen, 1979)

$$\sum_{n=-\infty}^{+\infty} F(\omega + n\omega_c) = T \sum_{n=-\infty}^{+\infty} f(nT) e^{-j\omega nT} \tag{2.27}$$

Thus, if $F(\omega) = 0$ for $|\omega| > \omega_c$, it then follows that

$$F(\omega) = T \sum_{n=-\infty}^{+\infty} f(nT) e^{-j\omega nT} \tag{2.28}$$

This equation is the basis of the practical implementation of the Fourier transform and defines the *discrete Fourier transform* (DFT).

2.7 Time domain reflectometry

2.7.1 General principles

The basic setup for the measurement of complex relative permittivity by TDR is shown in Figure 2.5. The three main components of the measurement system are a subnanosecond rise time pulse generator, an oscilloscope and a sampler. During the measurement, a sample is placed in a section of a coaxial line. A fast rise time step pulse is generated at the pulse generator and applied to the sample and the response signal is measured by means of a sampling oscilloscope. Upon the application of a step pulse, partial reflection and partial transmission occur at the front and rear faces of the sample. These multiple reflections are as shown in Figure 2.2.

Two different types of analysis have been used in TDR, namely, frequency domain analysis and time domain analysis. In the frequency domain analysis, the input and the response signals are first Fourier transformed and then the complex relative permittivity values are determined from the calculated values of the reflection or transmission coefficients directly or indirectly by using numerical methods. In the time domain analysis, the response signal is analyzed directly. This is done, in general, by first assuming an ideal step signal and then assuming a particular type of ideal behavior for the frequency dependence of the complex relative permittivity of the material such as Debye's relaxation equation. Although the frequency domain

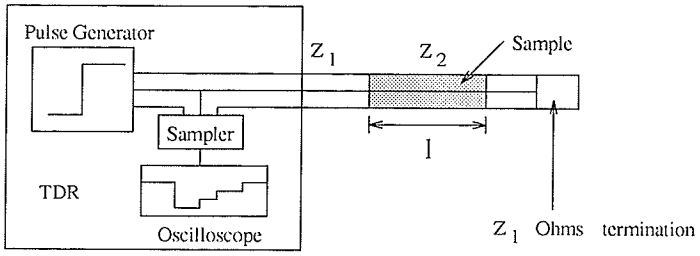

Figure 2.5: A basic experimental arrangement for the measurement of complex permittvity by TDR.

analysis is exact, the time domain analysis enables the signals to be analyzed rapidly and also eliminates the need for numerical Fourier transformation.

2.7.2 Single reflection TDR

This technique was first proposed by Fellner-Feldegg (1969). But the details of the Single reflection TDR, using an exact frequency domain analysis, were given by Loeb et al. (1971). It is the simplest form of TDR in which only the first reflection at the front face of the sample is monitored. The time range of the oscilloscope and the length of the dielectric filled section of the coaxial cable are selected to take into account only the signal due to the first reflection. The time-window, therefore, extends only up to a point in time at which interference from the next reflection from the rear face of the sample occurs. The reflection coefficient $\rho_{12}(f)$ at each frequency of interest is computed by using the Fourier transform of the incident signal, $V_i(f)$, and the first reflected signal, $V_{r1}(f)$, as

$$\rho_{12}(f) = \frac{V_{r1}(f)}{V_i(f)} \tag{2.29}$$

Knowing $\rho_{12}(f)$, the complex relative permittivity of the sample is obtained by

$$\epsilon(f) = [\frac{1-\rho_{12}(f)}{1+\rho_{12}(f)}]^2 \tag{2.30}$$

In the single reflection method, the measurement of the first reflection is achieved by delaying the arrival of the reflection from the rear face of the sample by using a relatively long sample so that the reflected signal arrives outside the time-window of interest. In the measurement of the complex relative permittivity of materials which are available in limited quantities (where long sample can not be prepared), the total reflection TDR is employed.

2.7.3 Total reflection TDR

In the total reflection method, the time scale and the length of the sample are selected so that all reflections (sum of all multiple reflections) from the front and rear faces of the sample are taken into account. The time window extends up to a point in time at which the time response of the material is complete. The total reflection coefficient, $S11(f)$, is calculated as a ratio of the Fourier transform of the total reflected signal, $V_{rt}(f)$, to the Fourier transform of the incident signal, $V_i(f)$, *i.e.*

$$S11(f) = \frac{V_{rt}(f)}{V_i(f)} \tag{2.31}$$

The equation relating $S11(f)$ to the complex permittivity of the material, $\epsilon(f)$, takes different forms for different experimental arrangements. In the arrangement shown in Figure 2.5, a finite sample of length l is inserted in a coaxial line, which is terminated in its characteristic impedance. In this case, the equation takes the form shown in equation (2.13). Substituting equation (2.14) in equation (2.13) and

rewriting the expression for $S11(f)$ yields

$$S11(f) = \frac{1-\sqrt{\epsilon(f)}}{1+\sqrt{\epsilon(f)}} \frac{1-e^{\frac{-4\pi j f l}{C}\sqrt{\epsilon(f)}}}{1-[\frac{1-\sqrt{\epsilon(f)}}{1+\sqrt{\epsilon(f)}}]^2 e^{\frac{-2\pi j f l}{C}\sqrt{\epsilon(f)}}} \tag{2.32}$$

where all variables are as previously defined in Section 2.2.

The most commonly used arrangement, however, is one in which a dielectric sample is placed at the end of a coaxial line so that it acts as its termination as shown in Figure 2.3. In this case, the relationship between $S11(f)$ and $\epsilon(f)$ has the form shown in equation (2.16). Substituting equation (2.17) and (2.18) in equation (2.16) and rewriting the resulting expression gives

$$S11(f) = \frac{\frac{1-\frac{Z_1}{Z_2}\sqrt{\epsilon(f)}}{1+\frac{Z_1}{Z_2}\sqrt{\epsilon(f)}} + e^{\frac{-4\pi j f l}{C}\sqrt{\epsilon(f)}}}{1+\frac{1-\frac{Z_1}{Z_2}\sqrt{\epsilon(f)}}{1+\frac{Z_1}{Z_2}\sqrt{\epsilon(f)}} e^{\frac{-4\pi j f l}{C}\sqrt{\epsilon(f)}}} \tag{2.33}$$

where Z_1 is the characteristic impedance of the cable and Z_2 is the characteristic impedance of the sample holder when it is filled with air.

Chapter 3

Error analysis

In this chapter the sources of errors in the measurement of complex relative permittivity by using the total reflection TDR will be identified. Then, suitable methods for minimizing these errors will also be devised. This will form the basis for a systematic procedure for carrying out the experiments.

3.1 Sources of errors

Most of the sources of error in TDR experiments have been previously identified and discussed by several researchers (Nicolson, 1968; Loeb et al., 1971; Iskander, 1972). These errors may be classified under three categories, namely, incident signal error, sampling errors, and time to frequency domain conversion errors. They may also be classified into two main categories as being of either time domain (T) or frequency domain (F) origin (Table 3.1). The time domain errors include those errors which are introduced during the measurement of the time domain wave forms. The frequency domain errors on the other hand include those errors which are introduced after the wave form data have been discrete Fourier transformed (DFT).

Sources of errors	*Origin*
Incident signal error	T
Sampling error	
Random electrical noise	T
Quantizing error	T
Time jitter error	T
Time to frequency domain conversion errors	
Aliasing in DFT	F
Truncation in DFT	F
Unwanted reflections	F
Time referencing error	F

Table 3.1: Summary of sources of error in total reflection TDR.

3.2 Incident signal error

This is an error caused by a variation in the output of the pulse generator with time. If the output of the pulse generator fluctuates with time and the interval of time between the readings of the incident and the reflected wave forms is long, error will be encountered in the calculated value of $S11(f)$. This type of error was first observed by Iskander (1972) with the solid state pulse generator that he used in his experiments. Iskander suggested that a short time interval be used between the observation of the incident and the reflected wave forms in order to minimize the incident signal error.

3.3 Sampling errors

3.3.1 Random electrical noise

There are three sources of random electrical noise in electrical devices: Shot, Johnson, and Flicker noises (MacDonald, 1962). Shot noise is generated in valves and semiconductor devices with p–n junctions and is due to the random fluctuation in the rate at which the charge carriers cross the potential barriers. Johnson noise on the other hand arises because of the Brownian motion of electrons in electric circuits in particular in resistors and for this reason it is also referred to as *thermal noise*. The third type of noise, Flicker noise, arises mainly in transistors. The power spectra of Flicker noise varies inversely approximately with frequency and its effect is usually negligible at frequencies greater than $1\,MHz$ (Robinson, 1966, cited by Dawkins et al., 1979).

Because it is random, the random electrical noise sometimes add to the real wave form and sometimes subtract energy from the real wave form. Its magnitude can only be reduced by increasing the signal–to–noise ratio for the measurement (Dawkins et al., 1979). This can be done by averaging each wave form. Since the average amount of noise added to the wave form will be nearly the same as the average amount of noise subtracted from the wave form, by averaging several noisy wave forms together the noise can be averaged out (Clarkson et al. 1975).

3.3.2 Quantizing and time jitter errors

The step signal produced by the pulse generator and the reflected signal from the sample are continuous-time signals. The conversion of these signals to digital ones is done by using an analog–to–digital converter (ADC) and the conversion process is referred to as *Quantization*. Because the resolution of the ADC used to digitize

these signal is finite, the conversion process leads to round-off errors in the digitized values. This error is called *Quantizing error*, Q_e, and is irreducible error due to the quantizing process and depends on the quantization levels or resolution of the quantizer.

In addition to quantizing error, it is also possible for time jitter to occur. If the scanning voltage is held fixed corresponding to a fixed point on the wave form being measured, the measured voltage will fluctuate because of imprecise triggering as the gate moves around over a small time segment about its mean poistion (Iskander, 1972). This results in uncertainty in the time location of the sample $f(nT)$. The sampling window is centered over a time interval $nT + Q_j$ instead of nT, *i.e.* the window is offset by the amount of Q_j.

3.4 Time to frequency domain conversion errors

3.4.1 Aliasing

Aliasing is an error in the spectrum of a signal caused by representing the continuous time domain signal by a uniformly train of samples. In Section 2.6.2, it was shown that if $F(\omega)$ denotes the Fourier transform of $f(t)$, the discrete Fourier transform of $f(t)$ for a sampling interval of T is given by,

$$F_T(\omega) = \sum_{n=-\infty}^{+\infty} F(\omega + \frac{2\pi n}{T}) \tag{3.1}$$

The frequency spectrum is thus periodic, recurring at an angular frequency interval of $\frac{2\pi}{T}$, the sampling frequency.

Figure 3.1*a* shows a frequency spectrum of a continuous signal with frequency components limited to f_c. When this signal is sampled at a rate f_s, the sampling process results in the signal frequency spectrum shown in Figure 3.1*b*. Here, because

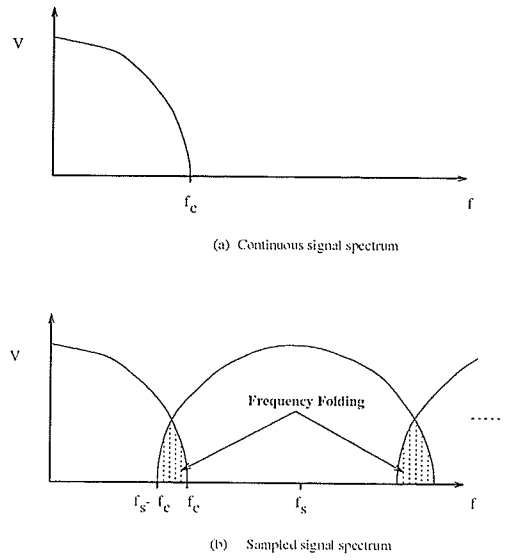

Figure 3.1: Spectra showing frequency folding.

the sampling frequency, f_s, is not sufficient, some of the high frequency components of the signal are folded back into the signal spectrum. That is, $F_T(\omega)$ is equal to $F(\omega)$ plus an unwanted contribution from the adjacent lobes. The main problem here is that in the process of recovering the original signal from its frequency spectrum, the folding frequency components cause distortion and can not be separated or distinguished from the original signal. This undesirable phenomenon of overlapping of adjacent lobes is referred to as *aliasing* (Chen, 1979) and it is under the control of the sampling frequency, f_s. It can be seen from Figure 3.1*b* that by changing the sampling frequency such that $f_s - f_c \geq f_c$, we can obtain the result $f_s \geq 2f_c$ which demonstrates the sampling theorem.

For a bandwidth limited signal, aliasing can be eliminated by either of the following two ways: (a) by using a high enough sampling frequency $f_s \geq 2f_c$, or (b) by filtering the original signal to eliminate any frequency component above one half the sampling frequency.

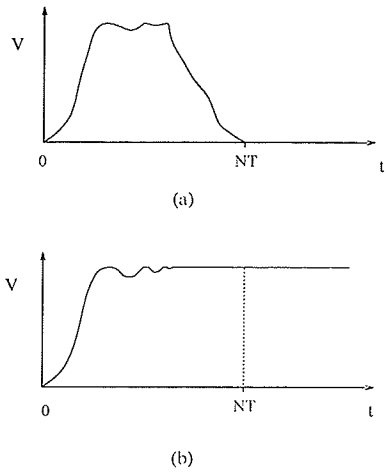

Figure 3.2: (*a*) A transient signal decaying to zero. (*b*) A typical *TDR* signal.

3.4.2 Truncation

In addition to aliasing, the other serious problem with the use of the discrete Fourier transform is *truncation*. The discrete Fourier transform function with sampled data assumes that the function to be transformed is finite (Clarkson et al., 1977). In many applications, the functions under consideration decay to zero. An example is shown in Figure 3.2*a*. No significant truncation error, in general, arises in computing the Fourier transform of this type of function.

In *TDR*, however, many of the signals considered do not decay to zero but to a finite value instead (Figure 3.2b). In this case, a large truncation error results in the frequency spectrum when a finite number of samples are used. This truncation error due to restricting the summation to a finite number of samples N (equation (2.28)) is given by,

$$F_{T\,error}(\omega) = \sum_{n=N}^{+\infty} f(nT)e^{-jn\omega T} \tag{3.2}$$

In *TDR*, generally three techniques are used to overcome this problem:

(*i*) In the first case, it is assumed that the signal decays as $e^{-\alpha t}$ for long times.

If the signal under consideration attains a constant value at time $t = NT$, equation (2.28) may be rewritten as

$$F_T(\omega) = T \sum_{n=0}^{N-1} f(nT) e^{-j\omega nT} + Tf(NT) \sum_{n=N}^{\infty} e^{-\alpha(n-NT)} e^{-j\omega nT} \tag{3.3}$$

As $\alpha \to 0$, the second summation converges to (Iskander, 1972)

$$\frac{e^{-jNT\omega}}{1 - e^{-jT\omega}} \tag{3.4}$$

Thus, when the spectrum of the signal is being computed, the following component may be added to correct for truncation error

$$Tf(NT) \frac{e^{-jNT\omega}}{1 - e^{-jT\omega}} \tag{3.5}$$

(*ii*) In the second case, a ramp function, as shown in Figure 3.3, is subtracted from the signal before computing the discrete Fourier transform of the signal (Nicolson, 1973). If the signal decays to a constant value at time $t = NT$, and if the values of the discrete Fourier transform of the signal are needed only at $\omega = k\omega_o$, where $\omega_o = \frac{2\pi}{NT}$ and $k = 1, 2, 3, ..., etc.$, a ramp function subtracted from the signal yields a frequency spectrum which is free from truncation error.

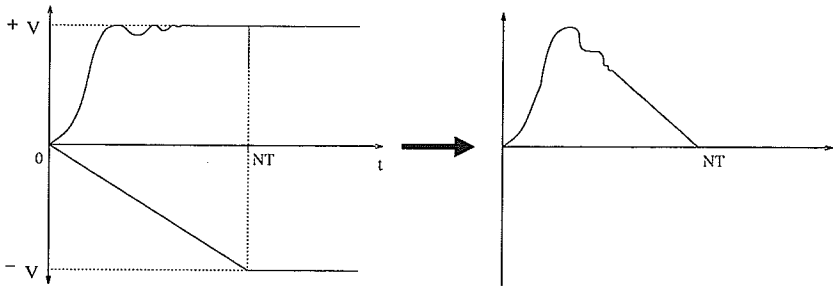

Figure 3.3: A signal with a ramp function.

(*ii*) In the third case, the Samulon (1951) (cited by Loeb et al., 1971) procedure is used where the signal is first subtracted from its value shifted by one–time interval

before Fourier transformation.

$$F_T(\omega) = \frac{T}{2j\sin(\omega T)} \sum_{n=0}^{N-1} \{f(nT) - f((n-1)T)\} e^{-j\omega nT} \tag{3.6}$$

This procedure, similar to the previous ones, assumes that the signals have attained a constant value at $t = NT$, so that all values beyond that point have backward differences, $\{f(nT) - f((n-1)T)\}$, of zero. So the calculated spectrum is free from truncation error.

3.4.3 Unwanted reflections errors

In an ideal coaxial line system only the step signal generated from the pulse generator and its reflection from the sample would be measured by the sampling head (Figure 2.5). In practice, however, the sampling head, the pulse generator, and each coaxial line connector introduce a small impedance discontinuity into the system which gives rise to unwanted multiple reflections. Occurrences of these reflections cause a serious error in the measurement of reflection or transmission coefficients and it is important that corrections be made to minimize their effects.

In TDR, the correction procedure developed by Giese and Tiemann (1975) is, in general, used to take into account the influence of these unwanted reflections quantitatively. Giese and Tiemann suggested that it be assumed that the pulse generator, the sampling head, and possibly the line section between these components make the major contribution to unwanted reflections. This combination may be characterized as a complex source reflection coefficient $S11_s(f)$ referred at the plane of the sampling head. The coaxial line system between this plane and the input plane of the sample is considered to be an ideal line of length l. With these assumptions, they showed that the relationship of the corrected reflection coefficient, $S11_c(f)$, and

the measured reflection coefficient, $S11_m(f)$ is

$$S11_c(f) = \frac{\{1 - S11_s(f)\}S11_m(f)e^{-2\gamma l}}{1 - S11_m(f)S11_s(f)e^{-2\gamma l}} \tag{3.7}$$

where γ is as previously defined in section 2.2. Writing $S11'_s(f) = S11_s(f)e^{-2\gamma l}$

$$S11_c(f) = S11_m(f)\frac{e^{-2\gamma l} - S11'_s(f)}{1 - S11_m(f)S11'_s(f)} \tag{3.8}$$

For a given experimental arrangement, $S11'_s(f)$ can be evaluated at each frequency of interest by a calibration measurement from a material having a known $S11_c(f)$ function.

Recently an alternative correction procedure was introduced by Heimovaara (1993) which uses a one-port error model used to improve network anlayzers measurements (Wiltron, 1989). In a one-port error model, errors in $S11(f)$ measurements are considered to be caused by source mismatch errors ($E_s(f)$) in the pulse generator, directivity errors ($E_d(f)$) in the sampling device and frequency tracking errors ($E_f(f)$) in the system. The measured reflection coefficient $S11_m(f)$ is related to the corrected reflection coefficient $S11_c(f)$ by the relationship

$$S11_m(f) = E_d(f) + \frac{S11_c(f)E_f(f)}{1 - E_s(f)S11_c(f)} \tag{3.9}$$

Solving the above relationship for $S11_c(f)$ yields

$$S11_c(f) = \frac{S11_m(f) - E_d(f)}{E_s(f)\{S11_m(f) - E_d(f)\} + E_f(f)} \tag{3.10}$$

The three error functions ($E_s(f)$, $E_d(f)$, and $E_f(f)$) can be evaluated from calibration measurements on three samples with a known $S11_c(f)$ function.

3.4.4 Time referencing errors

As already mentioned in Section (2.7.3), the computation of $S11(f)$ involves forming the Fourier transform of the incident and reflected wave forms. In complex

relative permittivity measurements at very high frequencies, a crucial point in the Fourier transformation is the establishment of a common time origin for the wave forms. The ratio of the Fourier transforms, $\frac{V_{rl}(f)}{V_i(f)}$, which is equal to $S11(f)$, has to be multiplied by $e^{j\omega\Delta t}$ when $V_i(t)$ and $V_{rl}(t-\Delta t)$ are shifted by Δt. A timing error of Δt introduces a phase error $\Delta\Phi = 2\pi f\Delta t$ in the measured $S11(f)$ also leading to an error in the calculated complex relative permittivity. It is, therefore, necessary that each wave form be time referenced before Fourier transformation.

Time referencing may be done in a number of ways. The most direct way of time referencing is the *Loeb's extrapolation technique* (Loeb et al., 1971). In this technique, the 'breakpoint' position on each wave form (X) (Figure 3.4) is used to define a relative time origin for the wave form. The breakpoint is the intersection of the line fitted to the horizontal portion prior to the falling (or rising) edge and to the steepest part of the wave form.

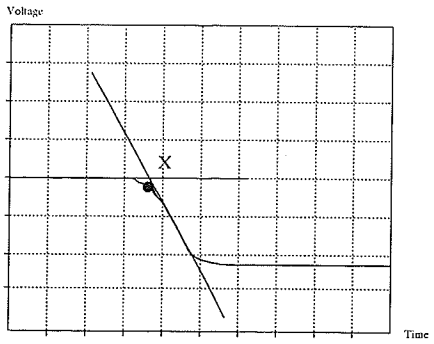

Figure 3.4: Loeb's extrapolation technique of time referencing.

The other technique commonly used in TDR is the one shown in Figure 3.5 and involves generation of time marker spikes on the experimental wave forms (Cole et al., 1980). The tip of the spike in each wave form is used as a relative time origin for the wave form.

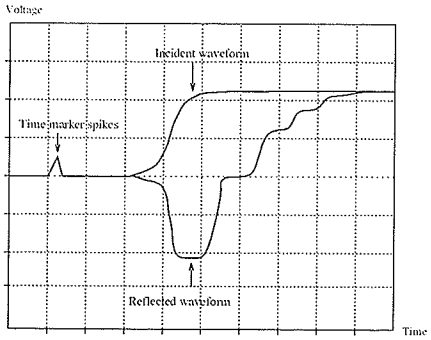

Figure 3.5: Incident and reflected wave forms with time marker spikes.

Chapter 4

Material and methods

4.1 Introduction

A method, using the forward scattering function ($S11(f)$) of a coaxial probe, to measure the complex relative permittivity of soil was developed and tested. The experiment was conducted in two parts. In the first part, the variations of the real and the imaginary parts of the complex relative permittivity of soil with frequency and with soil volumetric water content were investigated. In the second part, the effects of soil temperature and soil electrical conductivity on the complex relative permittivity of soil were studied. In the present chapter, the experimental apparatus used and the soil samples tested in the study will be described. This will be followed by a discussion on the probe design and calibration, the preparation of the soil samples, and the computation of complex relative permittivity from the wave forms.

4.2 Experimental apparatus and soils samples

The experimental apparatus used in this study consisted of a Tektronix 7854 mainframe oscilloscope (Figure 4.1) with a $7S12$ reflectometer unit (S-52 tunnel diode pulse generator and S-6 sampling head). The tunnel diode generates step pulses with approximately 35 picoseconds rise time and 0.24 V amplitude.

A computer program, $PROG\text{–}1.CPP$ (*Appendix A*), was written to interface the oscilloscope with an IBM–compatible computer and the interface was achieved by using the $GPIB$[1] protocol ($IEEE$ standard 488-1978). The oscilloscope was operated in remote mode via the computer and except for power–on, all the other operations (such as selection of the vertical mode and the horizontal time base unit, the points per wave form setting etc.) were programmed into the computer control system.

A coaxial probe, as explained in the next section, was constructed and connected to one end of a 40 cm long 50 Ω $RG58A$ coaxial cable (Figure 4.2). The other end of the cable was fitted with a male SMA coaxial connector so that it could be directly connected to the female SMA connector on the reflectometer unit. In order to minimize any discontinuity introduced by the connections, all connections were cleaned with a circuit cleaner.

