Permittivity profile reconstructions using transient electromagnetic reflection data

Fuks, Peter; Kristensson, Gerhard; Larson, Gunnar

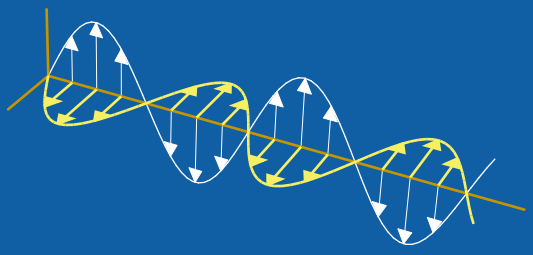
1990

CODEN:LUTEDX/(TEAT-7009)/1-88/(1990)

Permittivity profile reconstructions using transient electromagnetic reflection data

Peter Fuks, Gerhard Kristensson, and Gunnar Larson

Department of Electroscience
Electromagnetic Theory
Lund Institute of Technology
Sweden



Peter Fuks and Gunnar Larson

Department of Electromagnetic Theory
Royal Institute of Technology
SE-100 44 Stockholm
Sweden

Gerhard Kristensson

Department of Electromagnetic Theory
Lund Institute of Technology
P.O. Box 118
SE-221 00 Lund
Sweden

Editor: Gerhard Kristensson

Abstract

This paper is concerned with the permittivity reconstruction of inhomogeneous dielectric media. The method applies to profiles that vary with depth only, i.e. it provides a one-dimensional profile reconstruction. The data are collected and analyzed in the time domain. In the first part of the paper the theory of the method is reviewed. It is showed that a finite time trace of reflection data suffices to uniquely reconstruct the permittivity profile of the medium. The latter part of the paper presents the experimental set-up and contains also a thorough discussion of the errors that affect the measurements. The inverse scattering algorithm that is used is either based upon an imbedding procedure or on a Green functions approach. The input to either of these algorithms is the reflection kernel or the impulse response of the medium, i.e. the delta function response of the medium. Therefore, a deconvolution of the the measured reflected field and the incident field must be performed. This deconvolution problem is also addressed briefly in this paper.

1 Introduction

In this paper an experimental implementation of two inversion algorithms for inhomogeneous dielectric media is presented. The first algorithm is based upon an imbedding procedure which was suggested originally by Corones, Davison and Krueger [2]. The second algorithm utilizes the Green functions of the problem and this approach was first introduced by Krueger and Ochs [7]. These algorithms and the underlying theory are presented in Section 2.

The permittivity profile is assumed to vary only with depth and the variation is assumed to be continuous. Otherwise, it is arbitrary. Thus, no assumptions are made about a piecewise constant permittivity profile. The assumption on a continuous permittivity profile can be relaxed, and extensions of the theory so that finite jump discontinuities in the permittivity profile can be treated are possible, see [5]– [6]. These, extensions are, however, not presented in this paper. Furthermore, it is assumed that the medium is dispersion free and lossless. It is, however, possible to cope with lossy and dispersive profiles, but this requires an extension of the theory presented in Section 2. The theory of lossy and dispersive media are presented in [3]– [6] and [1], respectively.

To meet the assumptions made above and to have an easily controlled environment for the experiment a coaxial component set-up is adopted. The one-dimensional variation in the permittivity is then easy to realize and the dispersion free wave propagation in the wave guide is guaranteed by the TEM wave mode.

One of the prominent advantages with the theory and the algorithms used in this paper is that they only require a finite time trace of reflection data as input data for the inversion of the permittivity profile. More specifically, one round trip of reflection data is needed, where one round trip is defined as the time it takes for the pulse to propagate through the medium and back again. The design of the experiment can therefore be made so that all unwanted reflections from the

experimental set-up arrive after one round trip. In this way these reflections do not affect the measurements and it is possible to avoid some unwanted interference with the experimental set-up. This is not possible with an experimental set-up that measures the fixed frequency response of the dielectric sample.

Some general considerations are found in Section 3. In Section 4 the deconvolution problem encountered in this paper is described and analyzed. The experimental set-up is described in Section 5 and a detailed presentation of the error analysis is found in Section 6. Finally, in Section 7 the results of this paper are presented. An appendix contains the computer code used in this paper.

2 Theory

In this section the mathematical model is presented, some notations are introduced and the precise statement of the inverse problem is given.

2.1 Mathematical model

An inhomogeneous slab occupies the region $0 \leq z \leq L$. The slab is assumed to be lossless and is modeled by a relative permittivity $\epsilon(z)$ that varies with depth z. A homogeneous lossless medium is situated on either side of the slab. These homogeneous media have a constant permittivity ϵ_1 and ϵ_2, respectively, see Figure 1.

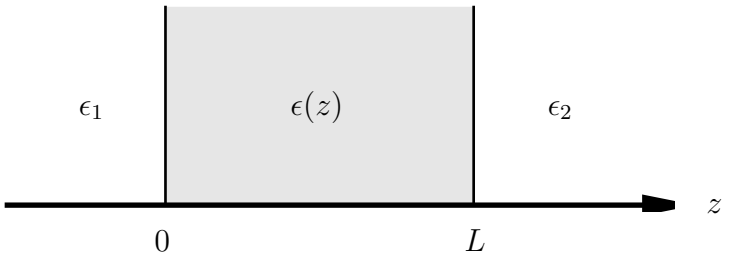


Figure 1: The geometry of the slab.

To simplify the theoretical analysis in this section it is assumed that the permittivity profile is continuous at the ends, $z = 0$ and $z = L$, respectively, and, furthermore, continuously differentiable inside the slab, $0 < z < L$. Typical plot of the permittivity profile is shown in Figure 2. The assumption of continuity at the left and right ends of the slab is no loss of generality, see e.g [5], and for the purpose of this paper, the present assumptions are quite sufficient. More explicitly, reflection and transmission data for a permittivity profile with finite jump discontinuities at the edges can always be transformed to reflection and transmission data for a permittivity profile where these jumps have been removed, i.e. the permittivity profile is continuous everywhere. These transformations consist of solving Volterra

integral equations of the second kind. These equations are well-posed and numerically efficient methods to solve these integral equations are easy to find. If the jump discontinuities are inside the slab similar methods apply.

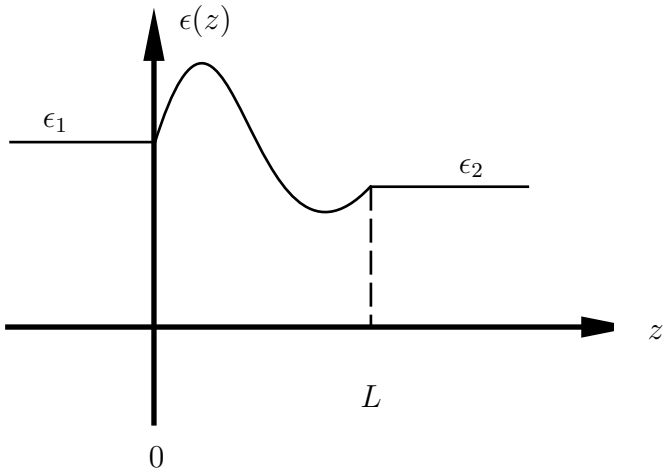


Figure 2: The permittivity profile $\epsilon(z)$ as a function of depth z.