Five types of soils were obtained from the soil physics laboratory at the University of Manitoba. The samples were chosen to give a wide range of textures (sand to clay). The particle size distributions and particle densities of the soils are shown in Table 4.1. (The names of the soils refer to their approximate location of collection). The soils were dried at room temperature and sieved. Experiments were conducted on soils passed though a 2–mm sieve. The soils were also tested for the presence of magnetism by using a magnet and none of them were found to be susceptible.

[1] General Purpose Interface Bus

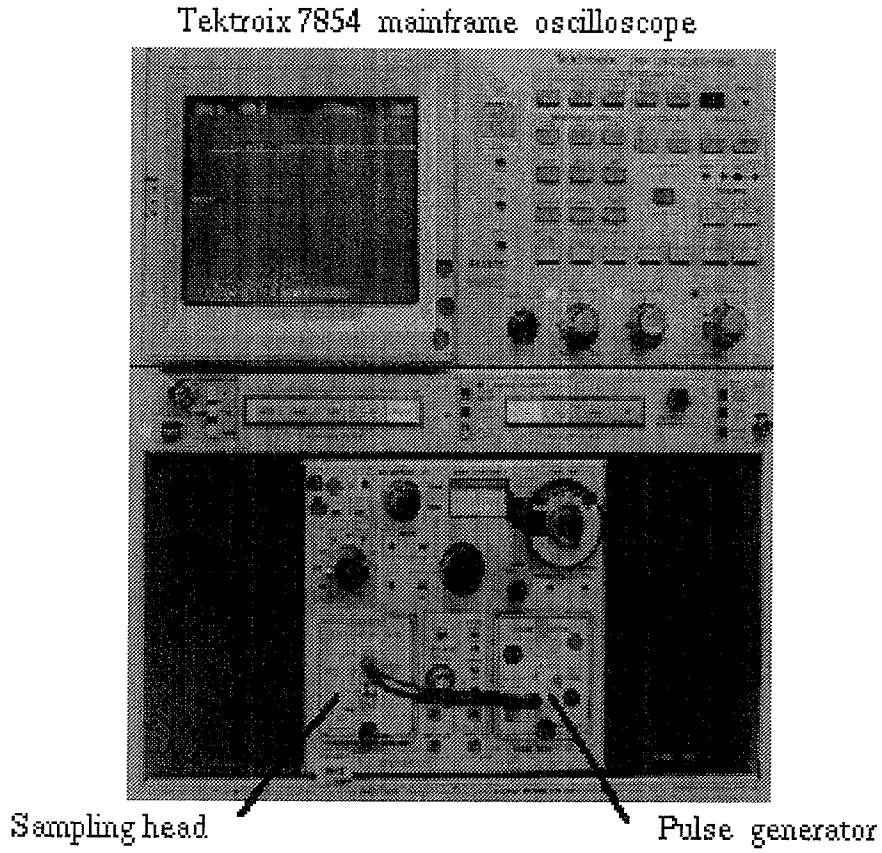

Figure 4.1: Tektronix 7854 mainframe oscilloscope with a $7S12$ reflectometer unit.

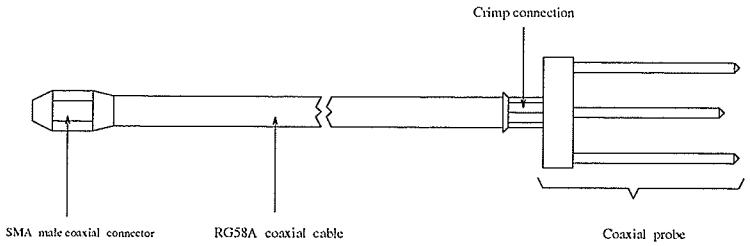

Figure 4.2: A coaxial probe connected to a 40 cm long $RG58A$ coaxial cable.

		Particle size			
Soil	*Particle Density*	*Sand*	*Silt*	*Clay*	*Textural Class*
Pelan	2.83	89.29	2.19	8.52	Sand
Carberry	2.72	70.00	17.50	12.50	Fine sand
Emerson	2.47	33.13	28.30	38.56	Clay loam
Fortier	2.63	10.67	31.58	57.75	Clay
Sperling	2.65	11.01	35.30	53.69	Clay

Table 4.1: Selected physical characteristics of the soils used in the study.

4.3 Probe design

The probe was constructed in a simlilar way as that used by Campbell (1989). It is basically a coaxial transmission line in which the outer coaxial tube is emulated by six stainless steel rods attached to a holder (A) (Figure 4.3). The inner coaxial conductor was replaced by a short stainless steel rod and was connected to a cylinderical holder (B). Both rod holders were made of brass and they were separated from each other by a ring made of $PTFE$ (commonly known as *Teflon*). The relative permittivity (ϵ) of the $PTFE$ used for the ring is 2.1 and its relative permeability (μ) is very close to unity. The characteristic impedance, Z, of the section in which the ring is inserted is given by (Kraus, 1984)

$$Z = \frac{60}{\sqrt{\epsilon\mu}} \ln(\frac{b}{a}) \tag{4.1}$$

where a is the inner diameter of the outer rod holder (or outer diameter of the teflon ring) and b is the outer diameter of the inner brass cylinder (or the inner diameter of the teflon ring). The ratio $\frac{b}{a}$ was chosen to be 3.35 in order to get a characteristic impedance equal to that of the connecting coaxial cable (*i.e.* 50 Ω). All rods were removable and their diameters were 1.6 mm. The coaxial cable's inner conductor

was directly soldered to the inner rod holder of the probe and the outer conductor of the coaxial cable was connected to the outer rod holder of the probe by making a crimp connection. Using equation (4.1) characteristic impedance of the probe was computed to be 144.89 Ω.

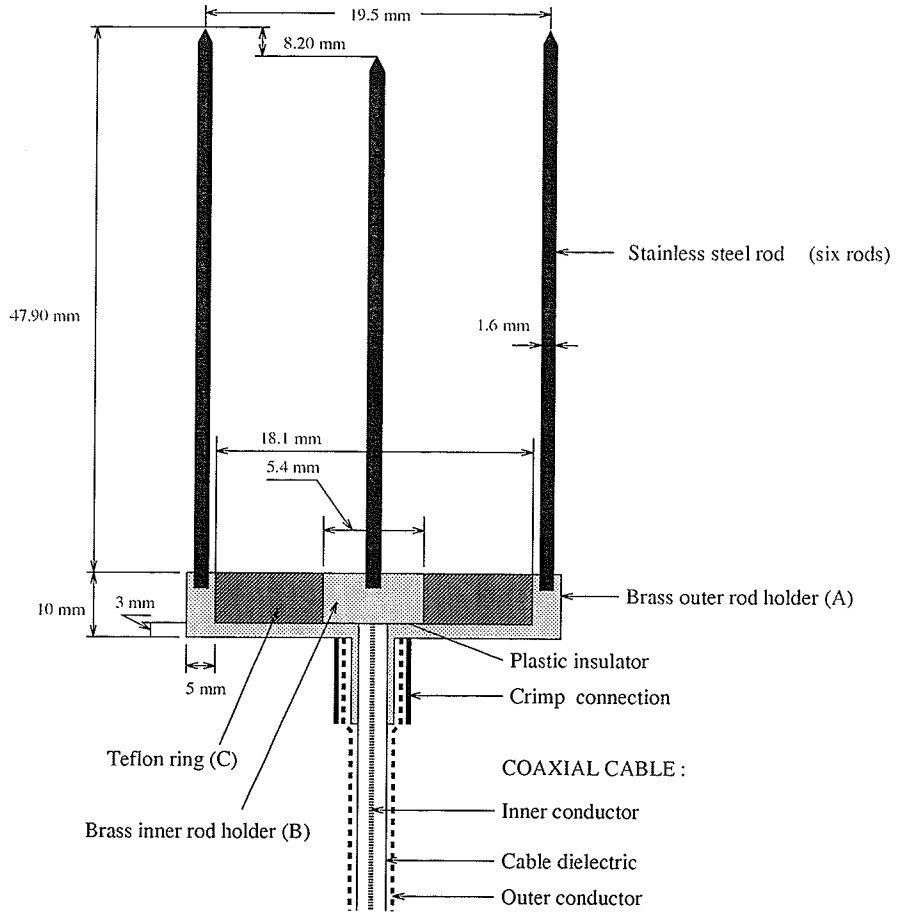

Figure 4.3: A sectional drawing of the coaxial probe used in the study.

The basic equation used for the measurement of complex relative permittivity of

soil was the equation derived in Section 2.7.3 and given by equation (2.33). Rewriting the equation

$$S11(f) = \frac{\frac{1-z\sqrt{\epsilon(f)}}{1+z\sqrt{\epsilon(f)}} + e^{\frac{-4\pi jfl}{C}\sqrt{\epsilon(f)}}}{1 + \frac{1-z\sqrt{\epsilon(f)}}{1+z\sqrt{\epsilon(f)}} e^{\frac{-4\pi jfl}{C}\sqrt{\epsilon(f)}}} \tag{4.2}$$

with

$$z = \frac{Z_0}{Z_p} \tag{4.3}$$

where z is the impedance mismatch factor, Z_0 is the characteristic impedance of the coaxial cable and Z_p is the characteristic impedance of the probe when it is empty (*i.e.* when it is in air).

4.4 Probe calibration

If the constructed probe was a perfect coaxial probe, equations (4.2) and (4.3) would be applicable for the probe without any modification. Because the probe is not a perfect coaxial probe, it must be calibrated by using measurements taken from a reference material with a known $\epsilon(f)$ function. The calibration was done in a similar way to that reported by Grant et al. (1989) by using ethanol as a reference material. The Cole-Cole parameters of ethanol at 25^oC: $\epsilon_\infty = 4.8$, $\epsilon_s = 24.4$, $f_{rx} = 1.14$ GHz, $\alpha = 0.0$, and $\sigma_{dc} = 0.0$ (all symbols are as previoulsy defined in Section 2.4) given by Grant et al. (1989) were used in equation (2.24) to derive the complex relative permittivity of ethanol as a function of frequency.

The scattering function, $S11(f)$, as shown in equations (4.2) and (4.3) depends on two physical properties of the probe, namely, length of the probe l and the impedance mismatch factor z. The calibration procedure is, therefore, used to find optimum values of l and z (Heimovaara, 1994) in order to minimize the difference between the $S11(f)$ measured by the probe and the correct one. Measurements from

the reference material and the evaluation of optimized values of l and z were done in the following manner:

1. Approximately 250–cm^3 of ethanol was placed in a glass beaker and the beaker was kept in a temperature controlled chamber at 25^oC for two hours. The beaker was then transported to the experiment site in a foam insulated container and during the measurement the temperature of the ethanol was measured to be $(25 \pm 0.2)^oC$.

2. After the incident wave form was acquired, as explained in the next section, the probe was lowered vertically into the beaker containing the ethanol (Figure 4.4) and the reflected wave form was collected. Care was taken to ensure that no air bubbles were trapped beneath the probe.

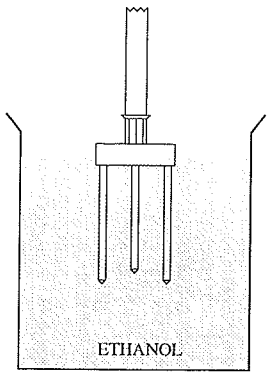

Figure 4.4: A coaxial probe immersed in ethanol.

3. The actual calibration was done by approximating or predicting the reflected wave form and minimizing the sum of the squares of the residual differences between the approximated and the observed reflected wave forms as follows:

   (a) Using the Cole–Cole parameters of ethanol given above and equation (2.24), the complex relative permittivity of ethanol, at each frequency of interest, was computed. The computed complex relative permittivity values were

inserted into equations (4.2) and (4.3) and $S11_{app}(f)$ was approximated (*app* stands for approximated).

(b) The observed incident wave form was discrete Fourier transformed. From the $S11_{app}(f)$ and the discrete Fourier transform of the incident wave form, $V_i(f)$, the discrete Fourier transform of the reflected wave form, $V_{rl\ app}(f)$, was approximated.

$$V_{rl\ app}(f) = S11_{app}(f)V_i(f) \tag{4.4}$$

The inverse discrete Fourier transform of $V_{rl\ app}(f)$ was computed to get the approximated reflected wave form, $V_{rl\ app}(t)$.

(c) The sum of the squares of the residual differences between the approximated and the observed wave forms, SS, was computed

$$SS = \sum_{n=0}^{N-1} \{V_{rl\ app}(t) - V_{rl}(t)\}^2 \tag{4.5}$$

where $V_{rl}(t)$ is the observed reflected wave form and N is the number of data points.

4. In the first approximation of the reflected wave form, the value of l was equal to the measured length of the probe and z was equal to the impedance mismatch factor computed theoretically assuming the probe as a perfect coaxial transmission line. Thereafter by keeping the value of l constant and by changing the value of z and vise versa, step 3 was repeated until SS became minimal. The values of l and z, when SS was minimal, were taken to be the values of the optimized probe length and the impedance mismatch factor, respectively. A computer program, *PROG–2.CPP* (*Appendix A*), was written for this task and all the operations in step 3 were performed by this program.

4.5 Collection and handling of waveform data

As already mentioned previously the experiment was conducted in two parts. In the first part, tests were conducted on all of the five soils. Prior to the tests, each soil was kept in a temperature controlled chamber at 25^oC for five hours. The soil was moistened by a distilled water in 7 to 10 steps to cover the range from dry to saturation. At each water content the wet soil was placed in a 250–cm^3 glass beaker and compacted to a bulk density as uniformly as possible. The coaxial probe was connected to the reflectometer unit and the incident and the reflected wave forms were aquired. A known volume of the soil sample was taken and oven dried at 105^oC for 24 hours and the gravimetric water content and the bulk density were determined. From the bulk density and the gravimetric water content, the volumetric water content was obtained. For each soil the measurements at each water content was replicated three times. The temperature of the wet soil during the experiments was $(25 \pm 0.5)^oC$.

In the second part, the tests were conducted only on one soil type (Carberry soil). First, the soil was conditioned to 15^oC, 25^oC, and 35^oC temperatures for five hours and the readings were taken in the same way as described above. Then, by maintaining the temperature of the soil at 25^oC, and by bringing the electrical conductivity of the water used to wet the soil to 0.13 $dS \cdot m^{-1}$, 0.88 $dS \cdot m^{-1}$, 1.74 $dS \cdot m^{-1}$, 2.57 $dS \cdot m^{-1}$, and 3.54 $dS \cdot m^{-1}$, similar readings were taken.

The reflected wave form was obtained by inserting the probe vertically into the beaker filled with soil and recording the reflected wave form. The incident wave form, on the other hand, was acquired by removing the inner conductor of the probe and recording the wave form reflected from the open end of the probe in air (Nicolson, 1968; Kao, 1989). The rise time of the incident wave form was measured to be 200 picoseconds. For each wave form, 1024 digital points were recorded and transferred to the computer by using $PROG$–$1.CPP$. The time interval between the digitized

points was 9.765×10^{-13} s. Before digitizing each wave form, one hundred wave forms were averaged by the oscilloscope in order to increase the signal to noise ratio.

4.6 Calculation of complex relative permittivity

Prior to computation of the complex relative permittivity of the soils, the frequency window (bandwidth) of the measurement system was determined. The frequency window may be defined as the range of frequencies in which calculations of complex relative permittivities are considered accurate (Kao, 1989).

The factors considered in finding the upper frequency limit, f_H, of the window are the rise time, t_r, of the incident wave form (Cole et al., 1980) and the time interval between digitized points on the wave forms, Δt. Using the rise time, f_H may be computed by the relationship $f_H = \frac{1}{2t_r}$ (Cole et al., 1980; Tocci, 1980). Using Δt, f_H may be taken to be equal to the folding or Nyquist critical frequency, f_c, given by $f_c = \frac{1}{2\Delta t}$ (In engineering practice, however, it is taken to be equal to $\frac{1}{20\Delta t}$ or one–tenth of f_c, the Nyquist critical frequency, to ensure no aliasing of wave forms occurs).

The upper frequency limit from the rise time and from the time interval between the digitized points, given in the previous section, were computed to be 2.50 GHz and 512.00 MHz, respectively. The upper frequency limit for the measurement system used in this study was thus taken to be 512.00 MHz. The sampling frequency, f_s, was also determined to be 10.24 GHz and was more than adequate for the frequency bandwidth of the wave forms.

The factors considered in determining the lower frequency limit is the duration of the pulse generated by the pulse generator (Boned et al., 1982; Heimovaara, 1994). The pulse duration, being 800 nanoseconds for the S–52 tunnel diode pulse

generator (the lowest frequency limit for the system used in this study), was taken to be 2.00 MHz (Boned et al., 1982). Based on the above frequency window, complex relative permittivity of the soil was measured at fifteen frequency points starting from 10.00 MHz at uniform intervals of 10.00 MHz.

The major part of the calculation of the complex relative permittivity of the soils was the computation of the discrete Fourier transforms of incident and reflected waveforms. The discrete Fourier transformation of the waveforms was evaluated by using the Fast Fourier Transform (FFT) routine written by Press et al. (1988) based on the *Danielson-Lanczos FFT algorithm*. This routine was modified and included in $PROG$–3.CPP (*Appendix A*). It was thoroughly checked for its correctness. The truncation error in discrete Fourier transformation of the waveforms was limited by using the Nicolson method (1973) as described in Section 3.4.2. From the discrete Fourier transform of the reflected waveform and that of the incident waveform, $S11(f)$ was computed (equation (2.31)) and corrected for unwanted reflections by using the Giese and Tiemann (1975) correction procedure as explained by Section 3.4.3. Since the measurements of the complex relative permittivity of the soils were limited to low frequencies, no time referencing was done (Arcone et al., 1986; Kao, 1989). The first data point on each waveform was taken to be a relative time origin for the waveform (Heimovaara, 1994). In order to minimize the effect of the incident signal error (Section 3.2), the incident waveform and the reflected one were collected in a short time interval from each other as suggeted by Iskander (1972). Using $S11(f)$ values and equations (4.2) and (4.3), the complex relative permittivity values were computed by using a complex version of the *Newton-Raphson Method of Iteration* ($PROG$–4.CPP in *Appendix A*).

The operating sequence from acquisition of wave forms up to computation of the complex relative permittivity values is summerised in Figure (4.5).

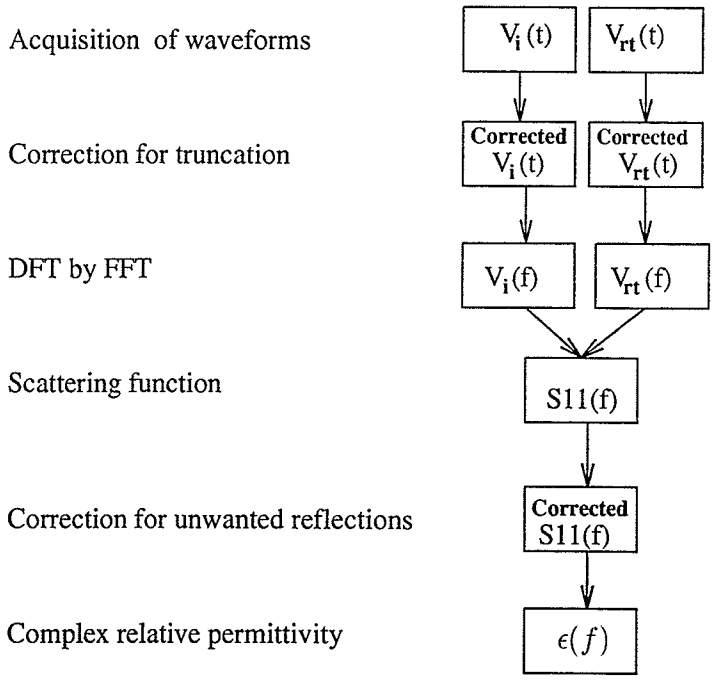

Figure 4.5: Summary of the operating sequence.

Chapter 5

Results and discussion

In this chapter results obtained in the probe calibration processe are presented first. This is followed by the results and discussions on the tests conducted on the five soils.

5.1 Probe calibration

The probe was constructed and calibrated as explained in the previous chapter. Optimized values of l and z were found to be 41.00 mm and 0.33, respectively. The reflected wave form measured from the reference material (ethanol) and the approximated wave form are shown in Figure 5.1. The approximated wave form follows the measured wave form very closely. However, there is some discrepancy between the two wave forms which may be due to the non–coaxiality of the probe.

To minimize the effect of truncation error, the incident wave form used in the probe calibration process was corrected for truncation errors by a Nicolson's method (Nicolson, 1973).

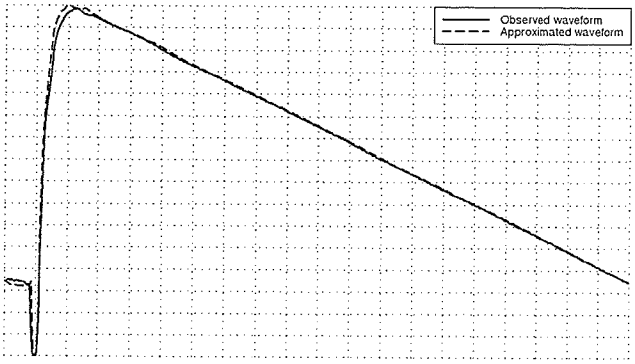

Figure 5.1: Comparison of the approximated and the measured wave forms from a coaxial probe immersed in ethanol.

5.2 Frequency dependency of complex relative permittivity of soil

Figure (5.2) through Figure (5.6) show the real and the imaginary parts of the complex relative permittivity of the five soils as a function of frequency and as a function of volumetric water content. The data plotted are mean values of three replications.

It is evident from the result that both the real and the imaginary parts of the complex relative permittivity are strongly dependent on frequency and water content. For all of the five soils, the real part of the complex relative permittivity decrease with an increase in frequency and is invariably lowest at the lowest volumetric water content and highest at highest volumetric water content. This dispersion in the real part of the complex relative permittivity of the soils may be attributed to the presence of water and relaxation of water molecules. As already explained in

Section 2.4 as the frequency of the electric field increases, oscillation of water molecules lags behind the electric field and the molecule contribute less to the total polarization and hence to the real part of the complex relative permittivity (equation (2.23)).

The frequency dependency of the real part of the complex relative permittivity decreases with a decrease in volumetric water content. For example, for Fortier clay soil at 48% volumetric water content the real part of the complex relative permittivity drops from 29.79 to 18.38 (38% drop in magnitude) in the frequency range of 10 to $150\,MHz$. But at 0% volumetric water content, it drops from 9.35 to 7.14 (24% drop in magnitude) in the same frequency range. This may be due to the large amount of relaxation at the high volumetric water content and the low amount of relaxation at low volumetric water content. As already explained, dielectric relaxation (*i.e.* a decrease in the real part of the complex relative permittivity with an increase in frequency) is a phenomenon associated with polar substances. As the amount of water, a polar substance, in a given porous material decreases the extent of dispersion of the material decreases. The above substantial decrease in the real part of the complex permittivity with a decrease in volumetric water content is to be expected. A similar behavior was observed by Wobschall (1977) by using the semidisperse complex relative permittivity model.

A sharp drop in the real part of the complex relative permittivity is also observed in fine textured soils (*i.e.*, Fortier Clay) as compared with a gradual drop in coarse texture soil (*i.e.*, Carberry fine sand and Pelan sand soils). This may be due to increased porosity and thus higher water holding capacity in fine texture soil as compared with that in coarse texture soil.

The imaginary part of the complex relative permittivity showed a pronounced dispersion and soil dependency. For all the five soils, it decreased with an increase in frequency. This may be due to the decrease in ionic conductivity in the soil with the increase in frequency. Recently, Campbell (1990) studied a number of possible

dispersion mechanisms of the imaginary part of the complex relative permittivity (such as ionic conductivity, surface conduction, bound water relaxation etc.) and conclude that ionic conductivity in soils is the main dispersion mechanism in the imaginary part of the complex relative permittivity. The contributions of the other mechanisms of dispersion to the imaginary part of the complex relative permittivity were found to be negligible as compared with that of the ionic conductivity. The imaginary part of the complex relative permittivity, from equation (1.1), is given by $\{\epsilon''(f) + \frac{\sigma_{dc}}{2\pi f \epsilon_o}\}$. Assuming ionic conductivity, σ_{dc}, as the only dispersion mechanism, the magnitude of the imaginary part of the complex relative permittivity is proportional to $\frac{\sigma_{dc}}{2\pi f \epsilon_o}$ and varies inversely with frequency. The decay of the imaginary part of the complex relative permittivity of the five soils, studied in this research, with an increase in frequency is, therefore, to be expected and is in good agreement with Campbell's observation.

The results also showed volumetric water content and soil type dependency of the imaginary part of the complex relative permittivity. These may be due to the dependency of electrical conductivity of soil on water content and soil type, respectively. Change in electrical conductivity of a silty clay soil with an increase in water content was studied by Smith–Rose (1933). He observed that the electrical conductivity of the soil first increases sharply with an increase in water content up to approximately 25% (mass basis) and then levels off. A similar 1behavior was also reported for other types of soils by Chernyak (1967). Parkhomenk (1967) (cited by Hoekstra and Delaney, 1974) studied the difference in electrical conductivity among various saturated soils and reported that the electrical conductivity of a saturated soil varies from $10^{-1}\, dS \cdot m^{-1}$ in typical clay soil to $10^{-3}\, dS \cdot m^{-1}$ in sandy soil. This sort of dependency of electrical conductivity of soil on water content and on soil type

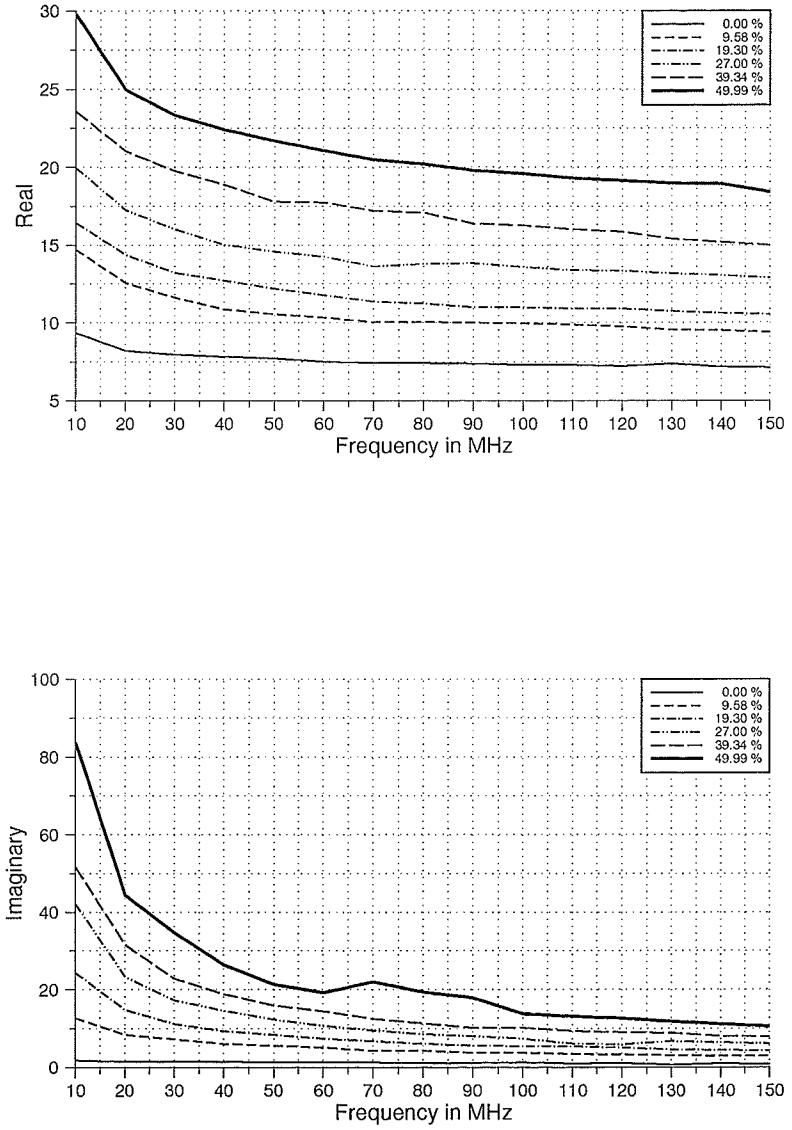

Figure 5.2: The real and the imaginary parts of complex relative permittivity of Fortier clay soil as a function of frequency at six water contents.

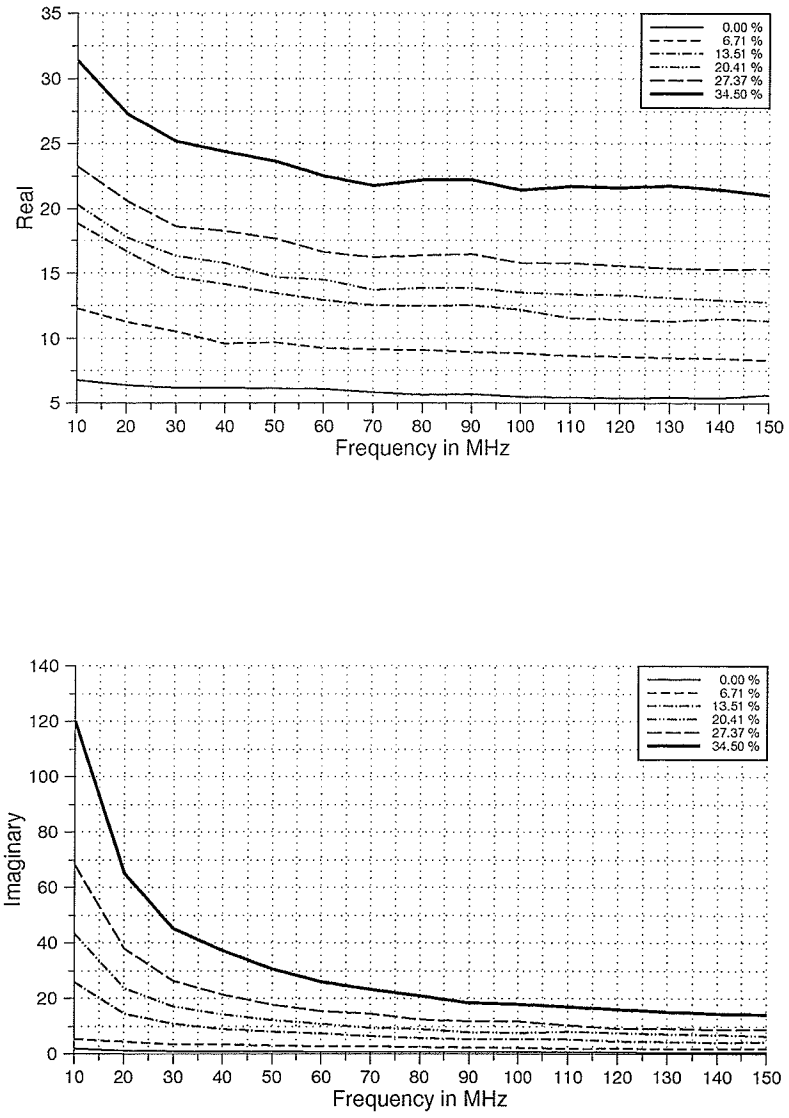

Figure 5.3: The real and the imaginary parts of complex relative permittivity of Emerson clay loam soil as a function of frequency at six water contents.

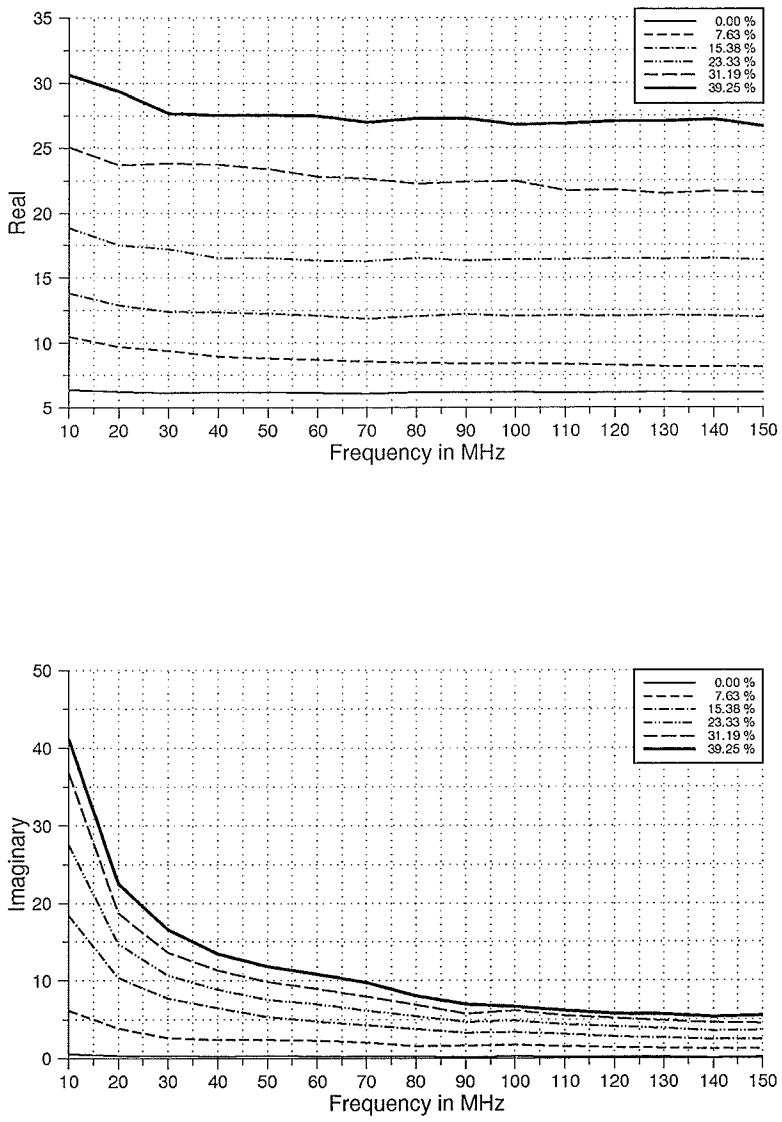

Figure 5.4: The real and the imaginary parts of complex relative permittivity of Carberry fine sand soil as a function of frequency at six water contents.

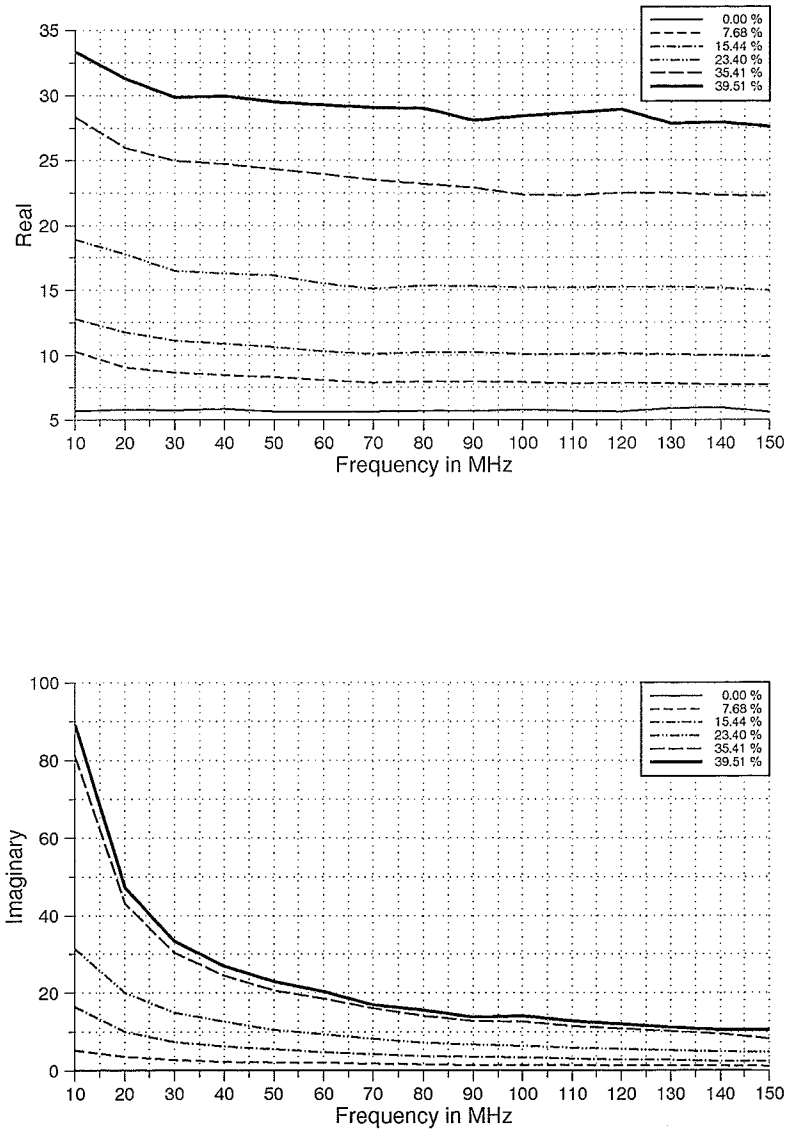

Figure 5.5: The real and the imaginary parts of complex relative permittivity of Pelan sand soil as a function of frequency at six water contents.

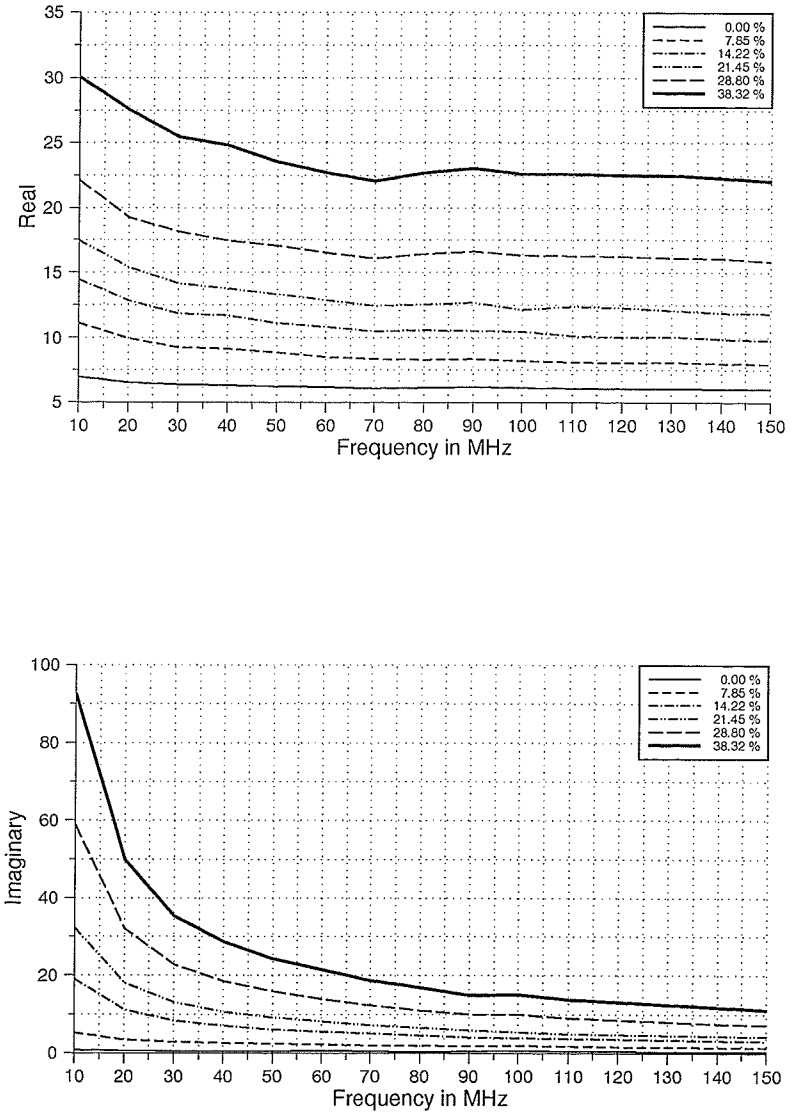

Figure 5.6: The real and the imaginary parts of complex relative permittivity of Sperling clay soil as a function of frequency at six water contents.

may,therefore, be a cause for the dependency of the imaginary part of the complex relative permittivity on the volumetric water content and soils type, respectively, as observed in this study.

5.3 Volumetric water content dependency of complex relative permittivity of soil

Figure (5.7) through Figure (5.11) show the real and the imaginary parts of the complex relative permittivity of the previous five soils plotted as a function of volumetric water content at 10, 50, 100, and 150 MHz frequencies.

When the relationship between water content and the complex relative permittivity of soil is being established, volumetric water content rather than gravimetric water content is generally used. It is the number of water molecules per unit volume that determines the contribution of water to the polarization (Hoekstra and Delaney, 1974) and hence to the real part of the complex relative permittivity (equation 2.23). For this reason, the water contents of all the five soils which were determined gravimetrically during the experiments, as explained in Section 4.4, were converted into volumetric water contents and plotted.

For all of the five soils both the real and the imaginary parts of the complex relative permittivity fell within a relatively narrow band indicating that the main parameter determining complex relative permittivity of soil at a given frequency is the volumetric water content. A similar strong dependency of the real and the imaginary parts of the complex relative permittivity on volumetric water content was observed by Campbell (1990), Dasberg and Hopmans (1992), Topp et al. (1980), and Hoekstra and Delaney (1974).

The real part of the complex relative permittivity of each soil was fitted to

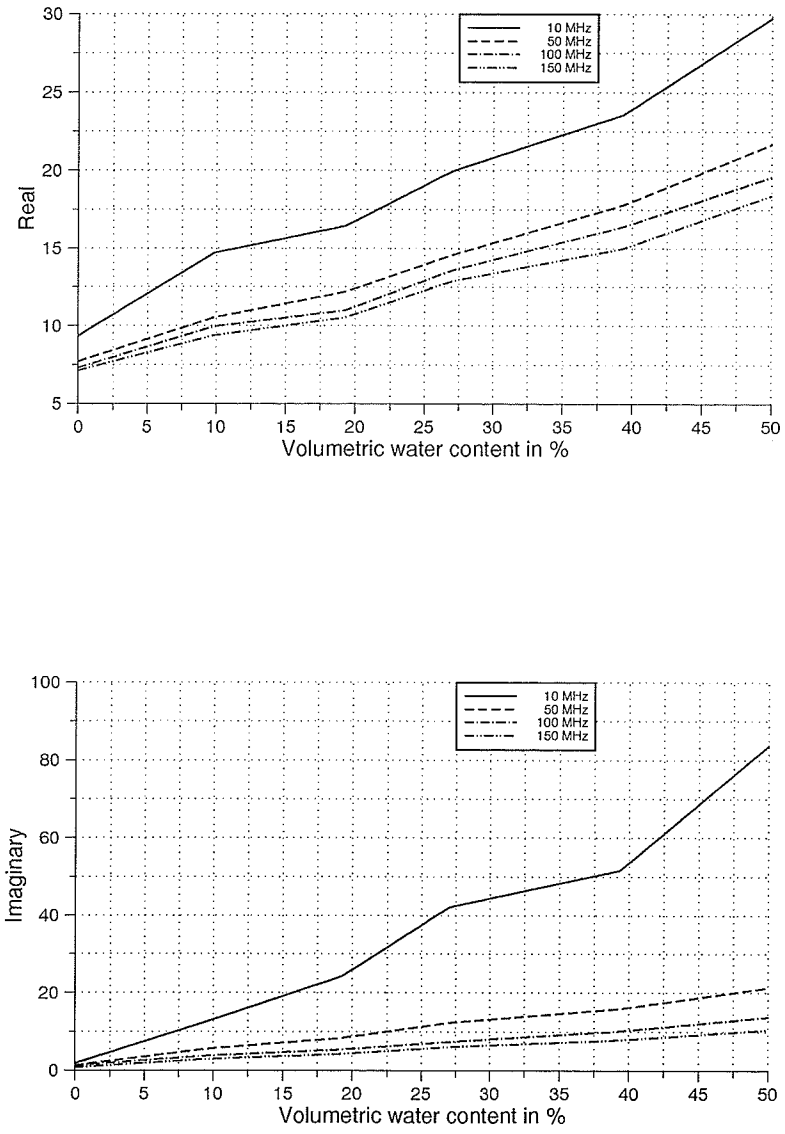

Figure 5.7: The real and the imaginary parts of complex relative permittivity of Fortier clay soil as a function of volumetric water content at four frequencies.

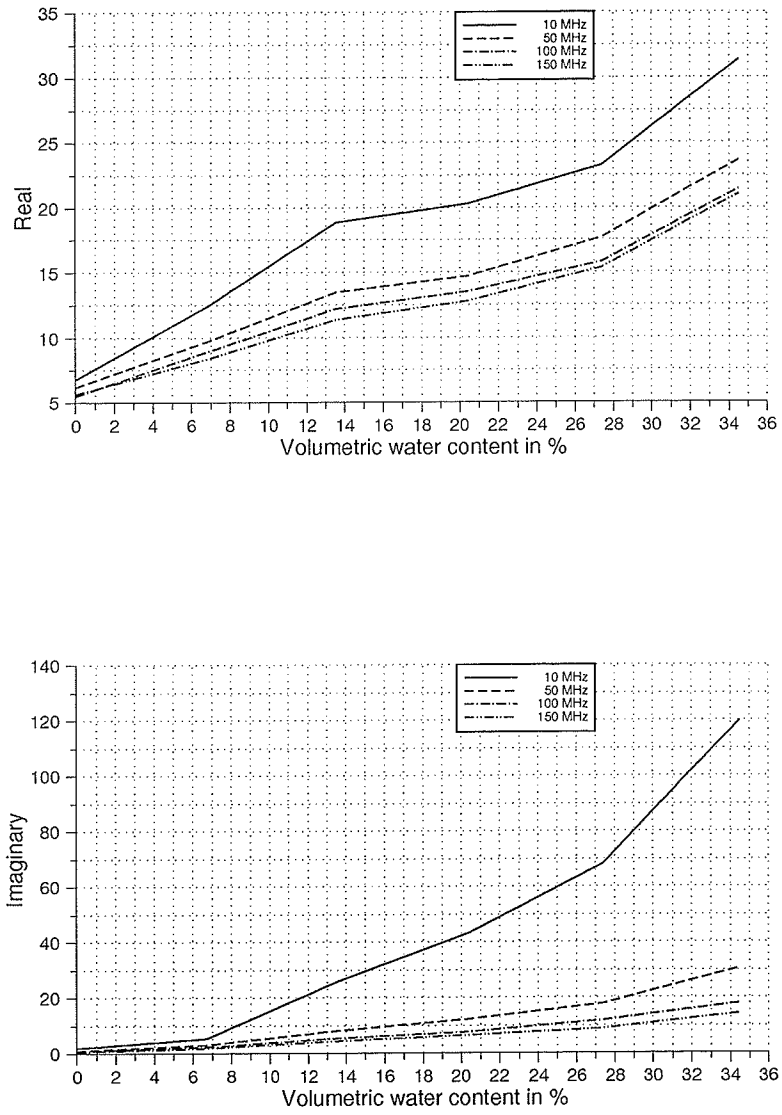

Figure 5.8: The real and the imaginary parts of complex relative permittivity of Emerson clay loam soil as a function of volumetric water content at four frequencies.