The profile is excited by an electromagnetic wave impinging normally on the slab from the left. Assuming that all fields only depend spatially on the depth z, the Maxwell equations in the absence of free charges and currents imply (the fields E and D are assumed to have components along the x-axis and B and H along the y-axis)

$$\begin{cases} \partial_z E(z,t) = -\partial_t B(z,t) \\ \partial_z H(z,t) = -\partial_t D(z,t). \end{cases} \tag{2.1}$$

Here ∂_z and ∂_t denotes (partial) differentiation with respect to depth z and time t, respectively. The constitutive relations

$$\begin{cases} D(z,t) = \epsilon_0 \epsilon(z) E(z,t) \\ H(z,t) = B(z,t)/\mu_0, \end{cases}$$

and (2.1) then imply that the electric field $E(z,t)$ inside the slab satisfies the wave equation

$$\partial_z^2 E(z,t) - c^{-2}(z)\partial_t^2 E(z,t) = 0, \tag{2.2}$$

where the local phase velocity $c(z) = \{\epsilon_0 \epsilon(z) \mu_0\}^{-\frac{1}{2}}$.

2.2 Scattering representation and wave splitting

To the left of the slab, $z < 0$, the electric field is a sum of two parts, one right going incident wave, $E^i(t)$, and one left going reflected wave, $E^r(t)$. Similarly, to the right

of the slab, $z > L$, the electric field consists of one right going transmitted wave, $E^t(t)$. The total field $E(z,t)$ outside the slab is therefore

$$E(z,t) = \begin{cases} E^i(t - z/c(0)) + E^r(t + z/c(0)), & z < 0 \\ E^t(t - \ell - (z-L)/c(L)), & z > L, \end{cases}$$

where $c(0) = \{\epsilon_0\epsilon_1\mu_0\}^{-\frac{1}{2}}$ and $c(L) = \{\epsilon_0\epsilon_2\mu_0\}^{-\frac{1}{2}}$ are the phase velocities on the left and right hand side of the slab, respectively (remember the continuity in the permittivity profile at the ends). The quantity $\ell = \int_0^L \sqrt{\epsilon_0\epsilon(z)\mu_0}\, dz$ is the time it takes for the wave front to go through the slab from $z = 0$ to $z = L$.

The incident and the scattered fields, respectively, are related by scattering operators. These relations are integral operators represented by

$$\begin{aligned} E^r(t) &= \int_{-\infty}^{t} R^+(t-t')E^i(t')\, dt' \\ E^t(t) &= \sqrt{\frac{c_2}{c_1}}\left\{E^i(t) + \int_{-\infty}^{t} T^+(t-t')E^i(t')\, dt'\right\}, \end{aligned} \tag{2.3}$$

where the kernels $R^+(t)$ and $T^+(t)$ are the reflection and the transmission kernels of the slab, respectively, for an incident wave from the left. These kernels are independent of how the slab is excited, i.e. totally determined by the $\epsilon(z)$ profile. Notice that if $E^i(t) = \delta(t)$ (where δ is the Dirac delta function) then it follows that $E^r(t) = R^+(t)$ and $E^t(t) = \sqrt{\frac{c_2}{c_1}}\{\delta(t) + T^+(t-t')\}$. Hence, the scattering kernels R^+ and T^+ are the impulse responses of the medium.

One of the key stones in the theory of this paper is the wave splitting transformation. This is a transformation of dependent variables from the pair $\{E, \partial_z E\}$ to another pair $\{E^+, E^-\}$ defined by

$$E^\pm(z,t) = \frac{1}{2}\left\{E(z,t) \mp c(z)\int_{-\infty}^{t} \partial_z E(z,t')\, dt'\right\}.$$

This wave splitting transformation can be written in a matrix shorthand notation as

$$\begin{pmatrix} E^+ \\ E^- \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & -c\partial_t^{-1} \\ 1 & c\partial_t^{-1} \end{pmatrix}\begin{pmatrix} E \\ \partial_z E \end{pmatrix} = T\begin{pmatrix} E^+ \\ E^- \end{pmatrix}. \tag{2.4}$$

The operator T has a formal inverse

$$T^{-1} = \begin{pmatrix} 1 & 1 \\ -c^{-1}\partial_t & c^{-1}\partial_t \end{pmatrix},$$

that will be used below.

In a region where the phase velocity c is constant this wave splitting transformation has the effect of projecting out the left and the right going parts of the field. More explicitly, in a region where the phase velocity c is constant the general solution to (2.2) is

$$E(z,t) = f(t - z/c) + g(t + z/c),$$

where f and g are arbitrary functions. It is then easy to calculate the fields $E^+(z,t)$ and $E^-(z,t)$ defined in (2.4). They are

$$\begin{cases} E^+(z,t) = f(t - z/c) \\ E^-(z,t) = g(t + z/c). \end{cases}$$

In a region where c is not constant the transformation defined in (2.4) is still well-defined. In this case the fields $E^+(z,t)$ and $E^-(z,t)$ are *defined* as the left and the right going parts of the field, respectively, even though no interpretation such as in the constant phase velocity case above can be made. Notice that

$$E(z,t) = E^+(z,t) + E^-(z,t), \tag{2.5}$$

for all profiles.

The fields $E^+(z,t)$ and $E^-(z,t)$ satisfy the following partial differential equation

$$\partial_z \begin{pmatrix} E^+ \\ E^- \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} E^+ \\ E^- \end{pmatrix}, \tag{2.6}$$

where

$$\begin{cases} \alpha = -c^{-1}\partial_t + \frac{c'}{2c} \\ \beta = -\frac{c'}{2c} \\ \gamma = -\frac{c'}{2c} \\ \delta = c^{-1}\partial_t + \frac{c'}{2c}. \end{cases}$$

This can most easily be seen by combining (2.2) and (2.4). This equation is equivalent to the wave equation, (2.2), and gives the dynamics of the fields $E^+(z,t)$ and $E^-(z,t)$.

2.3 Invariant imbedding

Consider now a subsection $[z, L]$ of the region $[0, L]$, see Figure 3. Mathematically, the original problem, $[0, L]$, is imbedded in a family of problems where the left edge of the slab, z, is the parameter that is varied.

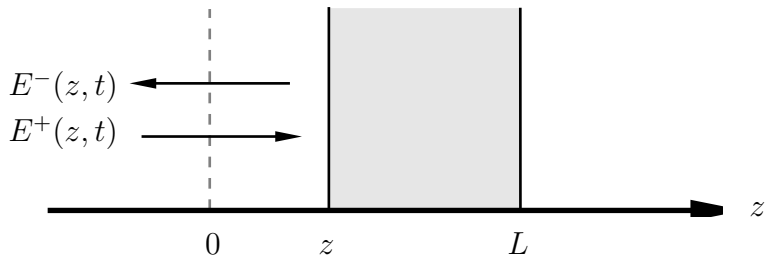


Figure 3: The geometry of the subsection problem $[z, L]$.

The fields $E^+(z,t)$ and $E^-(z,t)$, defined at the position z, are related to each other in a similar way as the incident field $E^i(t)$ and the reflected field $E^r(t)$ in (2.3) of the original physical problem are related to each other. This relation is represented as an integral operator as

$$E^-(z,t) = \int_{-\infty}^{t} R^+(z,t-t')E^+(z,t')\,dt'. \tag{2.7}$$

The kernel $R^+(z,t)$ can be interpreted as the reflection kernel for the subsection $[z,L]$, where the medium to the left of z is of constant permittivity $\epsilon(z)$. The field $E^+(z,t)$ serves as an incident field, while $E^-(z,t)$ is a reflected field, for this subsection problem. For the special value $z=0$, the physical reflection kernel $R^+(t)$ in (2.3) is identical to $R^+(0,t)$. Hence, the reflection kernel $R^+(t)$ for the physical region $[0,L]$ is imbedded in a family of subsection problems $[z,L]$ with reflection kernels $R^+(z,t)$.

The dynamics, (2.6), and the relation between the fields $E^+(z,t)$ and $E^-(z,t)$, given by (2.7), imply that the reflection kernel $R^+(z,t)$ satisfies a non-linear differential equation. Lengthy, but straightforward, calculations show that

$$\partial_z R^+(z,t) - \frac{2}{c(z)}\partial_t R^+(z,t) = \frac{c'(z)}{2c(z)}\int_0^t R^+(z,t-t')R^+(z,t')\,dt' \tag{2.8}$$

$$R^+(z,0) = \frac{1}{4}c'(z) \tag{2.9}$$

$$R^+(L,t) = 0. \tag{2.10}$$

It is intuitively clear that the dependence of the reflection kernel $R^+(z,t)$ on z is related to the local properties of the slab at z. This is expressed mathematically in (2.9). Equation (2.10) implies that the reflection kernel is zero at $z=L$, i.e. no scatterer present. Notice that (2.8) is non-linear due to the convolution integral on the right hand side of the equation, and that (2.8) has a directional derivative in the $(z,-\frac{2}{c(z)})$ direction. This latter property solves the inverse problem, which now can be stated more explicitly.

The inverse problem solved in this paper is the reconstruction of the permittivity profile $\epsilon(z)$ from reflection data. Specifically, given reflection data for one round trip, i.e. $R^+(t)$, for $0 \le t \le 2\ell$, find the permittivity profile $\epsilon(z)$, $0 \le z \le L$, see Table 1. The numerical algorithm based upon the imbedding equation (2.8) is now presented.

Problem	**Known**	**Sought**
Direct	$\epsilon(z)$, $0 \le z \le L$	$R^+(t)$, $0 \le t \le 2\ell$
Inverse	$R^+(t)$, $0 \le t \le 2\ell$	$\epsilon(z)$, $0 \le z \le L$

Table 1.

In order to make the numerical computations more easy a normalized travel time coordinate transformation is made. The transformation is

$$\begin{aligned} x &= x(z) = \int_0^z \frac{dz'}{\ell c(z')} \\ s &= t/\ell \\ R(x,s) &= \ell R^+(z,t), \end{aligned} \tag{2.11}$$

where $\ell = \int_0^L \sqrt{\epsilon_0 \epsilon(z) \mu_0}\, dz$ is the time it takes for the wave front to go through the slab. This transformation maps the slab $z \in [0, L]$ to the interval $x \in [0, 1]$, and the time s is normalized so that $s = 1$ is the time it takes for the wave front to go through the slab. The new scaled reflection kernel $R(x, s)$ satisfies

$$\partial_x R(x,s) - 2\partial_s R(x,s) = -\frac{1}{2} A(x) \int_0^s R(x, s-s') R(x,s')\, ds' \tag{2.12}$$

$$R(x,0) = -\frac{1}{4} A(x) \tag{2.13}$$

$$R(1,s) = 0, \tag{2.14}$$

where

$$A(x) = -\ell c'(z(x)) = -\frac{d}{dx} \ln c(z(x)), \quad 0 < x < 1. \tag{2.15}$$

The inverse transformations of (2.11) and (2.15) are

$$z(x) = c(0)\ell \int_0^x \left\{ \exp\left\{ -\int_0^{x'} A(x'')\, dx'' \right\} dx' \right\}, \quad 0 < x < 1 \tag{2.16}$$

$$\epsilon(z(x)) = \epsilon_1 \exp\left\{ 2 \int_0^x A(x')\, dx' \right\}, \quad 0 < x < 1. \tag{2.17}$$

Thus, the knowledge of $A(x)$, $0 \le x \le 1$ and the constants ℓ and ϵ_1 determine z, L and $\epsilon(z)$, $0 \le z \le L$. How $A(x)$, $0 \le x \le 1$ can be reconstructed numerically from (2.12) and (2.13) is now presented.

The numerical implementation of the imbedding equation (2.12) is most easily done by writing (2.12) as

$$\partial_x R(x, s-2x) = -\frac{1}{2} A(x) \int_0^{s-2x} R(x, s-2x-s') R(x,s')\, ds',$$

and integrate from $x - h$ to x and let then time be $s + 2x$. The result is

$$R(x,s) - R(x-h, s+2h) = -\frac{1}{2} \int_{x-h}^x A(x') (R * R)(x', s + 2(x - x'))\, dx', \tag{2.18}$$

where star $*$ denotes time convolution, i.e.

$$(R * R)(x,s) = \int_0^s R(x, s-s') R(x,s')\, ds'.$$

Introduce a uniform grid of points (x_i, s_j) in (x, s) space, where $x_i = ih$, $s_j = 2jh$, $i = 0, 1, 2, \ldots, N$, $j = 0, 1, 2, \ldots, N - i$, $h = 1/N$ and N is an integer. Denote

$$\begin{aligned} R_{i,j} &= R(x_i, s_j) \\ A_i &= A(x_i). \end{aligned}$$

With the use of the trapezoidal rule in (2.18) the following algorithm is easily found

$$\begin{aligned} R_{i,j} \;=\; & \left\{ R_{i-1,j+1} - \frac{h^2}{2} \left\{ A_i \sum_{k=1}^{j-1} R_{i,j-k} R_{i,k} \right.\right. \\ & \left.\left. + A_{i-1} \sum_{k=1}^{j+1} R_{i-1,j+1-k} R_{i-1,k} \right\} \right\} \left\{ 1 - \frac{h^2}{8} A_i^2 \right\}^{-1}, \end{aligned} \tag{2.19}$$

where $i = 1, 2, \ldots, N$, $j = 0, 1, 2, \ldots, N - i$, and where the values of A_i are determined from (take $s = 0$ in (2.18) and use (2.13))

$$A_i = -4R_{i-1,1} \left\{ 1 + \frac{h^2}{8} A_{i-1}^2 \right\}. \tag{2.20}$$

The error made in (2.19) and (2.20) is of order $\mathcal{O}(h^3)$.