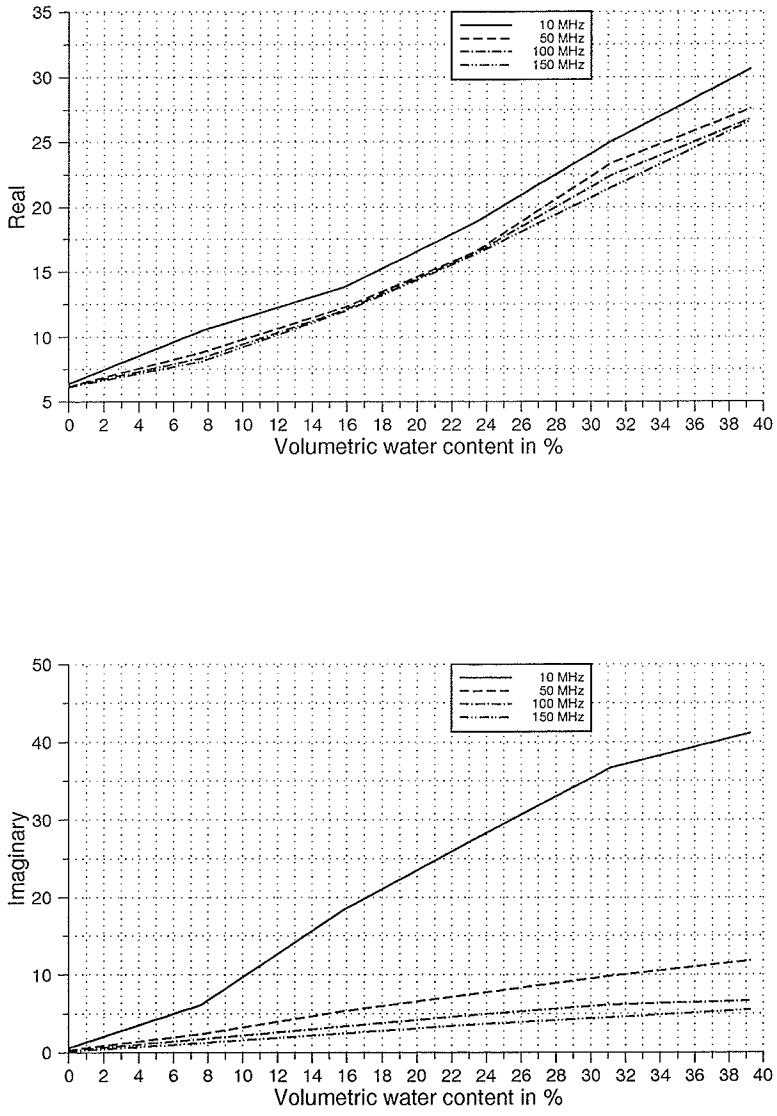

Figure 5.9: The real and the imaginary parts of complex relative permittivity of Carberry fine sand soil as a function of volumetric water content at four frequencies.

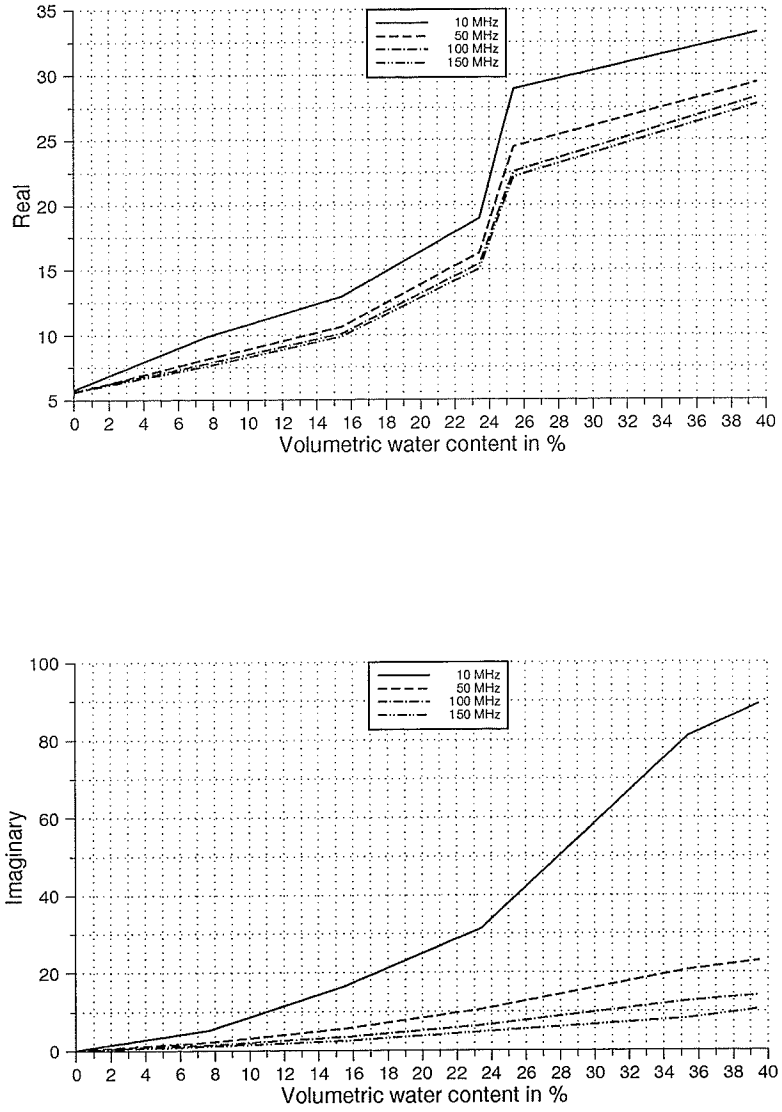

Figure 5.10: The real and the imaginary parts of complex relative permittivity of Pelan sand soil as a function of volumetric water content at four frequencies.

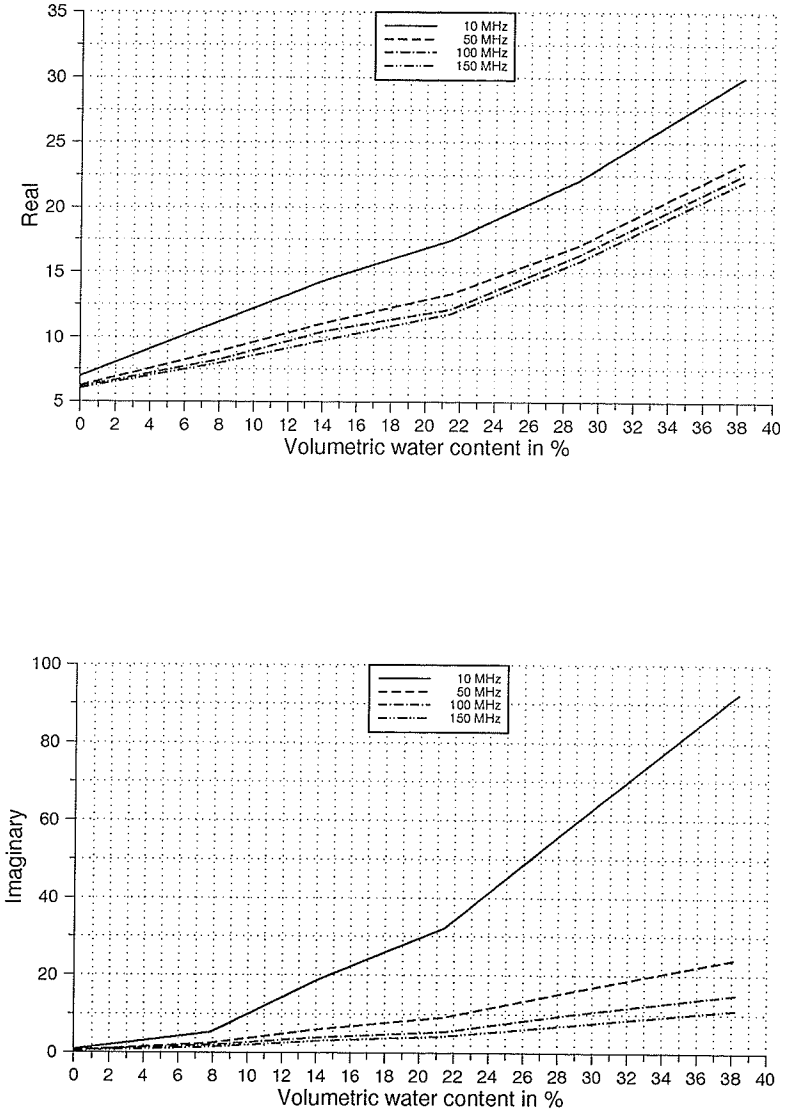

Figure 5.11: The real and the imaginary parts of complex relative permittivity of Sperling clay soil as a function of volumetric water content at four frequencies.

a best fit third–order polynomial equation and the coefficient of determination, r^2, obtained ranged from 0.98 for Sperling clay soil at 10 MHz to approximately unity for Carberry fine sand at 150 MHz.

The real part of the complex relative permittivity of all the five soils at 10, 50, 100, and 150 MHz frequencies was also combined and fitted to a best fit third–order polynomial equation. The equations for the lines were

$$\epsilon' = 6.83 + (7.41 \times 10^{-1})\Theta - (2.37 \times 10^{-2})\Theta^2 + (6.68 \times 10^{-4})\Theta^3, \quad 10\,MHz \quad (5.1)$$

$$\epsilon' = 6.28 + (3.59 \times 10^{-1})\Theta - (4.90 \times 10^{-2})\Theta^2 + (3.09 \times 10^{-4})\Theta^3, \quad 50\,MHZ \quad (5.2)$$

$$\epsilon' = 6.16 + (2.88 \times 10^{-1})\Theta - (1.97 \times 10^{-2})\Theta^2 + (2.53 \times 10^{-4})\Theta^3, \; 100\,MHz \quad (5.3)$$

$$\epsilon' = 6.08 + (2.31 \times 10^{-1})\Theta - (9.89 \times 10^{-2})\Theta^2 + (2.02 \times 10^{-4})\Theta^3, \; 150\,MHz \quad (5.4)$$

with r^2 values of 0.90, 0.88, 0.86, and 0.86, respectively. In the equations Θ is the volumetric water content (in %) and ϵ' is the real part of the complex relative permittivity.

The same data were also fitted to a third–order polynomial equation assuming Θ as dependent and ϵ' as independent variable. Because, in practice, it is these type of equations which are used to determine volumetric water content from ϵ' determined by TDR. The equations of the lines were

$$\Theta = -10.456 + 1.342\epsilon' - 5.00 \times 10^{-2}(\epsilon')^2 - 1.60 \times 10^{-3}(\epsilon')^3, \quad 10\,MHz \quad (5.5)$$

$$\Theta = -17.456 - 2.753\epsilon' + 3.00 \times 10^{-2}(\epsilon')^2 - 2.841 \times 10^{-3}(\epsilon')^3, \quad 50\,MHz \quad (5.6)$$

$$\Theta = -27.100 + 5.265\epsilon' - 1.292 \times 10^{-2}(\epsilon')^2 - 4.54 \times 10^{-4}(\epsilon')^3, \; 100\,MHz \quad (5.7)$$

$$\Theta = -32.219 + 6.643\epsilon' - 2.216 \times 10^{-2}(\epsilon')^2 - 2.268 \times 10^{-3}(\epsilon')^3, \; 150\,MHz \quad (5.8)$$

with the r^2 values of 0.92, 0.94, 0.93, and 0.94, respectively. The values of the coefficients among the four equation showed significant differences. This may be

due to a large difference among the frequencies to which the equations correspond. Similar differences in the values of coefficients were observed by Campbell (1990) and Kao (1989).

5.4 Effects of soil temperature and soil electrical conductivity

To investigate the effect of soil temperature on the complex relative permittivity of soil, tests were conducted on a moist Carberry fine sand soil conditioned to 15^0C, 25^0C, and 35^0C (Figure 5.12). The result showed strong dependency of the imaginary part of the complex relative permittivity on temperature particularly at lower frequency. This may be due to increased electrical conductivity of the soil with increased temperature. The highest temperature dependency was observed at $10\,MHz$ and the lowest at $150\,MHz$. As the frequency increased, the temperature dependency of the imaginary part of the complex relative permittivity decreased.

The real part of the complex relative permittivity did not show a similar trend as the imaginary part. The data fall in a narrow band which shows little dependency of the real part of the complex relative permittivity on temperature. A similar effect of soil temperature on the real and the imaginary parts of the complex relative permittivity was observed by Campbell (1990).

The effect of soil electrical conductivity was also studied by conducting tests on Carberry fine sand soil moisten by water with electrical conductivity of 0.13 $dS \cdot m^{-1}$, 0.88 $dS \cdot m^{-1}$, 1.74 $dS \cdot m^{-1}$, 2.57 $dS \cdot m^{-1}$, and 3.54 $dS \cdot m^{-1}$. A pronounced effect of electrical conductivity was observed on the imaginary part of the complex relative permittivity. At $10\,MHz$ frequency, the imaginary part increases from 18.76 at 0.13 $dS \cdot m^{-1}$ to 36.81 at 3.54 $dS \cdot m^{-1}$. This is approximately 100% increase

in magnitude. At higher frequencies, the dependency of the imaginary part of the complex relative permittivity on electrical conductivity is lower as compared with that at the lower frequencies. It is evident from the result that the effect of conductivity on the real part is not the same as that on the imaginary part. A similar effect of conductivity on the complex relative permittvity was reported by Wobschall (1977) in the frequency range of 0 to 1 GHz.

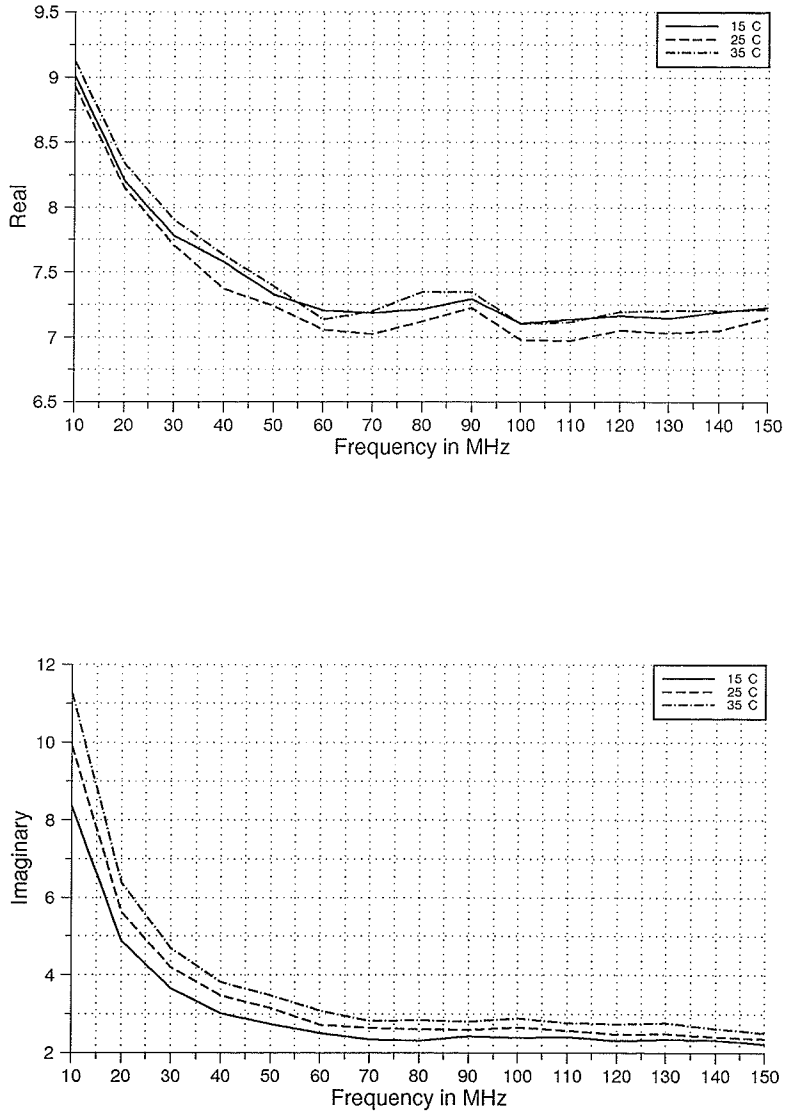

Figure 5.12: The real and imaginary parts of complex relative permittivity of Carberry fine sand soil as a function of frequency and temperature at 13.86% volumetric water content.

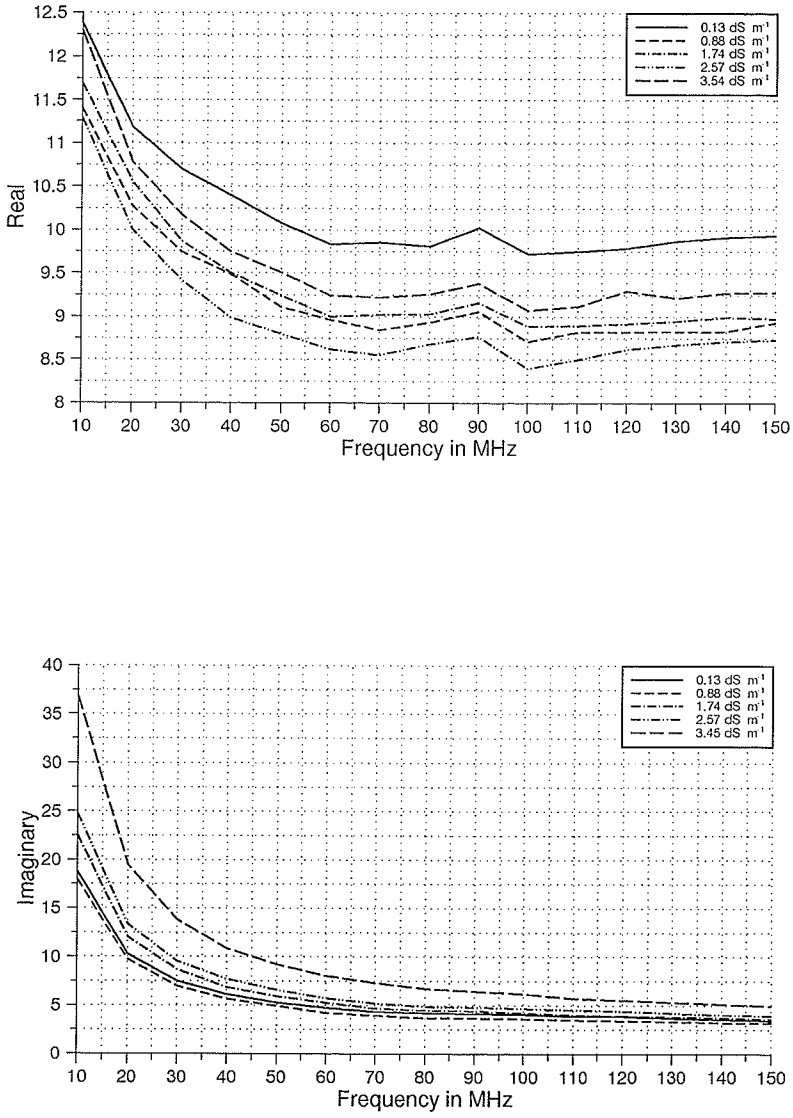

Figure 5.13: The real and imaginary parts of complex relative permittivity of Carberry fine sand soil as a function frequency and soil electrical conductivity at 13.85% volumetric water content.

Chapter 6

Summary and conclusions

A method for the measurement of complex relative permittivity of soils which utilizes the total reflection time domain reflectometry was described. The basic theoretical and practical basis underlying the measurement of complex relative permittivity by time domain reflectometry was presented. This includes multiple reflection in a coaxial cable, the microscopic concept of polarization, frequency dependency of permittivity, and Fourier analysis. The basic $S11(f)$ equation used for the computation of complex relative permittivity was derived. Many of the sources of errors in total reflection TDR were discussed. Various techniques contained in the literature either for minimization or elimination of these errors were explained. Computer programs for waveform data acquisition and analysis were developed.

Complex relative permittivity of five soils as a function of volumetric water content and as a function of frequency was studied. The effects of soil temperature and soil electrical conductivity on the complex relative permittivity of soil were investigated. Calibration equations which may be used for predicting volumetric water content of a soil from the real part of the complex relative permittivity of the soil determined by total reflection TDR were developed. The results indicate that for all of the five soils

tested, the real part of the complex relative permittivity decreases with a decrease in volumetric water content and with an increase in frequency. The dispersion in the real part of the complex relative permittivity was observed to be higher at high volumetric water contents as compared with that at the lower ones. Similar to the real part, the imaginary part of the complex relative permittivity of all the five soils was observed to decrease with a decrease in volumetric water content and with an increase in frequency. The imaginary part of the complex relative permittivity showed a pronounced dispersion and soil type dependence.

The effects of soil temperature and electrical conductivity were observed to be higher on the imaginary part when compared with that on the real part the complex relative permittivity. The imaginary part increased in magnitude with an increase in temperature and with an increase in electrical conductivity. The real part did not show a trend similar to that of the imaginary part.

The results observed are consistent with that obtained by Campbell (1990) by using a network analyzer and by Wobschall (1977) by using the theoretical semidisperse soil complex relative permittivity model. The comparisons of the results obtained in this study with those contained in the literature were qualitative rather than quantitative. In many cases, the temperature of the soils for the measurements, the frequency range over which the measurements were conducted, and the type of soils were different and therefore exact quantitative comparisons were impossible.

The method developed is successful in measuring the complex relative permittivity of soils. It is believed that this work has laid the foundation for future use of the total reflection time domain reflectometry and the frequency domain analysis in the studies of dielectric properties of soil.

Bibliography

[1] Arcone, S. A., and R. Wills. 1986. A numerical study of dielectric measurement using single reflection time domain reflectometry. J. Phys. E: Sci. Instrum. 19 : 448–453.

[2] Baker, J. M. and R. R. Allmaras. 1992. System for automating and multiplexing soil moisture measurement by time domain reflectometry. Soil Sci. Am. J. 54 : (1) : 1–6.

[3] Baker, J. M. and R. J. Lascano. 1989. The spatial sensitivity of the time domain reflectometry. Soil Sci. 147(5) : 378–383.

[4] Boned, C., and J. Peyrelasses. 1982. Automatic measurement of complex permittvity (from 2 MHz to 8 GHz) using time domain spectroscopy. J. Phys. Chem. 15 : 534–538.

[5] Borland International, Inc.. 1992. Turob C++ user's guide. Scotts Valley, CA.

[6] Bottreau, A. M., Y. Dutuit, and J. Moreau. 1977. On a multiple reflection time domain method in dielectric spectroscopy: Application to the study of some normal primary alcohols. J. Chem. Phys. 66(8) : 3331–3336.

[7] Bussey, H. E. 1980. Dielectric measurement in shield open circuit coaxial line. IEEE Trans. Instrum. Meas. 29(2) : 120–124.

[8] Camplbell, J. E. 1990. Dielectric properties and influence of conductivity in soils at one to fifty megahertz. Soil. Sci. J. Am. 54 : 332–341.

[9] Chen, C. 1979. One dimensional digital signal processing. Electrical Engineering and Electronics/9. Marcel Dekker Inc., New York.

[10] Chernyak, C. Y. 1967. Dielectric methods for investigating moist soils (Translated from Russian). Israel Program for Scientific Translation, Jerusalem.

[11] Claasen, T. A. C. M., and M. J. C. van Gemert. 1975. Approximate solution in multiple reflection time domain spectroscopy. J. Chem. Phys. 63(1) : 68–73.

[12] Clark, A. H., P. A. Quickenden, and A. Suggett. 1974. Multiple reflection time domain spectroscopy. J. Chem. Phys. 6 : 64–66.

[13] Clarkson, T. S., L. Glaser, R. W. Tuxworth, and G. Willams. 1977. An appreciation of experimental factors in time domain spectroscopy. Advan. Mol. Relax. Processes 10 : 173–202.

[14] Clarkson, T. S., and G. Williams. 1975. Practical aspects of time domain reflectometery. Chem. Phys. Letters. 34(3) : 461–465.

[15] Cole, k. S., and R. H. Cole. 1941. Dispersion and adsorption in dielectrics: I. Alternating current characteristics. J. Chem. Phys. 9 : 341 – 349.