The input data of the numerical algorithm is

$$\begin{cases} R^+(2j\ell h), \quad i = 0, 1, 2, \ldots, N \\ \ell \\ \epsilon_1. \end{cases} \tag{2.21}$$

The reconstruction of the permittivity profile goes in two steps. First reconstruct $A(x)$ from the reflection data R^+ data (the constant ℓ is also needed). The from $A(x)$ and the constants ℓ and ϵ_1 determine $z(x)$ and $\epsilon(z(x))$. This is done from (2.16) and (2.17). The length of the slab is also determined by $L = z(1)$.

The initialization of the numerical algorithm is made by assigning

$$R_{0,j} = R(0, 2jh) = \ell R^+(0, 2j\ell h) = \ell R^+(2j\ell h), \quad j = 0, 1, 2, \ldots, N.$$

The algorithm then proceeds as follows, see also Figure 4.

1. Use (2.20) to calculate $A(x_i)$ from data at grid line $i - 1$. The two arrows in Figure 4 indicate this operation.

2. Use (2.19) to calculate $R_{i,j}$, $j = 1, 2, \ldots, N - i$ from old and new data at grid lines $i - 1$ and i, respectively. Notice that the right hand side of (2.19) is now known from the previous step.

3. Repeat the previous steps to move one grid line deeper into the medium, until the right edge of the slab, $i = N$, is reached.

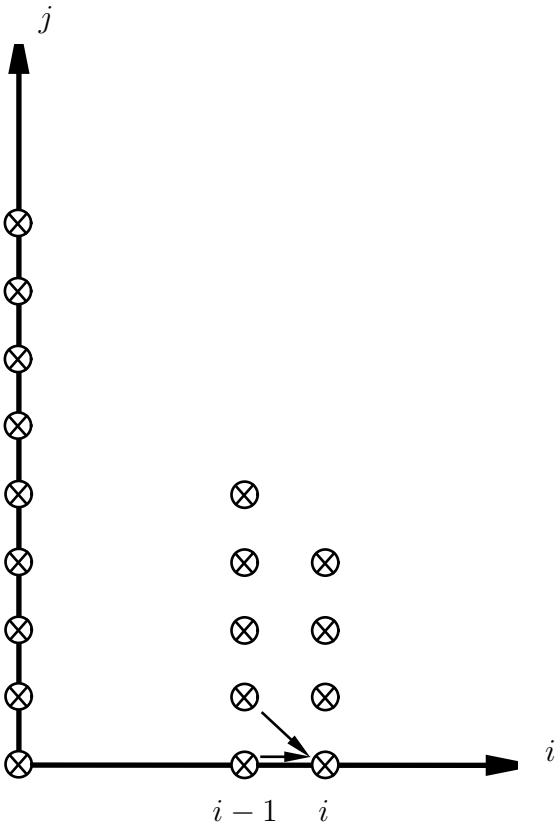


Figure 4: The numerical algorithm.

2.4 The Green functions

It is also possible to arrive at a numerically faster algorithm by using another relation between the fields $E^+(z,t)$ and $E^-(z,t)$, defined at the position z. The representation used to derive the imbedding equation in Section 2.3, given by (2.7), gives the reflection kernel for the subsection problem $[z, L]$. It is also possible to relate the fields $E^+(z,t)$ and $E^-(z,t)$ to the external excitation $E^+(0,t)$. These relations are integral operators represented by

$$E^+(z,t) = \sqrt{\frac{c(z)}{c_1}}\left\{E^+(0,t-\ell x(z)) + \int_{-\infty}^{t-\ell x(z)} G_1(z,t-t')E^+(0,t')\,dt'\right\} \quad (2.22)$$

$$E^-(z,t) = \sqrt{\frac{c(z)}{c_1}}\int_{-\infty}^{t-\ell x(z)} G_2(z,t-t')E^+(0,t')\,dt', \quad (2.23)$$

where $x(z)$ is defined in (2.11). The two kernels $G_1(z,t)$ and $G_2(z,t)$ are called the Green functions. The sum of (2.22) and (2.23) gives the total internal field in the slab, see (2.5) and Figure 5. This is not true for the sum of the fields $E^+(z,t)$ and $E^-(z,t)$ in Section 2.3 since the left end point of the slab is varied and thus the physical set-up changed.

The representations in (2.22) and (2.23) lead to a very efficient way to calculate the internal field, see [7]. However, calculations of the internal fields are not the

main topic of this paper and this matter is, therefore, not pursued here, since the focus is on solving the inverse problem.

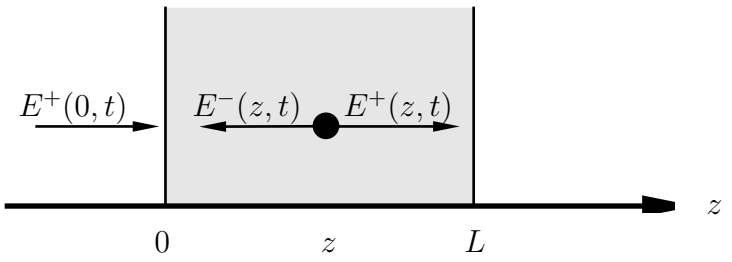


Figure 5: Wave splitting.