[16] Cole, R. H. 1974. Dielectric response by real time analysis of time domain spectroscopy data. J. Phys. Chem. 78(14) : 1440–1441.

[17] Cole, R. H. 1975. Evaluation of dielectric behavior by time domain spectroscopy. I. Dielectric Response by real time analysis. J. Phys. Chem. 79(14) : 1459–1468.

[18] Cole, R. H. 1975. Evaluation of dielectric behavior by time domain spectroscopy. II. Complex permittivity. J. Phys. Chem. 79 : 1469–1475.

[19] Cole, R. H., S. Mashimo, and P. Winsor. 1980. Evaluation of dielectric behavior by time domain spectroscopy. III. Precision difference methods. J. Phys. Chem. 84 : 786–793.

[20] Cole, R. H., G. Delbos, P. Winsor IV, T. K. Bose, and J. M. Moreau. 1985. Study of dielectric properties of water/oil and oil/water microemulsion by time domain and resonance cavity methods. J. Phys. Chem. 89 : 3338–3343.

[21] Dalton, F. N. 1992. Development of time domain reflectometry for measuring soil water content and bulk soil electrical conductivity. In R. Green and G. C. Topp (ed.). Advances in measurement of soil physical properties: Bringing theory into practice. SSSA Spec. Publ. 30. Soil Sciences Socity of America, Madison, WI.

[22] Dasberg, S. and F. N. Dalton. 1985. Time domain reflectometry field measurements of soil water content and electrical conductivity. Soil Sci. Am. J. 49 : 293–297.

[23] Dasberg, S. and J. W. Hopmans. 1992. Time domain reflectometry calibration for uniformly and nonuniformly wetted sandy and clayey loam soils. Soil Sci. Am. J. 56 : 1341–1345.

[24] Dawkins, A. W., E. H. Grant., and R. J. Sheppard. 1981. An on-line computer based system for performing time domain spectroscopy. III. Presentation of results for total reflection TDS. J. Phys. Chem. E: Sci. Instrum. 14 : 1429–1434.

[25] Dawkins, A. W., R. J. Sheppard, and E. H. Grant. 1979. An on-line computer based system for performing time domain spectroscopy. I. Main features of the basic system. J. Phys. Chem. E: Sci. Instrum. 12 : 1091–1099.

[26] De Loor, G. P., M. J. C. van Gemert, and H. Gravesteyn. 1973. Measurement of dielectric permittivity with time domain reflectometry: Extension of the method to lower relaxation frequencies. Chem Phys. Letters. 18(2) : 295–299.

[27] Debye, P. 1929. Polar molecules. Dover, New York.

[28] Dobson, M. C., F. T. Ulaby, M. T. Hallikanen, M. A. 1985. Microwave dielectric behavior of wet soil. Part II: Dielectric mixing models. IEEE Trans. Geosci. Remote Sensing. 23 : 35–46.

[29] Drungil, C. E. C., K. Abt, and T. J. Gish. 1987. Soil moisture determination in gravelly soils with time domain relectometry. ASAE paper no. 87–2568, American Socity of Agricultural Engineers, St. Joseph, MI.

[30] Fellner-Feldegg, H. 1969. The measurement of dielectric in the time domain. J. Phys. Chem. 73(3) : 616–622.

[31] Fellner-Feldegg, H., and E. F. Barnett. 1970. Reflection of a voltage step from a section of transmission line filled with a polar dielectric. J. Phys. Chem. 74(9) : 1962–1965.

[32] Fellner-Feldegg, H. 1972. A thin-sample method for the measurement of permeability, permittvity, and conductivity in frequency and time domain. J. Phys. Chem. 76(15) : 2116–2122.

[33] Franceschetti, G. 1967. A complete analysis of the reflection and transmission methods for measuring the complex permeability and permittivity of materials at microwaves. Alta Frequenza 36(8) : 757–764.

[34] Gabriel, C., A. W. J. Dawkins, R. J. Sheppard, and E. H. Grant. 1984. Comparison of the single reflection and total reflection TDS techniques. J. Phys. E: Sci. Instrum. 17 : 513–516.

[35] Gabriel, C., E. H. Grant, and I. R. Young. 1986. Use of time domain spectroscopy for measuring dielectric properties with a coaxial probe. J. Phys. E: Sci. Instrum. 19 : 843–846.

[36] Gestblom, B. 1981. On the time referencing problem in time domain reflectometry. J. Phys. E: Sci. Instrum. 14 : 895–896.

[37] Gestblom, B., and B. Jonsson. 1980. A computer controlled dielectric time domain spectrometer. J. Phys. E: Sci. Instrum. 13 : 1067-1070.

[38] Gestblom, B., and E. Noreland. 1977. A new transmission method in dielectric time domain spectroscopy. Chem. Phys. Letters. 47(2) : 349–351.

[39] Gestblom, B. and E. Noreland. 1977. Transmission methods in dielectric time domain spectroscopy. J. Phys. Chem. 81(8) : 782–788.

[40] Giese, K., and R. Tiemann. 1975. Determination of the complex permittivity from thin-sample time domain reflectometry: Improved analysis of the step response waveform. Advan. Mol. Relaxation Processes 7 : 45–59.

[41] Grant, J. P., R. N. Clarke, G. T. Symm, and N. M. Spyrou. 1989. A critical study of the openended coaxial line sensor techique for RF and microwave complex permittivity measurements. J. Phys. E: Sci. Instrum. 22 : 757–770.

[42] Gridley, J. H. 1967. Principles of electrical transmission lines in power and communication. Pergamon Press Ltd. Headington Hill Hall, Oxford, England.

[43] Han, K, C. M. Butler, L. Shen, H. He, and M. A. Harris. 1991. High-frequency, complex dielectric permittvity of saline solution at elevated temperatures. IEEE Trans. Geo. Remo. Sen 29(1) : 48–55.

[44] Hill, N. E. 1969. Dielectric properties and molecular behaviour. Van Nostrand Reinhold. New York.

[45] Heimovaara, T. J. 1994. Frequency domain analysis of time domain reflectometry waveforms I: Measurements of complex dielectric permittvity of soils. Water Resour. Res. 30(2) : 189–199.

[46] Hoekstra, P., and A. Delaney. 1974. Dielectric properties of soils at UHF and microwave frequencies. J. Geophys. Res., 79(11) : 1699–1706.

[47] Iskander, M. F., 1972. Permittvity measurement in time domain. M. Sc. Thesis, University of Manitoba, Canada.

[48] Jordan, B. P., R. J. Sheppard, and S. Szwarnowski. 1978. The dielectric properties of formamide, ethanediol, and methanol. J. Phys. D: Appl. Phys. 11 : 695–701.

[49] Kaatze, U., and K. Giese. 1980. Dielectric relaxation spectroscopy of liquids: Frequency domain and time domain experimental methods. J. Phys. E: Sci. Instrum. 13 : 133–141.

[50] Kao, H. P. 1989. Use of time domain reflectometry for the detection and measurement of cerebral edema. M. Sc. Thesis, University of Manitoba, Canada.

[51] Kraus. J. D., 1984. Electromagnetics, 3rd ed., McGraw-Hill, New York.

[52] Loeb, H. W., G. M. Young, P. A. Quickenden, and A. Suggett. 1971. New methods for measurement of complex permittvity up to 13 GHz and their application to the study of dielectric relaxation of polar liquids including aqueous solutions. Ber. Bunsenges. Phy. Chem. 75(11) : 1155–1165.

[53] MacDonald, D. K. C. 1962. Noise and fluctuations : An introduction. Wiley Inc., New York.

[54] Merabet, M. and T. K. Bose. 1988. Dielectric measurements of water in radio and microwave frequencies by time domain reflectometry. J. Phys. Chem. 92 : 6149–6150.

[55] Nicolson, A. M. 1968. Broad-band microwave transmission characteristics from a single measurement of the transient response. IEEE Trans. Instrum. Meas. 17(4) : 395–402.

[56] Nicolson, A. M. 1973. Forming the fast fourier transform of a set response in time domain metrology. Electron Lett. 9 : 317-318.

[57] Nicolson, A. M., C. L. Bennett, D. Lamensdorf, and L. Susman. 1972. Application of time domain metrology to the automation of broad-band microwave measurements. IEEE Trans. Microwave Theory Tech. 22 : 3–9.

[58] Nicolson, A. M., and G. F. Ross. 1970. Measurement of intrinsic properties of material by time domain techniques. IEEE Trans. Instrum. Meas. 19(4) : 377–382.

[59] Nozaki, R. and T.K. Bose. 1990. Broadband complex permittivity measurement by time domain spectroscopy. IEEE Trans. Instrum. Meas. 39(6) : 945–951.

[60] Parkhomenko, E. I., 1967. Electrical properties of rock, translated from Russian by B. V. Keller, Plenum Press, New York.

[61] Pohl, H. A. 1916 Dielectrophoresis : the behavior of neutral matter in nonuniform electric fields. Cambridge University Press. Cambridge, New York.

[62] Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1988. Numerical recipes in C : The art of scientific computing. Cambriage University Press, MA.

[63] Rzepecka, M. A., S. S. Stuchly, and M. A. K. Hamid. 1973. Modified infinite sample method for routine permittvity measurements at microwave frequencies. IEEE Trans. Instrum. Meas. 22(1) : 41–46.

[64] Stearns, S. D., 1975. Digital signal analysis. Hayden Book Company, Inc. Rochelle Park, New Jersey.

[65] Samulon, H. A. 1951. Spectrum analysis of transient response curves. Proc. IRE 39 : 175–180.

[66] Schwan, H. P., R. J. Sheppard, and E. H. Grant. 1976. Complex permittivity of water at 25^0C. J. Chem. Phys. 64(5) : 2257–2256.

[67] Sheppard, R. J. 1972. An automated coaxial line system for determining the permittvity of liquid. J. Phys. D: Appl. Phys. 5 : 1576-1587.

[68] Smith–Rose. 1933. The electrical properties of soils from alternating currents at radio frequencies, Proc. IEEE 39: 175-181.

[69] Smyth, C. P. 1955. Dielectric behavior and structure; dielectric constant and loss, dipole moment and molecular structure.
McGraw-Hil. New York.

[70] Stuart, R. D. 1961. An introduction to Fourier analysis. John Wiley, New York.

[71] Stuchly, S. S., and M. Matuszewski, 1978. A combined total reflection–transmission method in application to dielectic spectroscopy. IEEE Trans. Instrum. Meas. 27 : 285–289.

[72] Suggett, A., P. A. Mackness, M. J. Tait, H. W. Loeb, and G. M. Young. 1970. Dielectric relaxation studies by time domain spectroscopy. Nature. 228 : 31–32.

[73] Suggett, A. 1975. Microwave dielectric measurements using time domain spectroscopy: Note on recent technique advances. J. Phys. E: Sci. Instrum. 8 : 327–330.

[74] Tocci, R. J. 1980. Digital systems : Principles and applications. Englewood Cliffs, Prentice-Hall, New Jersey.

[75] Topp, G. C., J. L. Davis, and A. P. Annan. 1980. Electromagnetic determination of soil water content: Measurement in coaxial transmission line. Water Resour. Res. 16(3) : 574–582.

[76] Topp, G. C., M. Yanuka, W. D. Zebchuk, and S. Zegelin. 1988. Determination of electrical conductivity using time domain reflectometry: Soil and water experiments in coaxial lines. Water Resour. Res. 24 : 945–952.

[77] van Gemert, M. J. C. 1974. Evaluation of dielectric permittvity and conductivity by time domain spectroscopy. Mathematical analysis of Fellner-Feldegg's thin cell method. J. Chem. Phys. 60(10) : 3963–3974.

[78] van Gemert, M. J. C., and A. Suggett. 1975. Multiple reflection time domain spectroscopy. II. A lumped element approch leading to an analytical solution for the complex permittvity. J. Chem. Phys. 62(7) : 2720–2726.

[79] von Hippel, A. R. 1954. Dielectrics and waves, John Wiley, New York.

[80] von Hippel, A. R. 1954. Dielectric materials and application. John Wiley, New York.

[81] Whittingham, T. A. 1970. Permittvity measurement in the time domain. J. Phys. Chem. 34 : 1824.

[82] Wiltron. 1989. Vector network analyzer operation manual. Wiltron company. Morgan Hill, CA.

[83] Wobschall, D. 1977. A theory of the complex dielectric permittivity of soil containing water: The semidispers model. IEEE Trans. Geosci. Electron. 15(1) : 49–58.

Appendix A

Program description and listings

This appendix contains source codes of five computer programs (*PROG-1.CPP*, *PROG-2.CPP*, *PROG-3.CPP*, *PROG-4.CPP*, and *PROG-5.CPP*) written and used in this study. The purposes of these programs are described and information on how they may be recompiled in the future is given. The programs were written by using Turbo *C++* version 3.0 (Borland International, Inc.)

The first three programs require either one or both of the following two files: *EGAVGA.BGI* (Borland's graphics driver for *EGA* and *VGA*) and *TRIP.CHR* (triplex script font).

PROG-1.CPP This program was written to interface Tektronix 7854 mainframe oscilloscope with an *IBM* compatible computer by using the *GPIB* protocol and to perform waveform data acquisition.

To compile this program, first create a project file and add PROG-1.CPP, EGAVGA.OBJ, *TRIP.OBJ*, and *MCIBL.OBJ* to the project file and copy *DECL.H* to the working directory. *MCIBL.OBJ* and *DECL.H* are National Instruments (*NI*) medium memory model and header files, respectively. Then,

compile the project file. *EGAVGA.BGI* and *TRIP.OBJ* may be converted to linkable object files *EGAVGA.OBJ* and *TRIP.OBJ*, respectively, by using Borland's *BGIOBJ.EXE* utility.

PROG-2.CPP This program was written to perform optimization of the l and the z parameters as explained in section 4.4. The program first approximates the reflected waveform, from the probe immersed in the reference liquid. It, then, computes the residual difference between the approximated waveform and the measured one. It repeats these processes by changing the values of l and z, one at a time, until the residual difference become minimal.

To compile this program, create a project file and add *PROG–2.CPP* and *EGAVGE.OBJ* to the project file and then compile the project file.

PROG-3.CPP This program was written to correct the incident and reflected waveforms for truncation error by a Nicolson (1973) method, to compute the discrete Fourier transform of both waveforms, and to calculate the total reflection coefficient. A *C*–language fast Fourier transform routine written by Press et al. (1988) was included in this program to do the evaluation of the discrete Fourier transforms of the waveforms.

This program may be compiled exactly in the same way as the above program, except that *PROG–2.CPP* should be replaced by *PROG–3.CPP*.

PROG-4.CPP There is no exact analytical solution of equation (4.2) for $\epsilon(f)$. This program was written to solve this equations numerically by using the *Newton Raphson method of iteration.*

PROG-5.CPP This program was written to calculate correction factors for unwated reflections by using the Giese and Tiemann (1975) correction procedure. The

program doesn't need any file. It can be compiled by itself without creating a project file.