The following boundary conditions on the Green functions are easy to see by letting $z = 0$ in (2.22) and (2.23), and comparing with (2.7).

$$G_1(0,t) = 0$$
$$G_2(0,t) = R^+(0,t) = R^+(t).$$

Following the same line of analysis as in Section 2.3, the dynamics, (2.6), and this new relations between the fields $E^+(z,t)$ and $E^-(z,t)$, given by (2.22) and (2.23), imply that the Green functions $G_1(z,t)$ and $G_2(z,t)$ satisfy a system of first order linear differential equations. Lengthy, but straightforward, calculations show

$$\partial_z G_1(z,t) + \frac{1}{c(z)}\partial_t G_1(z,t) = -\frac{c'(z)}{2c(z)} G_2(z,t), \tag{2.24}$$

$$\partial_z G_2(z,t) - \frac{1}{c(z)}\partial_t G_2(z,t) = -\frac{c'(z)}{2c(z)} G_1(z,t), \tag{2.25}$$

with the boundary conditions

$$G_1(z,\ell x(z)) = -\frac{1}{8}\int_0^z \frac{c'(z')^2}{c(z')}\,dz' \tag{2.26}$$

$$G_2(z,\ell x(z)) = \frac{1}{4}c'(z). \tag{2.27}$$

The numerical computations, just as in Section 2.3, become more easy if a normalized travel time coordinate transformation is made.

$$x = x(z) = \int_0^z \frac{dz'}{\ell c(z')}$$
$$s = t/\ell$$
$$G^+(x,s) = \ell G_1(z,t)$$
$$G^-(x,s) = \ell G_2(z,t).$$

The system of first order linear differential equations then becomes

$$\partial_x G^+(x,s) + \partial_s G^+(x,s) = \frac{1}{2}A(x)G^-(x,s) \tag{2.28}$$

$$\partial_x G^-(x,s) - \partial_s G^-(x,s) = \frac{1}{2}A(x)G^+(x,s), \tag{2.29}$$

with the boundary conditions

$$G^+(x,x) = -\frac{1}{8}\int_0^x A^2(x')\,dx' \tag{2.30}$$

$$G^-(x,x) = -\frac{1}{4}A(x), \tag{2.31}$$

and where $A(x)$ is defined as in (2.15).

For the numerical computations create a grid of points (x_i, s_j) in (x,s) space similar to the one in Section 2.3. The discretized points are defined by $x_i = ih$, $s_{i+2j} = (i+2j)h$, $i = 0,1,2,\ldots,N$, $j = 0,1,2,\ldots,N-i$, where $h = 1/N$ and N is an integer. Denote

$$\begin{aligned} G^+_{i,j} &= G^+(x_i, s_{i+2j}) \\ G^-_{i,j} &= G^-(x_i, s_{i+2j}) \\ A_i &= A(x_i). \end{aligned}$$

With the use of the trapezoidal rule in (2.28) and (2.29) the following algorithm is easily found

$$\begin{aligned} G^+_{i,j} = &\left\{ G^+_{i-1,j} + \frac{h}{4}\left\{ A_i G^-_{i-1,j+1} + A_{i-1}G^-_{i-1,j} \right\} \right. \\ &\left. + \frac{h^2}{16}A_iA_{i-1}G^+_{i-1,j+1} \right\} \left\{ 1 - \frac{h^2}{16}A_i^2 \right\}^{-1} \end{aligned} \tag{2.32}$$

$$G^-_{i,j} = G^-_{i-1,j+1} + \frac{h}{4}\left\{ A_iG^+_{i,j} + A_{i-1}G^+_{i-1,j+1} \right\}, \tag{2.33}$$

where $i = 1,2,\ldots,N$, $j = 0,1,2,\ldots,N-i$. The discretized boundary values of G^+ and G^- in (2.30) and (2.31) become

$$G^+_{i,0} = G^+_{i-1,0} - \frac{h}{16}\left\{ A_i^2 + A_{i-1}^2 \right\} \tag{2.34}$$

$$G^-_{i,0} = -\frac{1}{4}A_i, \tag{2.35}$$

where $i = 1,2,\ldots,N$. The error made in (2.32), (2.33) and (2.34) is of order $\mathcal{O}(h^3)$. The values of A_i are determined by solving

$$A_i\left\{ 1 + hG^+_{i-1,0} - \frac{h^2}{16}A_{i-1}^2 - \frac{h^2}{16}A_i^2 \right\} + 4\left\{ G^-_{i-1,1} + \frac{h}{4}A_{i-1}G^+_{i-1,1} \right\} = 0, \tag{2.36}$$

for A_i. This equation is obtained by letting $j = 0$ in (2.33) and then using (2.34) and (2.35).

The input data in the numerical algorithm are the same as in Section 2.3, given by (2.21), and the reconstruction proceeds in two steps. First a reconstruction of $A(x)$ and then a transformation to $z(x)$ and $\epsilon(z(x))$, where the latter step is identical to the one in Section 2.3. To find $A(x)$ initialize G^+ and G^-

$$\begin{aligned} G^+_{0,j} &= 0 \\ G^-_{0,j} &= R(0,2jh) = \ell R^+(0,2j\ell h) = \ell R^+(2j\ell h), \end{aligned}$$

where $j = 0, 1, 2, \ldots, N$. The algorithm then proceeds similar to the one in Section 2.3.

1. Solve (2.36) for A_i from data at grid line $i-1$.
2. Use (2.32) and (2.33) to calculate $G^+_{i,j}$ and $G^-_{i,j}$, $j = 1, 2, \ldots, N-i$ from old and new data at grid lines $i-1$ and i, respectively. Notice that the right hand side of (2.32) is now known from the previous step.
3. Repeat the previous steps to move one grid line deeper into the medium, until the right edge of the slab, $i = N$, is reached.

This numerical algorithm based upon the Green functions approach is considerably faster than the one presented in Section 2.3, which was based upon an invariant imbedding procedure. This is simply due to the absence of time convolution integrals in the Green functions approach. However, the invariant imbedding approach provides a stronger mathematical tool for the theoretical analysis and it is therefore motivated to present both methods here. It is also possible to compare the reconstructions made with the two algorithms. No significant difference in the accuracy of the reconstructions is, however, found for the class of profiles that is considered in this paper.

3 General considerations

The reconstruction algorithms presented in Section 2 require that the impulse response $R^+(t)$ of the medium is known for one round trip. However, this is an idealized experimental situation where the incident field is a Dirac delta function. The real incident pulse has always a finite width. The algorithm can be used if the reflected wave form is deconvolved with the incident wave form, i.e. the reflection kernel or impulse response $R^+(t)$ can be extracted from (2.3)

$$E^r(t) = \int_{-\infty}^{t} R^+(t-t')E^i(t')\,dt'.$$

This deconvolution is also very efficient as a tool for removing certain errors which will be discussed in Section 6 below, but as any tool it has limitations, which restrict its use. It is sometimes stated that the deconvolution or some other suitable

data processing can significantly improve the experimental results. This is generally not true because the measured data contain noise. Thus, outside a limited frequency band the quality of the data is too low to be processed. This is easily demonstrated by the following experiment:

1. Measure the same signal twice and calculate the Discrete Fourier Transform (DFT) of the two measurements, see Figure 6a for an example of a measurement of the incident pulse and Figure 6b for its DFT.

2. Divide the Fourier transform of the first measurement with the Fourier transform of the second. The result is showed in Figure 7.

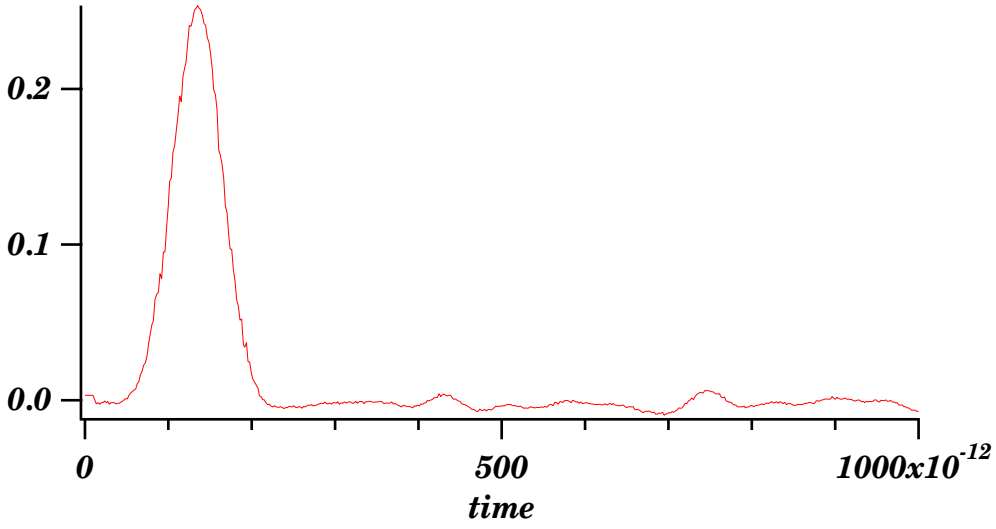


Figure 6a: An example of a recording of the incident waveform.

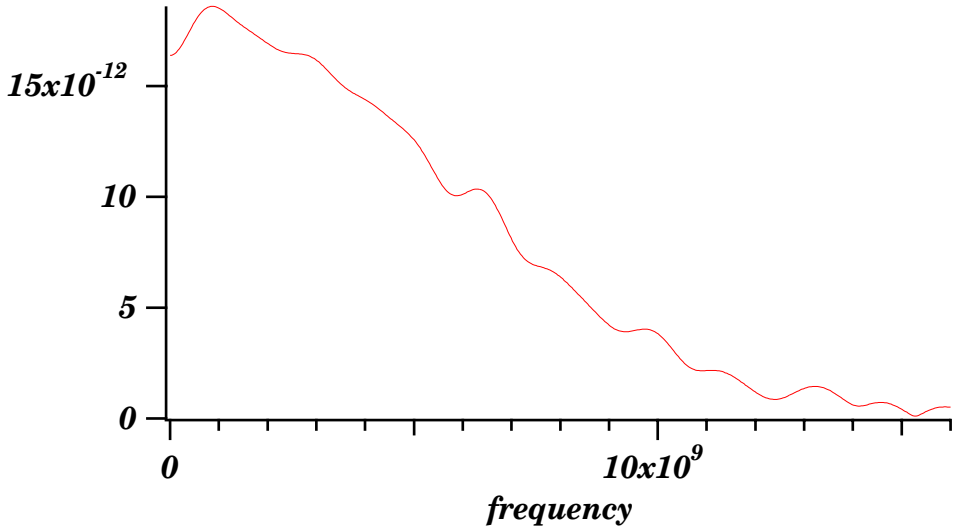


Figure 6b: The DFT of Figure 6a.

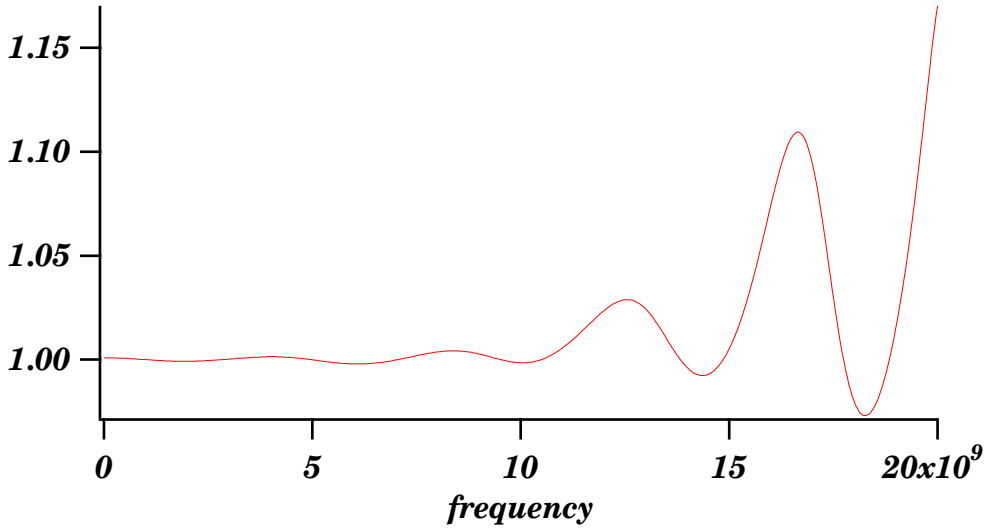


Figure 7: The quotient of the two DFT waveforms.

The expected result is that this quotient is identical to one and this is indeed the case at low frequencies. However, when the frequency approaches the upper frequency limit of the system, the quotient of two small noisy quantities results in violent oscillations. This effect can only be reduced by bandlimiting the measured signal in the time or frequency domain or both. This filtering, if carefully done, can somewhat extend the useful bandwidth. Additional information can be extracted from the "out of band" part of the data if additional properties of the signal are known by other means, i.e. *a priori* information. The more *a priori* knowledge about the unknown sample, the more specific algorithms can be used and the better results are expected. A good example to this is found in Section 6.6.

The main practical problem in deconvolution is to obtain a relevant incident field or waveform. The pulse must be measured with the same system as used for the actual measurement, i.e. for the reflected field or waveform. This can be obtained by introducing a short in the sample cell at the position of the sample. However, this requires frequently disassembling and assembling of the cell which wears and tears the precision connectors.

4 Deconvolution

The kernel $R^+(t)$, which is required by the reconstruction algorithm, is computed by a deconvolution in the frequency domain. There is also the possibility to perform the deconvolution in the time domain. A recent contribution to the time domain deconvolution utilizing the Singular Value Decomposition (SVD) is found in [10]. The time domain approach has not been pursued in the present paper.

The signal reflected from the sample is sampled in the time domain and transformed to the frequency domain through a DFT. Assume that $t_k = T\frac{k}{N}$, $k = 0, 1, 2, \ldots, N-1$, is a partition of the time interval $[0, T]$ into N equidistant points.