These programs were written by *Tilahun Mengesha Worku*, Department of Agricultural Engineering, University of Manitoba.

```
/*****************************************************************************************

                                    PROG-1.CPP

A program to interface Tektronix 7854  mainframe oscilloscope to an  IBM  compatible computer.  Interface  is
achieved by using the General Purpose Interface Bus (GPIB).  A model PCII NI-IEEE 488 board is addressed at
0x2B8 (Hex).  Primary GPIB address of the 7854 oscilloscope is set at 17 and configured using  IBCONF.EXE
(DEV1).  The rear panel switch is set at 10010001 binary [1 (on line  status), 00 (Talk/Listen operational mode),
and 10001 (I/O address of the oscilloscope)].  Prior to addressing the GPIB, the file GPIB.COM must be
installed in the root directory of the computer.

The  program  acquires two wave forms  from  the  oscilloscope  and saves them on the working dirctory on the
computer  with  WAVE1.DAT  and  WAVE2.DAT  filenames.  After  acquisition  of  the  first  wave form, the
program prompts the user to press a key inorder to start acquisition of  the second wave  form.  Each wave form
is an average of a number of real-time wave forms  (specified  by  PROMPT2  in  the  header  of this  program.
Default  100).  At  the  end,  the  program  reads  the  two  wave  form data files and  plot the wave forms on
the computer screen.

                                    Tilahun M. Worku
*****************************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <graphics.h>
#include <complex.h>
#include <conio.h>
#include <math.h>
#include <iostream.h>
#include <dos.h>
#include <time.h>
#include <complex.h>
#include <alloc.h>
#include <process.h>
#include "decl.h"

#define PROMPT0 "SCOPE VMDR HMDA"
#define PROMPT1 "1 0 2 4 >P/W"
#define PROMPT2 "1 0 0 AVG"
#define PROMPT3 "STORED"
#define PROMPT4 "0 WFM SENDX"
#define PROMPT5 "SCOPE"

#define X1 160
#define Y1 80
#define X2 460
#define Y2 320
#define NN 1024

#define FOR_ALL for(i=1; i<=NN; i++)

int  id;
int  gp;
float *wave1, *wave2;
```

```
void Title_page();
void Initialize_GPIB_Boared_and_Oscilloscope()();
void Lock_Front_Panel_of_Oscilloscope();
void Select_Verical_and_Horizontal_Modes();
void Set_Points_Per_Waveform_Value();
void Average_Waveforms_and_Store_in_0WFM_Address();
void Send_Waveform_Quiery();
void Get_First_Waveform_Data_and_Save();
void Prompt_User_to_Press_Any_Key();
void Get_Second_Waveform_Data_and_Save();
void Unlock_Front_Panel_of_Oscilloscope();
void Dynamic_Memory_Allocator();
void Read_Waveform_No1();
void Read_Waveform_No2();
void Draw_Waveforms_on_Computer_Screen();
void Tell_Source_of_error_to_user();
void Clock(int x);

main()
{
  clrscr();
  Title_page();
  Initialize_GPIB_Boared_and_Oscilloscope()();
  Lock_Front_Panel_of_Oscilloscope();
  Select_Verical_and_Horizontal_Modes();
  Set_Points_Per_Waveform_Value();
  /**********Aquire first wave form**********/
  Average_Waveforms_and_Store_in_0WFM_Address();
  Send_Waveform_Quiery();
  Get_First_Waveform_Data_and_Save();
  /****Prompt the user to press any key ****/
  Prompt_User_to_Press_Any_Key();
  /**********Aquire second waveform**********/
  Average_Waveforms_and_Store_in_0WFM_Address();
  Send_Waveform_Quiery();
  Get_Second_Waveform_Data_and_Save();
  Unlock_Front_Panel_of_Oscilloscope();
  Dynamic_Memory_Allocator();
  Read_Waveform_No1();
  Read_Waveform_No2();
  Draw_Waveforms_on_Computer_Screen();

  return 0;
}

void Title_page()
{
 register int i;

 int color, gdriver = DETECT, gmode, errorcode;
 float yf = 480.0/200.0;
 char c1[10], c2[60], b1[1], b2[1];

 clrscr();
```

```
 normvideo();
 errorcode = registerbgidriver(EGAVGA_driver);
 /* report any drive registration errors */
 if (errorcode < 0)
 {
   printf("Graphics error: %s\n", grapherrormsg(errorcode));
   printf("Press any key to halt:");
   getch();
   exit(1);  /* terminate with an error code */
 }
 /* Initialize graphics and local variables */
 initgraph(&gdriver, &gmode, "");
 /* read result of initialization */
 errorcode = graphresult();
 if (errorcode != grOk)  /* an error occurred */
 {
   printf("Graphics error: %s\n", grapherrormsg(errorcode));
   printf("Press any key to halt:");
   getch();
   exit(1); /* terminate with an error code */
 }
 setpalette(0,WHITE);
 setcolor(EGA_RED);
 settextstyle(1, HORIZ_DIR, 1);
 outtextxy(185,240,"A TDR Wave form Acquisition Routine");
 outtextxy(150,260,"(For Tektronix 7854 Mainframe Oscilloscope)");
 setcolor(EGA_BLUE);
 outtextxy(280,340,"Written by :");
 outtextxy(260,380,"Tilahun M. Worku");
 outtextxy(175,400,"Department of Agricultural Engineering");
 outtextxy(280,420,"June 1994");
 setcolor(EGA_LIGHTGREEN);
 outtextxy(230,455,"  * Please Wait * ");
 errorcode = registerbgifont(triplex_font);
 /* report any font registration errors */
 if (errorcode < 0)
 {

   printf("Graphics error: %s\n", grapherrormsg(errorcode));
   printf("Press any key to halt:");
   getch();
   exit(1);
 }
 setcolor(EGA_GREEN);
 setlinestyle(SOLID_LINE, 1,3);
 rectangle(2,2,637, 477);
 setlinestyle(SOLID_LINE, 1,3);
 setlinestyle(DOTTED_LINE,1,1);
 settextstyle(TRIPLEX_FONT, HORIZ_DIR, i);
 setcolor(EGA_BLUE);
 textwidth(c1);
 textheight(c2);
 for(i=1; i<=10; i++)
 {
```

```
    outtextxy(150+i, 20*yf+i, "TWAR");
    outtextxy(151+i, 20*yf+i, "TWAR");
  }
  outtextxy(152+i, 20*yf+i, "TWAR");
  Clock(7);
  graphdefaults();
  restorecrtmode();
  closegraph();
}

//   This function set duration of display of the title-page

void Clock(int x)
{
   time_t Clock_start_time, Clock_end_time;
   Clock_start_time = time(NULL);
   for(;;)
   {
   Clock_end_time = time(NULL);
   if ((Clock_end_time)-(Clock_start_time) > x) break;
   }
}

void Initialize_GPIB_Boared_and_Oscilloscope()()
{
   gp = ibfind("gpib0");
   if(gp < 0)
            Tell_Source_of_error_to_user();
   id = ibfind("DEV1");
   if(id < 0)
            Tell_Source_of_error_to_user();
   printf("\n\n\n          Connected to Tektronix 7854 Oscilloscope..id# = %d\n\n\n\n",id);
}

void Lock_Front_Fanel_of_Oscilloscope()
{
   ibsre(gp, 0);          // Turn REN signal on.   REN is used to select b/n remote and local mode.
   ibcmd(gp, "Q", 1);  // Send My Listen Address (MLA) of the device. The oscilloscope will not enter the
                          // desired   mode  until it recieves it listen address.
}

void Select_verical_and_horizontal_modes()
{
   ibwrt(id, PROMPT0, 15);
}

/* Set points/waveform value. P/W can take the following values : 128, 256, 512, and 1024.
   The higher the P/W value, the greater the resolution */

void Set_Points_Per_Waveform_Value()
{
          ibwrt(id, PROMPT1, 12);
}
```

```
void Average_Waveforms_and_Store_in_0WFM_Address()
{
   ibwrt(id,PROMPT2, 9);
   gotoxy(18, 10);
   printf("Now averaging 100 real-time wave forms...\n");
   ibwrt(id,PROMPT3, 6);
   Clock(10);
}

void Send_Waveform_Quiery()
{
  clrscr();
  gotoxy(17, 10);
  printf("Now recieving & processing averaged wave form ...\n");
  ibwrt(id, "0 WFM SENDX", 11);
  if((ibsta >= 0) & (id > 0) & (ibsta < 4096) & (iberr != 6) ){}
  else Tell_Source_of_error_to_user();
}

void Get_First_Waveform_Data_and_Save()
{
        ibrdf(id, "WAVE1.DAT");
        ibclr(id);
        Clock(10);
}

void Get_Second_Waveform_Data_and_Save()
{
        ibrdf(id, "WAVE2.DAT");
        ibclr(id);
        Clock(10);
}

void Prompt_User_to_Press_Any_Key()
{
  clrscr();
  gotoxy(15,7);
  printf(" Press Any Key to Acquier Second Waveform ...");
  getch();
}

void Unlock_Front_Panel_of_Oscilloscope()()
{
  ibsre(gp,0);                  // Turn remote enable signal off
  ibcmd(gp, "Q", 1);            // Q is MYL(My Listen Address) of 7854 Dig.
}

void Dynamic_Memory_Allocator()
{
   wave1 = (float *)malloc(NN*sizeof(float));
   wave2 = (float *)malloc(NN*sizeof(float));
}

void Read_Waveform_No1()
```

```
{

 register int i, j, k;

 char C, P_buffer[6];
 FILE *fp1;

 clrscr();
 P_buffer[5] = '\0';
 fp1 = fopen("WAVE1.DAT", "r");
 for(;;)
 {
        C = fgetc(fp1);
        if(C == 'C')
        {
          P_buffer[0] = C;
          C = fgetc(fp1);
          if(C == 'U')
          {
                  P_buffer[1] = C;
                  C = fgetc(fp1);
                  if(C == 'R')
                  {
                    P_buffer[2] = C;
                    C = fgetc(fp1);
                    if(C == 'V')
                    {
                            P_buffer[3] = C;
                            C = fgetc(fp1);
                            if(C == 'E');
                            {
                              P_buffer[4] = C;
                              break;
                            }
                    }
                  }
          }
        }
}
FOR_ALL
{
        char Tempo2[15];

        j=0;
        for(;;)
        {
                C = fgetc(fp1);
                if(C == ',' || C == EOF )    // <-- After the last data point, there is  an end of
                {                            //     file character. The second expression tests to
                                             //     see whether EOF is reached or not.
                        *(wave1+i) = atof(Tempo2);
                        break;
                }
                Tempo2[j++] = C;
```

```
                Tempo2[j] = '\0';
        }
  }
  fclose(fp1);
}

void Read_Waveform_No2()
{
  register int i, j, k;

  char C, P_buffer[6];
  FILE *fp1;

  clrscr();
  P_buffer[5] = '\0';
  fp1 = fopen("WAVE2.DAT", "r");
  for(;;)
  {
        C = fgetc(fp1);
        if(C == 'C')
        {
          P_buffer[0] = C;
          C = fgetc(fp1);
          if(C == 'U')
          {
                P_buffer[1] = C;
                C = fgetc(fp1);
                if(C == 'R')
                {
                  P_buffer[2] = C;
                  C = fgetc(fp1);
                  if(C == 'V')
                  {
                        P_buffer[3] = C;
                        C = fgetc(fp1);
                        if(C == 'E');
                        {
                          P_buffer[4] = C;
                          break;
                        }
                  }
                }
          }
        }
  }
  FOR_ALL
  {
        char Tempo2[15];

        j=0;
        for(;;)
        {
                C = fgetc(fp1);
                if(C == ',' || C == EOF )
```

```
                {

                        *(wave2+i) = atof(Tempo2);
                        break;
                }
                Tempo2[j++] = C;
                Tempo2[j] = '\0';
        }
  }
  fclose(fp1);
}

void  Tell_Source_of_error_to_user()
{
   if(id < 0)       printf("\n\n Device is not installed. Use IBCONF.EXE then reboot");
   if(iberr == 0)  printf("\n\n Dos error(Is device installed");
   if(iberr == 1)  printf("\n\n Function requires GPIB-PC to be CIC");
   if(iberr == 2)  printf("\n\n No listener on write function");
   if(iberr == 3)  printf("\n\n GPIB-PC not addressed correctly");
   if(iberr == 4)  printf("\n\n Invalide argument to function call");
   if(iberr == 5)  printf("\n\n GPIB-PC not system controller as required");
   if(iberr == 6)  printf("\n\n Timeout error.  GPIB Error ");
   if(iberr == 7)  printf("\n\n Non existent GPIB-PC board");
   if(iberr == 10) printf("\n\n I/O started before pervious operation completed");
   if(iberr == 12) printf("\n\n File system error");
   if(iberr == 14) printf("\n\n Command error during device call");
   if(iberr == 15) printf("\n\n Serial poll status byte lost");
   if(iberr == 16) printf("\n\n SRQ stuck in on position");
}

void  Draw_Waveforms_on_Computer_Screen()
{
 register int i;
 float XF, YF;
 int color, gdriver = DETECT, gmode, errorcode;
 float *(datag1), *(datag2);

 clrscr();
 datag1 = (float *)malloc(NN*sizeof(float));
 datag2 = (float *)malloc(NN*sizeof(float));
 XF = float(X2-X1)/float(NN);
 YF = float(Y2-Y1)/7.7;
 FOR_ALL
 {
   *(datag1+i) = int(*(wave1+i)*YF);
   *(datag2+i) = int(*(wave2+i)*YF);
 }
 normvideo();
 if (errorcode < 0)
 {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1);
```

```
  }
  initgraph(&gdriver, &gmode, "");
  errorcode = graphresult();
  if (errorcode != grOk)
  {
          printf("Graphics error: %s\n", grapherrormsg(errorcode));
          printf("Press any key to halt:");
          getch();
          exit(1);
  }
  highvideo();
  setpalette(0, EGA_WHITE);
  setcolor(EGA_GREEN);
  setlinestyle(SOLID_LINE, 1,3);
  rectangle(1, 1, 638, 478);
  setcolor(EGA_BLUE);
  setlinestyle(DOTTED_LINE,1,1);
  for(i=1; i<10; i++) line(X1+i*(int(float(X2-X1)/10.0)),Y1,X1+i*(int(float(X2-X1)/10.0)),Y2);
  for(i=1; i<8 ; i++) line(X1,Y1+i*(int(float(Y2-Y1)/8.0)), X2, Y1+i*(int(float(Y2-Y1)/8.0)));
  setlinestyle(DOTTED_LINE,1,2);
  setcolor(LIGHTMAGENTA);
  line(X1+5*(int(float(X2-X1)/10.0)),Y1, X1+5*(int(float(X2-X1)/10.0)),Y2);
  line(X1,Y1+4*(int(float(Y2-Y1)/8.0)), X2, Y1+4*(int(float(Y2-Y1)/8.0)));
  setlinestyle(SOLID_LINE,1,1);
  setcolor(EGA_LIGHTBLUE);
  for(i=1; i<=1023;i++) line(X1+int(XF*float(i)), (Y1+4*(int(float(Y2-Y1)/8.0)))-*(datag1+i),
                                                  X1+int(float(i+1)*XF),
(Y1+4*(int(float(Y2-Y1)/8.0)))-*(datag1+(i+1)));
  setcolor(EGA_LIGHTRED);
  for(i=1; i<=1023;i++) line(X1+int(XF*float(i)), (Y1+4*(int(float(Y2-Y1)/8.0)))-*(datag2+i),
                                                  X1+int(float(i+1)*XF),
(Y1+4*(int(float(Y2-Y1)/8.0)))-*(datag2+(i+1)));
  setlinestyle(SOLID_LINE,1,1);
  setcolor(EGA_GREEN);
  setlinestyle(SOLID_LINE, 1,3);
  rectangle(X1, Y1, X2, Y2);
  settextstyle(1, HORIZ_DIR, 1);
  setcolor(EGA_LIGHTBLUE);
  outtextxy(210,455,"   * Press Any Key *  ");
  getch();
  restorecrtmode();
  closegraph();
}
```

```
/*****************************************************************************************
                                    PROG-2.CPP

This program finds optimum values of the l and the z parameters in the S11(f) equation. The program first
approximates the reflected wave form from a coaxial probe immersed in ethanol and then minimizes the sum of
squares of the residual differences between the approximated and the measured wave forms. It repeates this
process until the residual difference becomes minimal. The values of l and z, when the residual difference is
minimal, are taken to be the optimized values.

                                    Tilahun M. Worku
*****************************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <complex.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <iostream.h>
#include <dos.h>
#include <time.h>
#include <complex.h>

#define PER_FREE_SPACE  8.854e-12
#define CL              3e+8

#define X11 170
#define Y11 90
#define X21 470
#define Y21 390

#define INTERCHANGE(a,b)  tempr=(a); (a)=(b); (b)=tempr
#define NN                1024
#define FOR_ALL           for(i=1; i<=NN; i++)

double  far *wave1, far *wave2, far *wave3, far *Twave1, far *Twave2,
        far *wave1c, far *wave2c, far *Freq, Current_sum, DataTI, DeltaF,
        far *S11_Mat, far *S11n_Mat, far *S11_Ref, far *S11n_Ref;

double  epsilon_s = 24.4, epsilon_inf = 4.8, f_relax = 1.14, Beta = 0.0,
        R, b1 = 1.0 - Beta, Sigma_dc = 0.0, z = 0.3398,  L = 0.0461

void   Dynamic_Memory_Allocater();
void   Open_and_Read_waveform_Files();
void   Calculate_Data_Time_Interval_And_Frequency_Values();
void   Complex_Form();
void   Fast_Fourier_Transform_Waveform1(int isign);
void   Fast_Fourier_Transform_Waveform2(int isign);
void   Calculate_S11_Ref_Material_And_FFT_of_wave2(double zz, double LL);
void   Calculate_Wave2_And_Store_In_Wave3();
void   Intilaize_Graphics_Mode();
void   Draw_Actual_And_Approximation_Waveforms();
void   Calculate_Sum_Of_Square_Of_The_Differences();
void   Find_Optimal_Values_Of_Probe_Parameters();
```

```
void   Read_S11_Of_Material();
void   Calculate_S11();
void   Clock(int x);

main()
{
  clrscr();
  Dynamic_Memory_Allocater();
  Open_and_Read_waveform_Files();
  Calculate_Data_Time_Interval_And_Frequency_Values();
  Complex_Form();
  Fast_Fourier_Transform_Waveform1(1);
  Find_Optimal_Values_Of_Probe_Parameters();

  return 0;
}

void  Find_Optimal_Values_Of_Probe_Parameters()
{
register int i;

double  ZZ = 0.3320, LL = 0.0410, Tempo_Sum = 1000.0;
FILE *fp1;

fp1 = fopen("Tempo.dat", "w");
for(i=1;;i++)
{
 char   C;

 if(kbhit() != 0)
 {
          C = getch();
          if(C == 'x') break;
 }
 Calculate_S11_Ref_Material_And_FFT_of_wave2(ZZ, LL);
 Calculate_Wave2_And_Store_In_Wave3();
 Calculate_Sum_Of_Square_Of_The_Differences();
 Intilaize_Graphics_Mode();
 ZZ += 0.0002;
  LL -= 0.001;
 if((Tempo_Sum  < Current_sum)) break;
 fprintf(fp1,"Iteration No. = %d\n", i);
 fprintf(fp1,"z = %f\n", ZZ);
 fprintf(fp1,"L = %f\n", LL);
 fprintf(fp1,"Sum of square of the differences    =    %f\n\n", Current_sum);
 cout<<"Sum of square of the differences   =    "<<Current_sum<<"\n";Clock(1);
 Tempo_Sum = Current_sum;
}
fclose(fp1);
clrscr();
system("Type Tempo.dat | more");
}

void  Dynamic_Memory_Allocater()
```

```
{
  Freq     = (double *)malloc(NN*sizeof(double));
  wave1    = (double *)malloc(NN*sizeof(double));
  wave2    = (double *)malloc(NN*sizeof(double));
  wave3    = (double *)malloc(NN*sizeof(double));
  wave1c   = (double *)malloc(NN*sizeof(double));
  wave2c   = (double *)malloc(NN*sizeof(double));
  Twave1   = (double *)malloc((NN<<1)*sizeof(double));
  Twave2   = (double *)malloc((NN<<1)*sizeof(double));
  S11_Ref  = (double *)malloc((NN<<1)*sizeof(double));
  S11_Mat  = (double *)malloc((NN<<1)*sizeof(double));
  S11n_Mat = (double *)malloc(NN*sizeof(double));
  S11n_Ref = (double *)malloc(NN*sizeof(double));
}

void Open_and_Read_waveform_Files()
{
 register int i, j, k;

 char C, P_buffer[6];
 FILE *fp1, *fp2;

 clrscr();
 P_buffer[5] = '\0';
 fp1 = fopen("wave1.Dat", "r");
 for(;;)
 {
        C = fgetc(fp1);
        if(C == 'C')
        {
          P_buffer[0] = C;
          C = fgetc(fp1);
          if(C == 'U')
          {
                  P_buffer[1] = C;
                  C = fgetc(fp1);
                  if(C == 'R')
                  {
                    P_buffer[2] = C;
                    C = fgetc(fp1);
                    if(C == 'V')
                    {
                            P_buffer[3] = C;
                            C = fgetc(fp1);
                            if(C == 'E');
                            {
                              P_buffer[4] = C;
                              break;
                            }
                    }
                  }
          }
        }
}
```

```
FOR_ALL
{
        char Tempo2[15];

        j=0;
        for(;;)
        {
                C = fgetc(fp1);
                if(C == ',' || C == EOF )
                {
                  *(wave1+i) = atof(Tempo2);
                  break;
                }
                Tempo2[j++] = C;
                Tempo2[j] = '\0';
        }
}
fclose(fp1);
P_buffer[5] = '\0';
fp2 = fopen("wave2.Dat", "r");
for(;;)
{
        C = fgetc(fp2);
        if(C == 'C')
        {
          P_buffer[0] = C;
          C = fgetc(fp2);
          if(C == 'U')
          {
                P_buffer[1] = C;
                C = fgetc(fp2);
                if(C == 'R')
                {
                  P_buffer[2] = C;
                  C = fgetc(fp2);
                  if(C == 'V')
                  {
                        P_buffer[3] = C;
                        C = fgetc(fp2);
                        if(C == 'E');
                        {
                          P_buffer[4] = C;
                          break;
                        }
                  }
                }
          }
        }
}
FOR_ALL
{
        char Tempo2[15];

        j=0;
```

```
        for(;;)
        {
                C = fgetc(fp2);
                if(C == ',' || C == EOF )
                {
                  *(wave2+i) = atof(Tempo2);
                  break;
                }
                Tempo2[j++] = C;
                Tempo2[j] = '\0';
        }
  }
  fclose(fp2);
}

void Calculate_Data_Time_Interval_And_Frequency_Values()
{
 register int i, j, k;

  char C, Buffer[6], Tempo2[15], c[9];
  FILE *fp1;

  clrscr();
  Buffer[6] = '\0';
  fp1 = fopen("wave1.Dat", "r");
  for(;;)
  {
        C = fgetc(fp1);
        if(C == 'X')
        {
          Buffer[0] = C;
          C = fgetc(fp1);
          if(C == 'I')
          {
                Buffer[1] = C;
                C = fgetc(fp1);
                if(C == 'N')
                {
                  Buffer[2] = C;
                  C = fgetc(fp1);
                  if(C == 'C')
                  {
                        Buffer[3] = C;
                        C = fgetc(fp1);
                        if(C == 'R');
                        {
                          Buffer[4] = C;
                          break;
                        }
                  }
                }
          }
        }
 }
```

```
  c[9] = '\0';
  C = fgetc(fp1);
  fread(c, sizeof(c), 1, fp1);
  DataTI = atof(c);
  DeltaF = 1.0/(NN*DataTI);
  fclose(fp1);
  FOR_ALL
  {
        if(i<=((NN/2)+1))
          *(Freq+i) = (i-1)*DeltaF*1e-9;
        else
          *(Freq+i) = (i-1-NN)*DeltaF*1e-9;
  }
}

void  Complex_Form()
{
        register int i, k;

        double const Zero1 = *(wave1+1);