Then the DFT of $f_k = f(t_k)$ is defined as

$$F_p = \sum_{k=0}^{N-1} f_k e^{-i\omega_p t_k} = \sum_{k=0}^{N-1} f_k W^{pk}, \quad p = 0, 1, 2, \ldots, N-1, \tag{4.1}$$

where $\omega_p = \frac{2\pi p}{T}$, $p = 0, 1, 2, \ldots, N-1$ and where

$$W = \exp\left\{\frac{-i2\pi}{N}\right\}.$$

This can be computed in an efficient way using any of the FFT algorithms that are available. The inverse transform is

$$f_k = \frac{1}{N} \sum_{p=0}^{N-1} F_p e^{i\omega_p t_k}, \quad k = 0, 1, 2, \ldots, N-1.$$

The limitation of this approach is that the sample points $z_k = \exp\left\{\frac{i2\pi k}{N}\right\}$ are equally spaced around the unit circle in the complex z-plane, and $\Delta t \Delta f = \frac{T}{N}\frac{1}{T} = \frac{1}{N}$. Thus, the resolution in the time domain and the number of sampling points determine the resolution in frequency domain. This is often undesirable. In this paper the limited resolution in frequency domain was unacceptable. This problem can be solved by using "zero padding", which means increasing the N with a number of dummy zeros, but the computing time will then increase drastically. A more efficient way to remedy this problem, and to be free from the limitations inherent in the ordinary DFT, is to adopt another algorithm to generate the sampled amplitudes in the frequency domain - the Chirp z-Transform (CZT), see Ref. [9]. The CZT transform has the advantage that it is possible to choose the number of time-samples different from the number of frequency-samples and it is possible to calculate the frequency-samples only in the frequency interval of interest. The reference signal is also transformed to the frequency domain using the CZT transform.

The CZT algorithm is based on the identity $pk = \frac{1}{2}(p^2 + k^2 - (k-p)(k-p))$. The DFT defined in (4.1) is then

$$F_p = W^{\frac{p^2}{2}} \sum_{k=0}^{N-1} g_k W^{-\frac{(k-p)^2}{2}}, \quad p = 0, 1, 2, \ldots, M-1,$$

where $g_k = f_k W^{\frac{k^2}{2}}, k = 0, 1, 2, \ldots, N-1$. This can be viewed as a three-step process consisting of two multiplications and one convolution, which can be performed using FFT. The most important feature of this approach compared to the standard FFT is that the number of time samples need not be equal to the number of frequency samples, i.e. the spacing and the starting point of the argument of F_p are arbitrary, see Ref. [9].

Before the CZT transformation, same windowing is sometimes necessary. Depending on the shape of the signals the window function is chosen so that a good compromise between frequency resolution and amplitude accuracy is obtained. No

weighting function is ideal. The bandwidth, ripple, sidelobe level, sidelobe fall-off rate, shape factor can be optimized but not all with the same window. The Kaiser-Bessel weighting function, defined as

$$w(t) = 1 - 1.24\cos\frac{2\pi t}{T} + 0.244\cos\frac{4\pi t}{T} - 0.00305\cos\frac{6\pi t}{T}, \tag{4.2}$$

is superior to the other filters with respect to selectivity. This makes it excellent for separation of closely spaced components with widely different levels.

A formal inverse DFT of the quotient of the DFT of the reflected and the incident waveforms, respectively, is not possible because of noise at the higher frequencies (above 12 GHz). The deconvolution problem is an ill-posed problem and filtering is necessary to yield a stable and physically consistent result. A regularization filter is used here to stabilize these ill-posedness effects. The reflection kernel $\hat{R}^+(\omega)$ in the frequency domain then has the form

$$\hat{R}^+(\omega) = \frac{Y(\omega)X(\omega)^*}{|X(\omega)|^2 + \lambda C(\omega)},$$

where $Y(\omega)$ and $X(\omega)$ are the Fourier transformed reflected and incident waveforms, respectively, and where star $(*)$ denotes the complex conjugate. The function $C(\omega)$ is chosen as $C(\omega) = \omega^4$, see also Ref. [8] for more details about this choice. To determine the optimum value of the parameter λ the procedure described above is used. Two recordings of the same signal are measured and deconvolved using one of them as a reference. The constant λ is then adjusted so that a smooth transition from one to zero at the high frequency end of the spectrum is obtained. A Kaiser-Bessel window, see (4.2), is used on the kernel before it is transformed into the time domain. The transformation to the time domain is made by using an inverse CZT transform. It has the advantage that the number of time points, the beginning of the time trace, and the length of the time trace for the kernel can be varied arbitrarily.

The deconvolution procedure is illustrated below in Section 7.

5 Experimental set-up

The measurement system can be described as a pulsed radar. A short pulse is generated and sent towards the sample to be investigated. The reflected waveform is recorded and using the algorithms described in Section 2.3 or 2.4 the permittivity profile of the sample is reconstructed.

The set-up is similar to a Time Domain Reflectometer (TDR). However, a short duration pulse is used as excitation instead of a step. A traditional TDR uses a through-line sampler. This gives the best possible signal-to-noise ratio (SNR) and the largest bandwidth. In our application a terminated sampler and a power divider is used because the amplitude errors due to multiple reflections are reduced by attenuations in the divider. The loss of resolution and dynamic range due to losses in the divider are insignificant. Because the algorithms used are developed for a one-dimensional case, the system is build using coaxial components. Other TEM

transmission lines, such as striplines, have the advantage of being open structures which makes it easier to insert and remove samples in the sample cell. Flat samples are, however, more difficult to machine than cylindrical ones. A section of a coaxial air line was used as a sample cell. The longest air line available is only 200mm long which requires short samples and, consequently, pulses with very short rise time have to be used in order to obtain adequate relative resolution.