        double const Zero2 = *(wave2+1);

        FOR_ALL
        {
         *(wave1c+i) = *(wave1+i) - Zero1;
         *(wave2c+i) = *(wave2+i) - Zero2;
        }
        double const Inci_nn = *(wave1c+NN);
        double const Refl_nn = *(wave2c+NN);
        FOR_ALL
        {
          *(wave1c+i) -= Inci_nn*(double(i-1))/double(NN-1);
          *(wave2c+i) -= Refl_nn*(double(i-1))/double(NN-1);
        }
        FOR_ALL
        {
          k = i<<1;

          *(Twave1+(k-1)) = *(wave1c+i);
          *(Twave1+k) = 0.0;
          *(Twave2+(k-1)) = *(wave2c+i);
          *(Twave2+k) = 0.0;
        }
}

void Fast_Fourier_Transform_Waveform1(int isign)
{
register int i, j, m, y;

double  wtemp, wr, wpr, wpi, wi, theta;
double  tempr, tempi;
int     n, mmax, istep;

FOR_ALL
```

```
{
  y = i<<1;

  *(Twave1+y) = -(*(Twave1+y));
}
n = NN<<1;
j=1;
for(i=1; i<n; i+=2)
{
        if(j>i)
        {
          INTERCHANGE(*(Twave1+j), *(Twave1+i));
          INTERCHANGE(*(Twave1+(j+1)), *(Twave1+(i+1)));
        }
        m=n>>1;
        while(m >= 2 && j > m )
        {
          j -= m;
          m >>= 1;
        }
        j += m;
}

mmax = 2;
while(n > mmax)
{
 istep = 2*mmax;
 theta = 2.0*M_PI/(isign*mmax);
 wtemp = sin(0.5*theta);
 wpr = -2.0*wtemp*wtemp;
 wpi = sin(theta);
 wr = 1.0;
 wi = 0.0;
 for(m = 1; m < mmax; m += 2)
 {
  for(i=m; i<=n; i+=istep)
  {
        j =i+mmax;
        tempr = wr*(*(Twave1+j))-wi*(*(Twave1+(j+1)));
        tempi = wr*(*(Twave1+(j+1)))+wi*(*(Twave1+j));
        *(Twave1+j) = *(Twave1+i)-tempr;
        *(Twave1+(j+1)) = *(Twave1+(i+1))-tempi;
        *(Twave1+i)  += tempr;
        *(Twave1+(i+1)) += tempi;
  }
  wr = (wtemp=wr)*wpr-wi*wpi+wr;
  wi = wi*wpr+wtemp*wpi+wi;
 }
 mmax = istep;
}
FOR_ALL  //....//
{
  y = i<<1;
```

```
  *(Twave1+y) = -(*(Twave1+y));
 }
}

void Fast_Fourier_Transform_Waveform2(int isign)
{
 register int i, j, m, y;

 double  wtemp, wr, wpr, wpi, wi, theta;
 double  tempr, tempi;
 int     n, mmax, istep;

 FOR_ALL
 {
  y = i<<1;

  *(Twave2+y) = -(*(Twave2+y));
 }
 n = NN<<1;
 j=1;
 for(i=1; i<n; i+=2)
 {
        if(j>i)
        {
          INTERCHANGE(*(Twave2+j), *(Twave2+i));
          INTERCHANGE(*(Twave2+(j+1)), *(Twave2+(i+1)));
        }
        m=n>>1;
        while(m >= 2 && j > m )
        {
          j -= m;
          m >>= 1;
        }
        j += m;
}

mmax = 2;
while(n > mmax)
{
 istep = 2*mmax;
 theta = 2.0*M_PI/(isign*mmax);
 wtemp = sin(0.5*theta);
 wpr = -2.0*wtemp*wtemp;
 wpi = sin(theta);
 wr = 1.0;
 wi = 0.0;
 for(m = 1; m < mmax; m += 2)
 {
  for(i=m; i<=n; i+=istep)
  {
        j =i+mmax;
        tempr = wr*(*(Twave2+j))-wi*(*(Twave2+(j+1)));
        tempi = wr*(*(Twave2+(j+1)))+wi*(*(Twave2+j));
        *(Twave2+j) = *(Twave2+i)-tempr;
```

```
        *(Twave2+(j+1)) = *(Twave2+(i+1))-tempi;
        *(Twave2+i)  += tempr;
        *(Twave2+(i+1)) += tempi;
  }
  wr = (wtemp=wr)*wpr-wi*wpi+wr;
  wi = wi*wpr+wtemp*wpi+wi;
 }
 mmax = istep;
 }
 FOR_ALL
 {
  y = i<<1;

  *(Twave2+y) = -(*(Twave2+y));
 }
}

void  Calculate_S11_Ref_Material_And_FFT_of_wave2(double ZZ, double LL)
{
 register int i, j;

 double far *Epsilon, z = ZZ, L = LL;

 complex J = complex(0.0, 1.0);
 Epsilon  = (double *)malloc((NN<<1)*sizeof(double));
 FOR_ALL
 {
 register int k;

 k = i<<1;
 complex a1  =  J;                 // a1  =  j
 complex a2  =  epsilon_inf;       // a2  =  ε∞
 complex a3  =  epsilon_s;         // a3  =  εs
 complex a4  =  *(Freq+i)*a1;      // a4  =  jf
 complex a5  =  a4/f_relax;        // a5  =  jf/frel
 complex a6  =  pow(a5, b1);       // a6  =  (jf/frel)^(1-ß)
 complex a7  =  1.0 + a6;          // a7  =  1 + (jf/frel)^(1-ß)
 complex a8  =  a3 - a2;           // a8  =  (εs - ε∞)
 complex a9  =  a8/a7;             // a9  =  (εs - ε∞)/(1 + (jf/frel)^(1-ß))
 complex a10 =  a2 + a9;           // a10 =  ε∞ + (εs - ε∞)/(1 + (jf/frel)^(1-ß))
 if(i == 1)
 {
       complex a15   =  a10;       // a15 =  [ε∞ + (εs - ε∞)/(1 + (jf/frel)^(1-ß))]
       *(Epsilon+(k-1)) = real(a15);
       *(Epsilon+k) = imag(a15);
 }
 else
 {
 complex a11 =  a1*Sigma_dc;               // a11 = jσdc
 complex a12 =  2.0*M_PI*(*(Freq+i));      // a12 = 2πf
 complex a13 =  a12*PER_FREE_SPACE;        // a13 = 2πfεo
 complex a14 =  a11/(a13*1e+9);            // a14 = jσdc/2πfεo
 complex a15 =  a10 - a14;                 // a15 = [ε∞ + (εs - ε∞)/(1 + (fj/frel)^(1-ß))] - iσdc/2πfεo
```

```
                                                    // THIS IS COLE-COLE EQUATION
   *(Epsilon+(k-1)) = real(a15);
   *(Epsilon+k) = imag(a15);
  }
  }
  FOR_ALL
  {
        register int k;

        k = i<<1;
        R = -2.0*M_PI*(*(Freq+i)*1e+9)*L/CL;   // R   = -2πfL/c
        complex c0  = complex(*(Epsilon+(k-1)), *(Epsilon+k));
                                                // c0  = ε
        complex c1  = sqrt(c0);                 // c1  = √ε
        complex c2  = R*J;                      // c2  = Rj
        complex c3  = c2*c1;                    // c3  = jR√ε
        complex c4  = 2.0*c3;                   // c4  = 2jR√ε
        complex c5  = exp(c4);                  // c5  = e^(2jR√ε)
        complex c6  = complex(z);               // c6  = z
        complex c7  = c6*c1;                    // c7  = z√ε
        complex c8  = complex(1.0);             // c8  = 1.0
        complex c9 =  (c8 - c7)/(c8 + c7);      // c9 = (1-z√ε)/(1+z√ε)
        complex c10 = c5*c9;                    // c10 = e^(2jR√ε)((1-z√ε)/(1+z√ε))
        complex c11 = (c9 + c5)/(c8 + c10);     // c11 = (((1-z√ε)/(1+z√ε)) +
                                                        e^(2jR√ε))/(1+e^(2jR√ε)((1-z√ε)/(1+z√ε)))
        *(S11_Ref+(k-1)) = real(c11);
        *(S11_Ref+k)     = imag(c11);
        *(S11n_Ref+i)    = norm(c11);
  }
}

void Calculate_Wave2_And_Store_In_Wave3()
{
 register int i, k;


 FOR_ALL
 {
        k = i<<1;

        complex C0 = complex(*(S11_Ref+(k-1)), *(S11_Ref+k))*
                     complex(*(Twave1+(k-1)), *(Twave1+k));
        *(Twave2+(k-1)) = real(C0);
        *(Twave2+k)     = imag(C0);
 }
 Fast_Fourier_Transform_Waveform2(-1);
 FOR_ALL
 {
   k = i<<1;

        *(wave3+i) = *(Twave2+(k-1))*(1.0/NN);
 }
 double const temp = *(wave3+1);
```

```
 FOR_ALL
 {
   k = i<<1;
   *(wave3+i) -= temp;
 }
}

void Calculate_Sum_Of_Square_Of_The_Differences()
{
  register int i;

  double Sum = 0.0;

  FOR_ALL
  {
        double Difference, Square_Difference;

        Difference = *(wave3+i)-(*(wave2c+i));
        Square_Difference = pow(Difference, 2);
        Sum += Square_Difference;
        Current_sum = Sum;
  }
}

void Intilaize_Graphics_Mode()
{
 register int i;
 int color, gdriver = DETECT, gmode, errorcode;

 clrscr();
 normvideo();
 initgraph(&gdriver, &gmode, "");
 errorcode = graphresult();
 if (errorcode != grOk)
 {
          printf("Graphics error: %s\n", grapherrormsg(errorcode));
          printf("Press any key to halt:");
          getch();
          exit(1);
 }
 highvideo();
 setpalette(0, EGA_WHITE);
 setcolor(EGA_GREEN);
 setlinestyle(SOLID_LINE, 1,3);
 rectangle(1, 1, 638, 478);
 Draw_Actual_And_Approximation_Waveforms(); getch();
 settextstyle(1, HORIZ_DIR, 1);
 setcolor(EGA_LIGHTGRAY);
 outtextxy(203,465,"   * Press  'x'  To Exit *   ");
 restorecrtmode();
 closegraph();
}

void Draw_Actual_And_Approximation_Waveforms()
```

```
{
 register int i, k;

 double far *wave2g, far *wave3g, XF, YF;

 wave3g    = (double *)malloc(NN*sizeof(double));
 wave2g    = (double *)malloc(NN*sizeof(double));

 XF = double(X21-X11)/double(NN);
 YF = double(Y21-Y11)/7.7;

 FOR_ALL
 {
   *(wave3g+i) = int(*(wave3+i)*YF);
   *(wave2g+i) = int(*(wave2c+i)*YF);
 }
 setcolor(EGA_BLUE);
 setlinestyle(DOTTED_LINE,1,1);
 for(i=1; i<10; i++) line(X11+i*(int(double(X21-X11)/10.0)),Y11,X11+i*(int(double(X21-X11)/10.0)),Y21);
 for(i=1; i<8 ; i++) line(X11,Y11+i*(int(double(Y21-Y11)/8.0)), X21, Y11+i*(int(double(Y21-Y11)/8.0)));
 setlinestyle(DOTTED_LINE,1,2);
 setcolor(LIGHTMAGENTA);
 line(X11+5*(int(double(X21-X11)/10.0)),Y11, X11+5*(int(double(X21-X11)/10.0)),Y21);
 line(X11,Y11+4*(int(double(Y21-Y11)/8.0)), X21, Y11+4*(int(double(Y21-Y11)/8.0)));
 setlinestyle(SOLID_LINE,1,1);
 setcolor(EGA_LIGHTRED);
 for(i=1;i<=NN-1;i++) line(X11+int(XF*double(i)), (Y11+4*(int(double(Y21-Y11)/8.0)))-*(wave3g+i),
                   X11+int(double(i+1)*XF), (Y11+4*(int(double(Y21-Y11)/8.0)))-*(wave3g+(i+1)));
 line(X11+30, Y21+30, X11+60, Y21+30);
 outtextxy(X11+100, Y21+30, "Modelled Waveform");
 setlinestyle(SOLID_LINE,1,1);
 setcolor(EGA_LIGHTBLUE);
 for(i=1;i<=NN-1;i++) line(X11+int(XF*double(i)), (Y11+4*(int(double(Y21-Y11)/8.0)))-*(wave2g+i),
                   X11+int(double(i+1)*XF), (Y11+4*(int(double(Y21-Y11)/8.0)))-*(wave2g+(i+1)));
 line(X11+30, Y21+50, X11+60, Y21+50);
 outtextxy(X11+100, Y21+50, "Observed Waveform");
 setlinestyle(SOLID_LINE,1,1);
 setcolor(EGA_GREEN);
 setlinestyle(SOLID_LINE, 1,3);
 rectangle(X11, Y11, X21, Y21);
 outtextxy(X11-10,((Y11+Y21)/2)-3,"0");
 setcolor(EGA_BLUE);
}

void Clock(int x)
{
 time_t Clock_start_time, Clock_end_time;
 Clock_start_time = time(NULL);

 for(;;)
 {
   Clock_end_time = time(NULL);
   if ((Clock_end_time)-(Clock_start_time) > x) break;
 }
```

```
}
```

```
/*****************************************************************************

                                PROG-3.CPP

This is program first correctes the incident and the reflected wave forms for truncation error and it then computes the discrete Fourier transform of both wave forms. From the transforms, it finally calculates the S11(f).

                                Tilahun M. Worku

*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <complex.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <iostream.h>
#include <dos.h>
#include <time.h>
#include <complex.h>

#define X11 80
#define Y11 10
#define X21 280
#define Y21 210
#define X12 80
#define Y12 240
#define X22 280
#define Y22 440
#define X13 360
#define Y13 10
#define X23 560
#define Y23 210
#define X14 360
#define Y14 240
#define X24 560
#define Y24 440

#define INTERCHANGE(a,b)  tempr=(a); (a)=(b); (b)=tempr
#define NN   1024
#define FOR_ALL          for(i=1; i<=NN; i++)
#define NUM  70
#define FOR_NUM          for(i=1; i<=NUM; i++)

#define FHL "0.7 GHz"

float  far *wave2, far *wave1, far *Twave2, far *Twave1, DataTI, DeltaF,
       far *wave1c, far *wave2c, far XF, far YF;


void   Dynamic_Memory_Allocater();
```

```
void   Read_waveform_Files();
void   Calculate_Data_Time_Interval();
void   Set_Zero_Level();
void   Apply_Correction_for_truncation_Nicolson73();
void   Fast_Fourier_Transform_Waveform1(int isign);
void   Fast_Fourier_Transform_Waveform2(int isign);
void   Write_Computed_S11_to_File();
void   Intilaize_Graphics_Mode_and_Draw_Waveforms();
void   Draw_Original_Waveforms();
void   Draw_Waveforms_Corrected_for_Truncation();
void   Draw_Spectra_of_Waveforms();
void   Draw_Scattring_Cofficient_vs_Frequency();
void   Clock(int x);

main()
{

 register int i, k;

 clrscr();
 Dynamic_Memory_Allocater();
 Read_waveform_Files();
 Calculate_Data_Time_Interval();
 Set_Zero_Level();
 Apply_Correction_for_truncation_Nicolson73();
 Fast_Fourier_Transform_Waveform1(1);
 Fast_Fourier_Transform_Waveform2(1);
 Write_Computed_S11_to_File();
 Intilaize_Graphics_Mode_and_Draw_Waveforms();

 return 0;
}

void  Dynamic_Memory_Allocater()
{
   wave1    = (float *)malloc(NN*sizeof(float));
   wave2    = (float *)malloc(NN*sizeof(float));
   wave1c   = (float *)malloc(NN*sizeof(float));
   wave2c   = (float *)malloc(NN*sizeof(float));
   Twave1   = (float *)malloc((NN<<1)*sizeof(float));
   Twave2   = (float *)malloc((NN<<1)*sizeof(float));
}


void Read_waveform_Files()
{
  register int i, j, k;

  char C, P_buffer[6];
  FILE *fp1;

  clrscr();
  P_buffer[5] = '\0';
  fp1 = fopen("wave1.Dat", "r");
```

```
for(;;)
{
        C = fgetc(fp1);
        if(C == 'C')
        {
          P_buffer[0] = C;
          C = fgetc(fp1);
          if(C == 'U')
          {
                  P_buffer[1] = C;
                  C = fgetc(fp1);
                  if(C == 'R')
                  {
                    P_buffer[2] = C;
                    C = fgetc(fp1);
                    if(C == 'V')
                    {
                            P_buffer[3] = C;
                            C = fgetc(fp1);
                            if(C == 'E');
                            {
                              P_buffer[4] = C;
                              break;
                            }
                    }
                  }
          }
        }
}
FOR_ALL
{
        char Tempo2[15];

        j=0;
        for(;;)
        {
                C = fgetc(fp1);
                if(C == ',' || C == EOF )
                {
                  *(wave1+i) = atof(Tempo2);
                  break;
                }
                Tempo2[j++] = C;
                Tempo2[j] = '\0';
        }
}
fclose(fp1);
P_buffer[5] = '\0';
fp1 = fopen("wave2.Dat", "r");
for(;;)
{
        C = fgetc(fp1);
        if(C == 'C')
        {
```

```
            P_buffer[0] = C;
            C = fgetc(fp1);
            if(C == 'U')
            {
                    P_buffer[1] = C;
                    C = fgetc(fp1);
                    if(C == 'R')
                    {
                      P_buffer[2] = C;
                      C = fgetc(fp1);
                      if(C == 'V')
                      {
                              P_buffer[3] = C;
                              C = fgetc(fp1);
                              if(C == 'E');
                              {
                                P_buffer[4] = C;
                                break;
                              }
                      }
                    }
            }
          }
  }
  FOR_ALL
  {
          char Tempo2[15];

          j=0;
          for(;;)
          {
                  C = fgetc(fp1);
                  if(C == ',' || C == EOF )
                  {
                    *(wave2+i) = atof(Tempo2);
                    break;
                  }
                  Tempo2[j++] = C;
                  Tempo2[j] = '\0';
          }
  }
  fclose(fp1);
}

void Calculate_Data_Time_Interval()
{
  register int i, j, k;

  char C, Buffer[6], Tempo2[15], c[9];
  FILE *fp1;

  clrscr();
  Buffer[6] = '\0';
  fp1 = fopen("wave2.Dat", "r");
```

```
    for(;;)
    {
            C = fgetc(fp1);
            if(C == 'X')
            {
              Buffer[0] = C;
              C = fgetc(fp1);
              if(C == 'I')
              {
                      Buffer[1] = C;
                      C = fgetc(fp1);
                      if(C == 'N')
                      {
                        Buffer[2] = C;
                        C = fgetc(fp1);
                        if(C == 'C')
                        {
                                Buffer[3] = C;
                                C = fgetc(fp1);
                                if(C == 'R');
                                {
                                  Buffer[4] = C;
                                  break;
                                }
                        }
                      }
              }
            }
    }
    c[9] = '\0';
    C = fgetc(fp1);
    fread(c, sizeof(c), 1, fp1);
    DataTI = atof(c);
    DeltaF = 1.0/(NN*DataTI);
    fclose(fp1);
}

void Set_Zero_Level()
{
        register int i;

        float ZeroI = *(wave1+1);
        float ZeroR = *(wave2+1);

        FOR_ALL
        {
         *(wave1c+i) = *(wave1+i) - ZeroI;
         *(wave2c+i) = *(wave2+i) - ZeroR;
        }
}

void Apply_Correction_for_truncation_Nicolson73()
{
        register i, k;
```

```
        float Inci_nn = *(wave1c+NN);
        float Refl_nn = *(wave2c+NN);

        FOR_ALL
        {
          *(wave1c+i) -= Inci_nn*(float(i-1))/float(NN-1);
          *(wave2c+i) -= Refl_nn*(float(i-1))/float(NN-1);
        }
        FOR_ALL
        {
          k = i<<1;
          *(Twave1+(k-1)) = *(wave1c+i);
          *(Twave1+k) = 0.0;
          *(Twave2+(k-1)) = *(wave2c+i);
          *(Twave2+k) = 0.0;
        }
}


void Fast_Fourier_Transform_Waveform1(int isign)
{
 register int i, j, m, y;

 float  wtemp, wr, wpr, wpi, wi, theta;
 float   tempr, tempi;
 int     n, mmax, istep;

 FOR_ALL  //....//
 {
   y = i<<1;

   *(Twave1+y) = -(*(Twave1+y));
 }
 n = NN<<1;
 j=1;
 for(i=1; i<n; i+=2)
 {
        if(j>i)
        {
          INTERCHANGE(*(Twave1+j), *(Twave1+i));
          INTERCHANGE(*(Twave1+(j+1)), *(Twave1+(i+1)));
        }
        m=n>>1;
        while(m >= 2 && j > m )
        {
          j -= m;
          m >>= 1;
        }
        j += m;
 }

 mmax = 2;
 while(n > mmax)
 {
```

```
  istep = 2*mmax;
  theta = 2.0*M_PI/(isign*mmax);
  wtemp = sin(0.5*theta);
  wpr = -2.0*wtemp*wtemp;
  wpi = sin(theta);
  wr = 1.0;
  wi = 0.0;
 for(m = 1; m < mmax; m += 2)
 {
  for(i=m; i<=n; i+=istep)
  {
          j =i+mmax;
          tempr = wr*(*(Twave1+j))-wi*(*(Twave1+(j+1)));
          tempi = wr*(*(Twave1+(j+1)))+wi*(*(Twave1+j));
          *(Twave1+j) = *(Twave1+i)-tempr;
          *(Twave1+(j+1)) = *(Twave1+(i+1))-tempi;
          *(Twave1+i)  += tempr;
          *(Twave1+(i+1)) += tempi;
  }
 wr = (wtemp=wr)*wpr-wi*wpi+wr;
 wi = wi*wpr+wtemp*wpi+wi;
 }
 mmax = istep;
 }
 FOR_ALL  //....//
 {
   y = i<<1;

   *(Twave1+y) = -(*(Twave1+y));
 }
}


void Fast_Fourier_Transform_Waveform2(int isign)
{
 register int i, j, m, y;

 float  wtemp, wr, wpr, wpi, wi, theta;
 float   tempr, tempi;
 int     n, mmax, istep;

 FOR_ALL  //....//
 {
   y = i<<1;

   *(Twave2+y) = -(*(Twave2+y));
 }
 n = NN<<1;
 j=1;
 for(i=1; i<n; i+=2)
 {
         if(j>i)
         {
           INTERCHANGE(*(Twave2+j), *(Twave2+i));
```

```
            INTERCHANGE(*(Twave2+(j+1)), *(Twave2+(i+1)));
          }
          m=n>>1;
          while(m >= 2 && j > m )
          {
            j -= m;
            m >>= 1;
          }
          j += m;
 }

 mmax = 2;
 while(n > mmax)
 {
  istep = 2*mmax;
  theta = 2.0*M_PI/(isign*mmax);
  wtemp = sin(0.5*theta);
  wpr = -2.0*wtemp*wtemp;
  wpi = sin(theta);
  wr = 1.0;
  wi = 0.0;
 for(m = 1; m < mmax; m += 2)
 {
  for(i=m; i<=n; i+=istep)
  {
          j =i+mmax;
          tempr = wr*(*(Twave2+j))-wi*(*(Twave2+(j+1)));
          tempi = wr*(*(Twave2+(j+1)))+wi*(*(Twave2+j));
          *(Twave2+j) = *(Twave2+i)-tempr;
          *(Twave2+(j+1)) = *(Twave2+(i+1))-tempi;
          *(Twave2+i)  += tempr;
          *(Twave2+(i+1)) += tempi;
  }
 wr = (wtemp=wr)*wpr-wi*wpi+wr;
 wi = wi*wpr+wtemp*wpi+wi;
 }
 mmax = istep;
 }
 FOR_ALL  //....//
 {
   y = i<<1;

   *(Twave2+y) = -(*(Twave2+y));
 }
}

void Write_Computed_S11_to_File()
{
 register int i;

 double xf, yf;
 FILE *fp1;

 fp1 = fopen("RCF.DAT", "w");
```

```
 FOR_ALL
 {
        register int k;

        k = i<<1;

   complex S11 = complex(*(Twave2+(k-1)), *(Twave2+k))/complex(*(Twave1+(k-1)), *(Twave1+k));
   real(S11) < 0   ?  fprintf(fp1,"%8.7f ", real(S11)):
                      fprintf(fp1,"%9.8f ", real(S11));
   imag(S11) < 0  ? fprintf(fp1,"%8.7f ", imag(S11)):
                     fprintf(fp1,"%9.8f ", imag(S11));
 }
 fprintf(fp1,"%s", "  ");
 fclose(fp1);
}

void Intilaize_Graphics_Mode_and_Draw_Waveforms()
{
 register int i;
 int color, gdriver = DETECT, gmode, errorcode;

 clrscr();
 normvideo();
 initgraph(&gdriver, &gmode, "");
 errorcode = graphresult();
 if (errorcode != grOk)
 {
          printf("Graphics error: %s\n", grapherrormsg(errorcode));
          printf("Press any key to halt:");
          getch();
          exit(1);
 }
 highvideo();
 setpalette(0, EGA_WHITE);
 setcolor(EGA_GREEN);
 setlinestyle(SOLID_LINE, 1,3);
 rectangle(1, 1, 638, 478);
 Draw_Original_Waveforms(); Clock(1);
 Draw_Waveforms_Corrected_for_Truncation(); Clock(1);
 Draw_Spectra_of_Waveforms(); Clock(1);
 Draw_Scattring_Cofficient_vs_Frequency(); Clock(1);
 settextstyle(1, HORIZ_DIR, 1);
 setcolor(EGA_LIGHTGRAY);
 outtextxy(203,465,"   * Press Any Key To Exit *   ");
 Clock(2);
 restorecrtmode();
 closegraph();
}

void Clock(int x)
{
  time_t Clock_start_time, Clock_end_time;
  Clock_start_time = time(NULL);
  for(;;)
```

```
    {
    Clock_end_time = time(NULL);
    if ((Clock_end_time)-(Clock_start_time) > x) break;
    }
  }

  void Draw_Original_Waveforms()
  {
   register int i;

   XF = float(X21-X11)/float(NN);
   YF = float(Y21-Y11)/7.7;
   FOR_ALL
   {
     *(wave1+i) = int(*(wave1+i)*YF);
     *(wave2+i) = int(*(wave2+i)*YF);
   }
   setcolor(EGA_BLUE);
   setlinestyle(DOTTED_LINE,1,1);
   for(i=1; i<10; i++) line(X11+i*(int(float(X21-X11)/10.0)),Y11,X11+i*(int(float(X21-X11)/10.0)),Y21);
   for(i=1; i<8 ; i++) line(X11,Y11+i*(int(float(Y21-Y11)/8.0)), X21, Y11+i*(int(float(Y21-Y11)/8.0)));
   setlinestyle(DOTTED_LINE,1,2);
   setcolor(LIGHTMAGENTA);
   line(X11+5*(int(float(X21-X11)/10.0)),Y11, X11+5*(int(float(X21-X11)/10.0)),Y21);
   line(X11,Y11+4*(int(float(Y21-Y11)/8.0)), X21, Y11+4*(int(float(Y21-Y11)/8.0)));
   setlinestyle(SOLID_LINE,1,1);
   setcolor(EGA_LIGHTBLUE);
   for(i=1;i<=NN-1;i++) line(X11+int(XF*float(i)), (Y11+4*(int(float(Y21-Y11)/8.0)))-*(wave1+i),
                        X11+int(float(i+1)*XF), (Y11+4*(int(float(Y21-Y11)/8.0)))-*(wave1+(i+1)));
   setcolor(EGA_LIGHTRED);
   for(i=1;i<=NN-1;i++) line(X11+int(XF*float(i)), (Y11+4*(int(float(Y21-Y11)/8.0)))-*(wave2+i),
                        X11+int(float(i+1)*XF), (Y11+4*(int(float(Y21-Y11)/8.0)))-*(wave2+(i+1)));
   setlinestyle(SOLID_LINE,1,1);
   setcolor(EGA_GREEN);
   setlinestyle(SOLID_LINE, 1,3);
   rectangle(X11, Y11, X21, Y21);
   outtextxy(X11-10,Y11+97,"0");
   setcolor(EGA_BLUE);
   outtextxy(X11+20,Y21+12,"Figure 1. Waveforms");
  }

  void Draw_Waveforms_Corrected_for_Truncation()
  {
   register int i;

   XF = float(X22-X12)/float(NN);
   YF = float(Y22-Y12)/7.7;
   FOR_ALL
   {
     *(wave1c+i) = int(*(wave1c+i)*YF);
     *(wave2c+i) = int(*(wave2c+i)*YF);
   }
   setcolor(EGA_BLUE);
   setlinestyle(DOTTED_LINE,1,1);
```

```
 for(i=1; i<10; i++) line(X12+i*(int(float(X22-X12)/10.0)),Y12,X12+i*(int(float(X22-X12)/10.0)),Y22);
 for(i=1; i<8 ; i++) line(X12,Y12+i*(int(float(Y22-Y12)/8.0)), X22, Y12+i*(int(float(Y22-Y12)/8.0)));
 setlinestyle(DOTTED_LINE,1,2);
 setcolor(LIGHTMAGENTA);
 line(X12+5*(int(float(X22-X12)/10.0)),Y12, X12+5*(int(float(X22-X12)/10.0)),Y22);
 line(X12,Y12+4*(int(float(Y22-Y12)/8.0)), X22, Y12+4*(int(float(Y22-Y12)/8.0)));
 setlinestyle(SOLID_LINE,1,1);
 setcolor(EGA_LIGHTBLUE);
 for(i=1; i<=NN-1; i++) line(X12+int(XF*float(i)), (Y12+4*(int(float(Y22-Y12)/8.0)))-*(wave1c+i),
                X12+int(float(i+1)*XF), (Y12+4*(int(float(Y22-Y12)/8.0)))-*(wave1c+(i+1)));
 setcolor(EGA_LIGHTRED);
 for(i=1; i<=NN-1; i++) line(X12+int(XF*float(i)), (Y12+4*(int(float(Y22-Y12)/8.0)))-*(wave2c+i),
                X12+int(float(i+1)*XF), (Y12+4*(int(float(Y22-Y12)/8.0)))-*(wave2c+(i+1)));
 setlinestyle(SOLID_LINE,1,1);
 setcolor(EGA_GREEN);
 setlinestyle(SOLID_LINE, 1,3);
 rectangle(X12, Y12, X22, Y22);
 outtextxy(X12-10,Y12+97,"0");
 setcolor(EGA_BLUE);
 outtextxy(X12-50,Y22+10,"Figure 2. Corrected for Truncation");
}

void Draw_Spectra_of_Waveforms()
{
 register int i, k;

 float *Four1, *Four2;

 XF = float(X23-X13)/float(NUM+1);
 YF = float(Y23-Y13)/14.0;
 Four1    = (float *)malloc(NN*sizeof(float));
 Four2    = (float *)malloc(NN*sizeof(float));
 FOR_ALL
 {
   k = i<<1;

   *(Four1+i) = log10(norm(complex(*(Twave1+(k-1)), *(Twave1+k))));
   *(Four2+i) = log10(norm(complex(*(Twave2+(k-1)), *(Twave2+k))));
 }
 FOR_ALL
 {
   *(Four1+i) = int(*(Four1+i)*YF);
   *(Four2+i) = int(*(Four2+i)*YF);
 }
 setcolor(EGA_BLUE);
 setlinestyle(DOTTED_LINE,1,1);
 for(i=1; i<10; i++) line(X13+i*(int(float(X23-X13)/10.0)),Y13,X13+i*(int(float(X23-X13)/10.0)),Y23);
 for(i=1; i<8 ; i++) line(X13,Y13+i*(int(float(Y23-Y13)/8.0)), X23, Y13+i*(int(float(Y23-Y13)/8.0)));
 setcolor(EGA_GREEN);
 line(X13,Y13+4*(int(float(Y23-Y13)/8.0)), X23, Y13+4*(int(float(Y23-Y13)/8.0)));
 setlinestyle(SOLID_LINE,1,1);
 setcolor(EGA_LIGHTBLUE);
 FOR_NUM line(X13+int(XF*float(i-1)), Y13+4*(int(float(Y23-Y13)/8.0)),X13+int(float(i-1)*XF),
(Y13+4*(int(float(Y23-Y13)/8.0)))-*(Four1+i));
```

```
 setcolor(EGA_LIGHTRED);
 FOR_NUM line(X13+int(XF*float(i-1)), Y23 ,X13+int(float(i-1)*XF), Y23-*(Four2+i));
 setlinestyle(SOLID_LINE,1,1);
 setcolor(EGA_GREEN);
 setlinestyle(SOLID_LINE, 1,3);
 rectangle(X13, Y13, X23, Y23);
 settextstyle(1, HORIZ_DIR, 1);
 setcolor(EGA_BLUE);
 outtextxy(X13-50,Y23+12,"Figure 3. Amplitude Spectra of Waveforms");
 setcolor(EGA_GREEN);
 outtextxy(X13-10,Y23-5,"0");
 outtextxy(X23+10,Y23-3,FHL);
 settextstyle(1, VERT_DIR, 1);
 outtextxy(X13-20,Y23-153,"log(AMPLITUDE)");
}

void Draw_Scattring_Cofficient_vs_Frequency()
{
 register int i, k;

 double xf, yf;

 float *RefCo, *Phas_Mat;

 YF = float(Y24-Y14)/8.0;
 xf = float(X24-X14)/float(NUM);
 yf = float(Y24-Y14)/(8.0*720.0);

 RefCo    = (float *)malloc(NN*sizeof(float));
 Phas_Mat = (float *)malloc(NN*sizeof(float));
 FOR_ALL
 {
   k=i<<1;

   complex S11 = complex(*(Twave2+(k-1)), *(Twave2+k))/
                                    complex(*(Twave1+(k-1)), *(Twave1+k));
   *(Phas_Mat +i) = yf*arg(S11)*180.0/M_PI;
   *(RefCo+i)     = norm(S11)*YF;
 }
 setcolor(EGA_BLUE);
 setlinestyle(DOTTED_LINE,1,1);
 for(i=1; i<10; i++) line(X14+i*(int(float(X24-X14)/10.0)),Y14,X14+i*(int(float(X24-X14)/10.0)),Y24);
 for(i=1; i<8 ; i++) line(X14,Y14+i*(int(float(Y24-Y14)/8.0)), X24, Y14+i*(int(float(Y24-Y14)/8.0)));
 setlinestyle(DOTTED_LINE,1,2);
 setcolor(LIGHTMAGENTA);
 line(X14,Y14+2*(int(float(Y24-Y14)/8.0)), X24, Y14+2*(int(float(Y24-Y14)/8.0)));
 line(X14,Y14+6*(int(float(Y24-Y14)/8.0)), X24, Y14+6*(int(float(Y24-Y14)/8.0)));
 setlinestyle(SOLID_LINE,1,2);
 setcolor(EGA_GREEN);
 FOR_NUM line(X14+int(XF*float(i-1)), (Y14+2*(int(float(Y24-Y14)/8.0)))-*(RefCo+i),
                  X14+int(XF*float(i))  , (Y14+2*(int(float(Y24-Y14)/8.0)))-*(RefCo+i+1));
 setcolor(EGA_BROWN);
 FOR_NUM line(X14+int(XF*float(i-1)), (Y14+6*(int(float(Y24-Y14)/8.0)))-*(Phas_Mat+i),
                  X14+int(XF*float(i))  , (Y14+6*(int(float(Y24-Y14)/8.0)))-*(Phas_Mat+i+1));
```

```
 setlinestyle(SOLID_LINE,1,1);
 setcolor(EGA_GREEN);
 setlinestyle(SOLID_LINE, 1,3);
 rectangle(X14, Y14, X24, Y24);
 line(X14,Y14+(Y24-Y14)/2, X24, Y14+(Y24-Y14)/2);
 settextstyle(1, HORIZ_DIR, 1);
 outtextxy(X14-20,Y14+24,"+1");
 outtextxy(X14-10,Y14+49,"0");
 outtextxy(X14-20,Y14+72,"-1");
 outtextxy(X14-10,Y24-51,"0");
 outtextxy(X24+10,Y14+97,FHL);
 setcolor(EGA_BLUE);
 outtextxy(X14-10,Y24+10,"Figure 4. S11 vs Frequency");
 setcolor(EGA_GREEN);
 settextstyle(1, VERT_DIR, 1);
 outtextxy(X14-20,Y24-183,"Magnitude");
 outtextxy(X14-20,Y24-63,"Phase");
}
```

```
/*****************************************************************************************
                                        PROG-4.CPP

This program first corrects the S11(f) data, computed by PROG-3.CPP, for unwanted reflection. It then
calculates the complex relative permittivities from the corrected S11(f) data, by solving the S11(f) equation, by
using the NEWTON-RAPHSON METHOD OF ITERATION.

                                        Tilahun M. Worku

*****************************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <complex.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <iostream.h>
#include <time.h>
#include <complex.h>

#define PER_FREE_SPACE  8.854e-12

#define NUM 100
#define NN  1024

#define CL              3e+8
#define ACCURACY    1e-10
#define FOR_ALL     for(i=1; i<=NN; i++)

double    *Freq,  *S11r_Mat,   *S11i_Mat,  *S11n_Mat, *RC, XF, YF, REAL, IMAG,
          *epsilonr,  *epsiloni, DataTI, DeltaF, *S11r_Rs, *S11i_Rs, *S11n_Rs;


void Memory_Allocator();
void Read_Rs_Data();
void Read_S11_Mat_Data();
void Calculate_Epsilon_By_Iteration();
void Correct_S11_Mat_For_Unwanted_Reflction();
void Calculate_Data_Time_Interval_And_Frequency_Values();
void Complex_Newton_Rapson(double Epsilonr, double Epsiloni, double f, int P);

main()
{
 clrscr();
 Memory_Allocator();
 Read_Rs_Data();
 Read_S11_Mat_Data();
 Calculate_Data_Time_Interval_And_Frequency_Values();
 Correct_S11_Mat_For_Unwanted_Reflction();
 Calculate_Epsilon_By_Iteration();

 return 0;
```

```
}

void Memory_Allocator()
{
  Freq      = (double *)malloc(NN*sizeof(double));
  S11r_Rs   = (double *)malloc(NN*sizeof(double));
  S11i_Rs   = (double *)malloc(NN*sizeof(double));
  S11n_Rs   = (double *)malloc(NN*sizeof(double));
  S11r_Mat  = (double *)malloc(NN*sizeof(double));
  S11i_Mat  = (double *)malloc(NN*sizeof(double));
  S11n_Mat  = (double *)malloc(NN*sizeof(double));
  epsilonr  = (double *)malloc(NN*sizeof(double));
  epsiloni  = (double *)malloc(NN*sizeof(double));
}

void Read_Rs_Data()
{
 register int i;

 char C1[11], C2[11];
 FILE *fp1;

 fp1 = fopen("RSVe.DAT", "r");
 FOR_ALL
 {
   fread(C1, sizeof(C1), 1, fp1);
   fread(C2, sizeof(C2), 1, fp1);
   *(S11r_Rs+i) = atof(C1);
   *(S11i_Rs+i) = atof(C2);
   C1[11] = '\0';
   C2[11] = '\0';
 }
 fclose(fp1);
}


void Read_S11_Mat_Data()
{
 register int i;

 char C0[11], C1[11], C2[11];
 FILE *fp1;

 fp1 = fopen("RCF.DAT", "r");
 FOR_ALL
 {
   fread(C1, sizeof(C1), 1, fp1);
   fread(C2, sizeof(C2), 1, fp1);
   *(S11r_Mat+i) = atof(C1);
   *(S11i_Mat+i) = atof(C2);
   *(S11n_Mat+i) = norm(complex(atof(C1),atof(C2)));
 }
}
```

```
void Correct_S11_Mat_For_Unwanted_Reflction()
{
 register int i;

 FOR_ALL
 {
  double w;
  complex J = complex(0.0, 1.0);
  w         = 2.0*M_PI*(*(Freq+i))*1e+9;                  // w
  complex U  = J*w/CL;                                    // Jw/c
  complex c0 = -2.0*U*L;                                  // -2τL
  complex c1 = exp(c0);                                   // e^(-2τL)
  complex c2 = complex(*(S11r_Rs+i),*(S11i_Rs+i));        // Rs
  complex c3 = complex(*(S11r_Mat+i),*(S11i_Mat+i)); // Rm
  complex c4 = c2*c3;                                     // RsRm
  complex c5 = c4*c1;                                     // RsRme^(-2τL)
  complex c6 = c3*c1;                                     // Rme^(-2τL)
  complex c7 = c6-c5;                                     // Rme^(-2τL)-RsRme^(-2τL)
  complex c8 = 1.0-c5;                                    // 1-RsRme^(-2τL)
  complex c9 = c7/c8;
  *(S11r_Mat+i) = real(c9);
  *(S11i_Mat+i) = imag(c9);
  *(S11n_Mat+i) = norm(c9);

 }
}

void Calculate_Data_Time_Interval_And_Frequency_Values()
{
 register int i, j, k;

 char C, Buffer[6], Tempo2[15], c[9];
 FILE *fp1;

 clrscr();
 Buffer[6] = '\0';
 fp1 = fopen("wave1.Dat", "r");
 for(;;)
 {
        C = fgetc(fp1);
        if(C == 'X')
        {
          Buffer[0] = C;
          C = fgetc(fp1);
          if(C == 'I')
          {
                  Buffer[1] = C;
                  C = fgetc(fp1);
                  if(C == 'N')
                  {
                    Buffer[2] = C;
                    C = fgetc(fp1);
                    if(C == 'C')
```

```
                    {
                        Buffer[3] = C;
                        C = fgetc(fp1);
                        if(C == 'R');
                        {
                          Buffer[4] = C;
                          break;
                        }
                    }
                  }
              }
            }
  }
  c[9] = '\0';
  C = fgetc(fp1);
  fread(c, sizeof(c), 1, fp1);
  DataTI = atof(c);
  DeltaF = 1.0/(NN*DataTI);
  fclose(fp1);
  FOR_ALL
  {
        if(i<=((NN/2)+1))
          *(Freq+i) = (i-1)*DeltaF*1e-9;
        else
          *(Freq+i) = (i-1-NN)*DeltaF*1e-9;
  }
}


void Calculate_Epsilon_By_Iteration()
{
 register int i;

 *(epsilonr+1) = epsilon_s;
 *(epsiloni+1) = 0.0;
 for(i=2; i<=NUM+1; i++)
 {
        if(i==1)
          Complex_Newton_Rapson(epsilon_s, 0.0, (*(Freq+i)), i);
        else
          Complex_Newton_Rapson(*(epsilonr+i-1),*(epsiloni+i-1),(*(Freq+i)), i);
        *(epsilonr+i) = REAL;
        *(epsiloni+i) = IMAG;
 }
}

void Complex_Newton_Rapson(double Epsilonr, double Epsiloni, double f, int i)
{
  double R;
  char C;

  complex epsilon = complex(Epsilonr, Epsiloni);
  complex S11Mat  = complex(*(S11r_Mat+i), *(S11i_Mat+i));
```

```
    for(;;)
    {
        if(kbhit() != 0)
        {
          C = getch();
          if(C=='x') break;
        }
        R = -20.0*M_PI*f*L/3.0;          // R   = 2πfL/c
        complex c1  = sqrt(epsilon);     // c1  = √ε
        complex c2  = complex(0.0, R);   // c2  = iR
        complex c3  = c1*c2;             // c3  = iR√ε
        complex c4  = 2.0*c3;            // c4  = 2iR√ε
        complex c5  = 4.0*c3;            // c5  = 4iR√ε
        complex c6  = exp(c4);           // c6  = e^(2iR√ε)
        complex c7  = exp(c5);           // c7  = e^(4iR√ε)
        complex c8  = z*c1;              // c8  = z√ε
        complex c9  = (1.0-c8)/(1.0+c8); // c9  = (1-z√ε)/(1+z√ε)
        complex c10 = c6*c9;             // c10 = ((1-z√ε)/(1+z√ε))e^(2iR√ε)
        complex S11c= (c6+c9)/(1.0+c10);// S11c= (e^(2iR√ε)+(1-z√ε)/(1+z√ε))/(1+((1-z√ε)/(1+z√ε))e^(2iR√ε))
        complex S11 = S11c - S11Mat;
                                         // S11 = (e^(2iR√ε)+(1-z√ε)/(1+z√ε)) /(1+((1-z√ε)/(1+z√ε))
                                                e^(2iR√ε) )- S11Mat
        complex c11 = c5*c6;             // c11 = 4iR√ε*e^(2iR√ε)
        complex c12 = z*(-1.0+c7+c11);  // c12 = z*(-1+e^(2iR√ε)+4iR√ε*e^(2iR√ε))
        complex c13 = c6*c8;             // c13 = z√ε*e^(2iR√ε)
        complex c14 = (-1.0-c6-c8+c13); // c14 = (-1-e^(2iR√ε)-z√ε+z√ε*e^(2iR√ε))
        complex c15 = c14*c14;           // c15 = (-1-e^(2iR√ε)-z√ε+z√ε*e^(2iR√ε))^2
        complex c16 = c15*c1;            // c16 = √ε*(-1-e^(2iR√ε)-z√ε+z√ε*e^(2iR√ε))^2
        complex DS11Deps = c12/c16;
        complex epsr = epsilon - (S11/DS11Deps);
        if((fabs(norm(epsr)-norm(epsilon)) ) < ACCURACY &&
                fabs(arg(epsilon)-arg(epsr))*180.0/M_PI < ACCURACY)
        {
                cout<<i<<"\t"<<-imag(epsr)<<"\n";
                fprintf(fp1,"%9f\n",-imag(epsr));
                if((i%24) == 0) getch();
                REAL = real(epsr);
                IMAG = imag(epsr);
                break;
        }
        else epsilon = epsr;
       }
}
```

```
/*****************************************************************************

                                PROG-5.CPP

This program calculates correction factores  for unwanted reflection by using the Giese and Tiemann
(1975) correction procedure.     The correction factores computed by this program are saved in a file
called  RSV.DAT  and  are  used  in PROG-4.CPP to correct the measured S11(f) values for unwated
reflections.

                              Tilahun M. Worku

*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <complex.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <iostream.h>
#include <time.h>
#include <complex.h>

#define NUM 75
#define NN  1024

#define PER_FREE_SPACE  8.854e-12


#define  X11       195
#define  Y11       10
#define  X21       445
#define  Y21       270
#define  FOR_ALL   for(i=1; i<=NN; i++)

#define  CL  3e+8



double  far *Freq, far *S11r_Mat, far *S11i_Mat, far *S11n_Mat, far *S11r_Ref,
        far *S11i_Ref, far *S11n_Ref, DataTI, DeltaF,  far *S11r_Rs, far *S11i_Rs,
        far *S11n_Rs;

double  epsilon_s = 24.4, z = 0.332, epsilon_inf = 4.8, f_relax = 1.14,
        Beta = 0.0, L = 0.041, R, b1 = 1.0 - Beta, Sigma_dc = 0.0;

void    Memory_Allocator();
void    Calculate_Data_Time_Interval_And_Frequency_Values();
void    Read_S11_Data();
void    Calculate_S11_of_Ref_material();
void    Calculate_Correction_For_Unwanted_Reflction();
```

```
main()
{
 register int i;

 clrscr();
 Memory_Allocator();
 Calculate_Data_Time_Interval_And_Frequency_Values();
 Read_S11_Data();
 Calculate_S11_of_Ref_material();
 Calculate_Correction_For_Unwanted_Reflction();

 return 0;
}

void Memory_Allocator()
{
  Freq      = (double *)malloc(NN*sizeof(double));
  S11r_Rs   = (double *)malloc(NN*sizeof(double));
  S11i_Rs   = (double *)malloc(NN*sizeof(double));
  S11n_Rs   = (double *)malloc(NN*sizeof(double));
  S11r_Mat  = (double *)malloc(NN*sizeof(double));
  S11i_Mat  = (double *)malloc(NN*sizeof(double));
  S11n_Mat  = (double *)malloc(NN*sizeof(double));
  S11r_Ref  = (double *)malloc(NN*sizeof(double));
  S11i_Ref  = (double *)malloc(NN*sizeof(double));
  S11n_Ref  = (double *)malloc(NN*sizeof(double));
}

void Calculate_Data_Time_Interval_And_Frequency_Values()
{
 register int i, j, k;

  char C, Buffer[6], Tempo2[15], c[9];
  FILE *fp1;

  clrscr();
  Buffer[6] = '\0';
  fp1 = fopen("wave1.Dat", "r");
  for(;;)
  {
        C = fgetc(fp1);
        if(C == 'X')
        {
          Buffer[0] = C;
          C = fgetc(fp1);
          if(C == 'I')
          {
                Buffer[1] = C;
                C = fgetc(fp1);
                if(C == 'N')
                {
                  Buffer[2] = C;
                  C = fgetc(fp1);
```

```
                    if(C == 'C')
                    {
                        Buffer[3] = C;
                        C = fgetc(fp1);
                        if(C == 'R');
                        {
                          Buffer[4] = C;
                          break;
                        }
                    }
                  }
              }
          }
  }
  c[9] = '\0';
  C = fgetc(fp1);
  fread(c, sizeof(c), 1, fp1);
  DataTI = atof(c);
  DeltaF = 1.0/(NN*DataTI);
  fclose(fp1);
  FOR_ALL
  {
        if(i<=((NN/2)+1))
          *(Freq+i) = (i-1)*DeltaF*1e-9;
        else
          *(Freq+i) = (i-1-NN)*DeltaF*1e-9;
  }
}

void Read_S11_Data()
{
 register int i;

 char C0[11], C1[11], C2[11];

 FILE *fp1;
 complex J = complex(0.0, 1.0);
 fp1 = fopen("RCF.DAT", "r");
 FOR_ALL
 {
  fread(C1, sizeof(C1), 1, fp1);
  fread(C2, sizeof(C2), 1, fp1);
  complex C3 = complex(atof(C1), atof(C2));
  *(S11r_Mat+i) = real(C3);
  *(S11i_Mat+i) = imag(C3);
  *(S11n_Mat+i) = norm(C3);
 }
}

void  Calculate_S11_of_Ref_material()
{
  register int i, j, k;

  double far *Epsilonr, far *Epsiloni;
```

```
complex J = complex(0.0, 1.0);
Epsilonr  = (double *)malloc(NN*sizeof(double));
Epsiloni  = (double *)malloc(NN*sizeof(double));
FOR_ALL
{
 complex a1  =  J;                    // a1  =  j
 complex a2  =  epsilon_inf;          // a2  =  ε∞
 complex a3  =  epsilon_s;            // a3  =  εs
 complex a4  =  *(Freq+i)*a1;         // a4  =  fj
 complex a5  =  a4/f_relax;           // a5  =  fj/frel
 complex a6  =  pow(a5, b1);          // a6  =  (fj/frel)^(1-ß)
 complex a7  =  1.0 + a6;             // a7  =  1 + (fj/frel)^(1-ß)
 complex a8  =  a3 - a2;              // a8  =  (εs - ε∞)
 complex a9  =  a8/a7;                // a9  =  (εs - ε∞)/(1 + (fj/frel)^(1-ß))
 complex a10 =  a2 + a9;              // a10 =  ε∞ + (εs - ε∞)/(1 + (jf/frel)^(1-ß))
 if(i == 1)
 {
       complex a15  =  a10;           // a15 =  [ε∞ + (εs - ε∞)/(1 + (fj/frel)^(1-ß))]
       *(Epsilonr+i) = real(a15);
       *(Epsiloni+i) = imag(a15);
 }
 else
 {
 complex a11 =  a1*Sigma_dc;          // a11 = jσdc
 complex a12 =  2.0*M_PI*(*(Freq+i)); // a12 = 2πf
 complex a13 =  a12*PER_FREE_SPACE;   // a13 = 2πfεo
 complex a14 =  a11/(a13*1e+9);       // a14 = jσdc/2πfεo
 complex a15 =  a10 - a14;            // a15 = [ε∞ + (εs - ε∞)/(1 + (fj/frel)^(1-ß))] - iσdc/2πfεo
                                      // THIS IS COLE-COLE EQUATION
 *(Epsilonr+i) = real(a15);
 *(Epsiloni+i) = imag(a15);
}
}
FOR_ALL
{
       R = -20.0*M_PI*(*(Freq+i))*L/3.0;     // R   = -2πfL/c
       complex c0  = complex(*(Epsilonr+i),
                             *(Epsiloni+i)); // c0  = ε
       complex c1  = sqrt(c0);               // c1  = √ε
       complex c2  = R*J;                    // c2  = Rj
       complex c4  = c2*c1;                  // c4  = jR√ε
       complex c5  = 2.0*c4;                 // c5  = 2iR√ε
       complex c6  = exp(c5);                // c6  = e^(2jR√ε)
       complex c7  = complex(z,0.0);         // c7  = z
       complex c8  = c7*c1;                  // c8  = z√ε
       complex c9  = complex(1.0, 0.0);      // c9  = 1.0
       complex c10 = ((c9 - c8)/(c9 + c8));  // c10 = (1-z√ε)/(1+z√ε)
       complex c11 = c6*c10;                 // c11 = e^(2jR√ε)((1-z√ε)/(1+z√ε))
       complex c12 = (c10 + c6)/(c9 + c11);  // c12 = (((1-z√ε)/(1+z√ε)) + e^(2jR√ε))
                                             //       /(1+e^(2jR√ε)((1-z√ε)/(1+z√ε)))
       *(S11r_Ref+i) = real(c12);
       *(S11i_Ref+i) = imag(c12);
       *(S11n_Ref+i) = norm(c12);
```

```
  }
 }

 void Calculate_Correction_For_Unwanted_Reflction()
 {
  register int i;

  FILE *fp1;
  complex J = complex(0.0, 1.0);
  FOR_ALL
  {
   double R, w;

   w          =  2.0*M_PI*(*(Freq+i))*1e+9;                    // w
   complex U   =  J*w/CL;                                      // Jw/c
   complex c0  =  2.0*U*L;                                     // 2τL
   complex c1  =  exp(c0);                                     // e^(2τL)
   complex c2  =  complex(*(S11r_Ref+i),*(S11i_Ref+i));        // S11c
   complex c3  =  complex(*(S11r_Mat+i),*(S11i_Mat+i));        // S11m
   complex c4  =  c3*c1;                                       // S11ce^(2τL)
   complex c5  =  c4-c3;                                       // S11ce^(2τL) - S11m
   complex c6  =  c2*c3;                                       // S11cS11m
   complex c8  =  c6-c3;                                       // S11cS11m-S11m
   if(real(c8) == 0.0 && imag(c8)==0.0)
   {
    *(S11r_Rs+i) = 1.0;
    *(S11i_Rs+i) = 0.0;
    *(S11n_Rs+i) = 0.0;
   }
   else
   {
    complex c9   = c5/c8;                                      // [S11ce^(2τL) - S11m] / [S11cS11m-S11m]
    *(S11r_Rs+i) = real(c9);
    *(S11i_Rs+i) = imag(c9);
    *(S11n_Rs+i) = norm(c9);
   }
  }
  fp1 = fopen("RSV.DAT", "w");
  FOR_ALL
  {
          if(*(S11r_Rs+i) < 0)
          {
           (int(*(S11r_Rs+i)) < -9) ?  fprintf(fp1,"%8.6f ", *(S11r_Rs+i)):
                                       fprintf(fp1,"%8.7f ", *(S11r_Rs+i));
          }
          else
          {
           (int(*(S11r_Rs+i)) > 9)  ?  fprintf(fp1,"%9.7f ", *(S11r_Rs+i)):
                                       fprintf(fp1,"%9.8f ", *(S11r_Rs+i));
          }

          if(*(S11i_Rs+i) < 0)
          {
```

```
            (int(*(S11i_Rs+i)) < -9) ?  fprintf(fp1,"%8.6f ", *(S11i_Rs+i)):
                                        fprintf(fp1,"%8.7f ", *(S11i_Rs+i));
          }
          else
          {
            (int(*(S11i_Rs+i)) > 9)  ?  fprintf(fp1,"%9.7f ", *(S11i_Rs+i)):
                                        fprintf(fp1,"%9.8f ", *(S11i_Rs+i));
          }
 }
 fprintf(fp1,"%s", "  ");
 fclose(fp1);
}
```