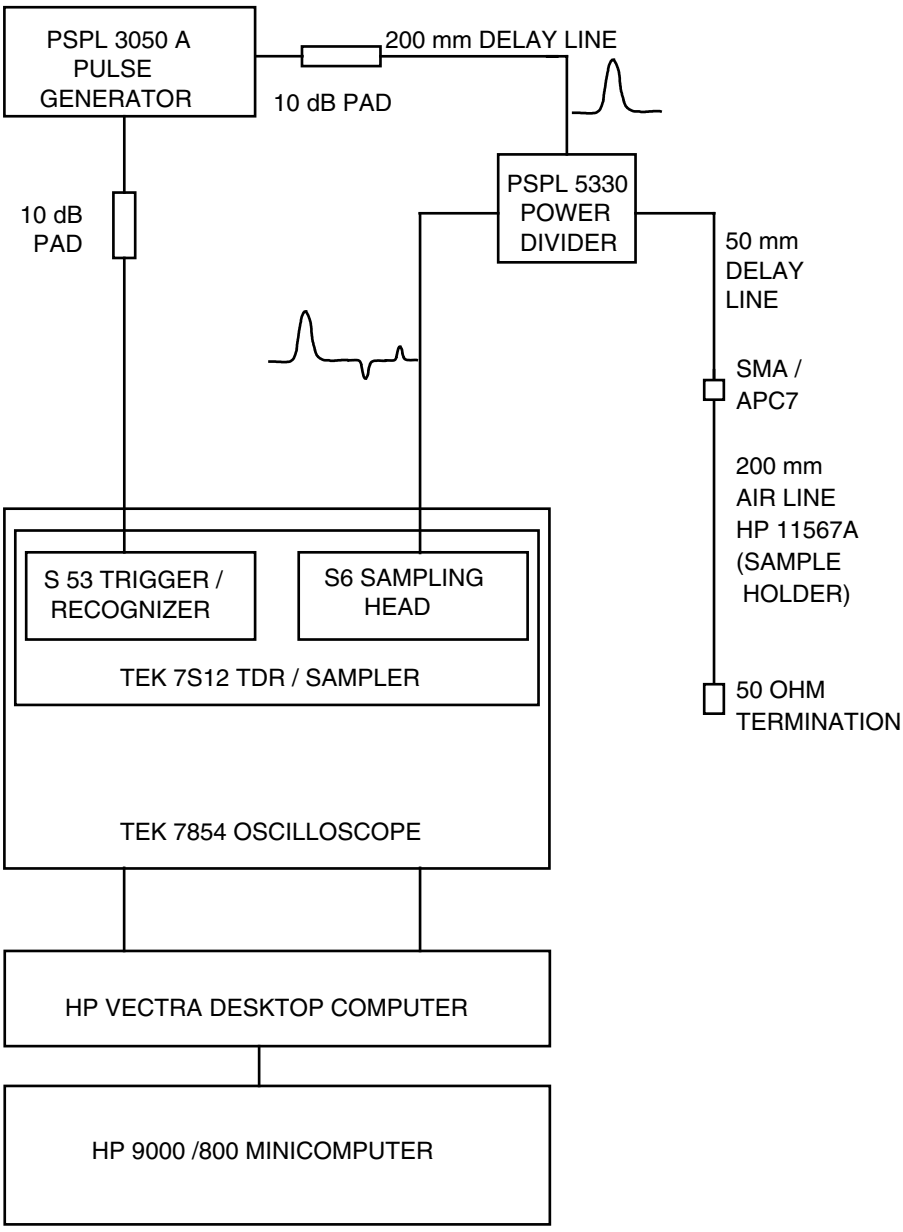


Figure 8: The experimental set-up.

The two main components of the system are the pulse generator (Picosecond Pulse Labs Model 3050A) and the sampling oscilloscope (Tektronix TDR/Sampler

7S12 with a S-6 head installed in a 7854 frame). The specifications of these components are summarized in Table 2 and 3, respectively, and an outline of the experimental set-up is given in Figure 8. The signal from the generator is connected to the sample cell and to the oscilloscope through a power divider optimized for broadband pulses. The generator is also a source of trigger pulses. The reflection from the sample is continuously monitored and can be sampled, averaged, and stored under the control of a desktop computer. Data are then transferred to a mainframe computer for processing. Immediately after recording of the reflection data from an unknown sample a short is placed in its place and a new recording is made. This waveform is an inverted copy of the incident waveform (including all imperfections of the measurement system) and is used as a reference for deconvolution.

Specifications	
Pulse width (FWHM, 50%)	60ps
Pulse width (10%)	115ps
Rise & Falltime (10%-90%)	45ps
Amplitude	4V
Residual ringing	±3%
Precursor	±1%
VSWR	3:1
Jitter	±2.5ps
Repetition rate	1MHz
Trigger risetime	0.5ns

Table 2: Specifications of the pulse generator PSPL Model 3050A.

Specifications	
Risetime	30ps
Displayed noise	5mV
Jitter	10ps
Maximum input voltage	1V p-p
Input resistance	10kOhm
Deflection factor accuracy	3%
Time/Div accuracy	3%

Table 3: Specifications of the Tektronix TDR/Sampler 7S12 with S-6 head installed in a 7854 frame.

6 Analysis of errors

In this section a detailed analysis of the errors of the experimental set-up is presented.

6.1 Signal quality

The most important parameter of the signal source is the rise time of the pulse, because it determines the resolution of the system. The generator is one of the fastest that is available commercially. Moreover, it has two important characteristics: relatively large output voltage, which gives a good SNR of the system, and very little residual ringing. This last property is essential, because low level perturbations on the baseline, which can only partially be removed from the recorded data, will seriously degrade the performance of the system. The generator also gives clean and stable trigger pulses which reduces the system jitter.

6.2 Cable loss

As the pulse travels down the line it is attenuated by dielectric losses and finite losses in the conductors. The losses are frequency dependent and high frequencies are generally more attenuated. This degrades the risetime of the pulse and limits the distance resolution. To avoid this problem the cables are kept as short as possible. However, certain lengths are necessary to avoid multiple reflections and to create a clean portion of the base line.

6.3 Internal reflection

The discontinuities present in the pulse source, connectors and other parts of the system will degrade the performance. Consider a transmission line with two discontinuities on it. If the first one has a reflection coefficient of 0.1, the error in determining the reflection coefficient of the other will be 6%. The connectors (APC7 and SMA) contribute very little to this error when they are new. However, after a few month of use, even if handled with care, reflections became noticeable. This effect can be reduced using deconvolution.

6.4 Oscilloscopes

The fastest available oscilloscopes have a rise time of the order of 300 ps. For analysis of faster signals the sampling technique must be used. The obvious drawback of this approach is that only repetitive waveforms can be captured. The main problem is that the design of the sampling bridge makes it difficult to determine the baseline level.

6.5 Input impedance error

The input of the oscilloscope has an impedance of 50 Ohms. For Tek S-6 terminated by a HP load, the VSWR is <1.05 which corresponds to a variation in resistance of 47.6 to 52.5 Ohms. When connected to a 50 Ohms source, the measured voltage can differ by 2.5% from the correct value. The attenuators, which generally have lower performance, contribute to this error. However, if, as in our measurements, all data are measured using the same system layout and the same sensitivity settings, this

effect is not noticeable. The discontinuities cause reflections back into the source. If the source is not matched, these reflections will be re-reflected and will distort the measured waveform. This can be avoided by using a suitable time window.

6.6 Baseline error

Due to the design of the sampling head, it is difficult to determine the exact position of the baseline. The reconstruction algorithm assumes that this position is at zero level. If this is not the case, the error is integrated and an erroneous slope in the reconstructed permittivity profile is introduced. Remember, that without any *a priori* knowledge, the reconstruction algorithms in Section 2.3 or 2.4, will always generate a permittivity profile that corresponds to the given reflection kernel. It is possible to eliminate this problem by an offset of the recorded data and a repetition of the calculation until the relative permittivity outside the sample is 1. This procedure requires that the approximate length of the sample is known, which is the case in our tests, but not in general. This corresponds to the *a priori* knowledge introduced into the inverse scattering problem. This baseline error, if not corrected, can easily create a 20% error in the reconstructed permittivity profile. If more information is available (i.e. that the sample has constant or piecewise constant permittivity), additional degrees of correction can be introduced. Figure 9 a and b clearly illustrate this effect.

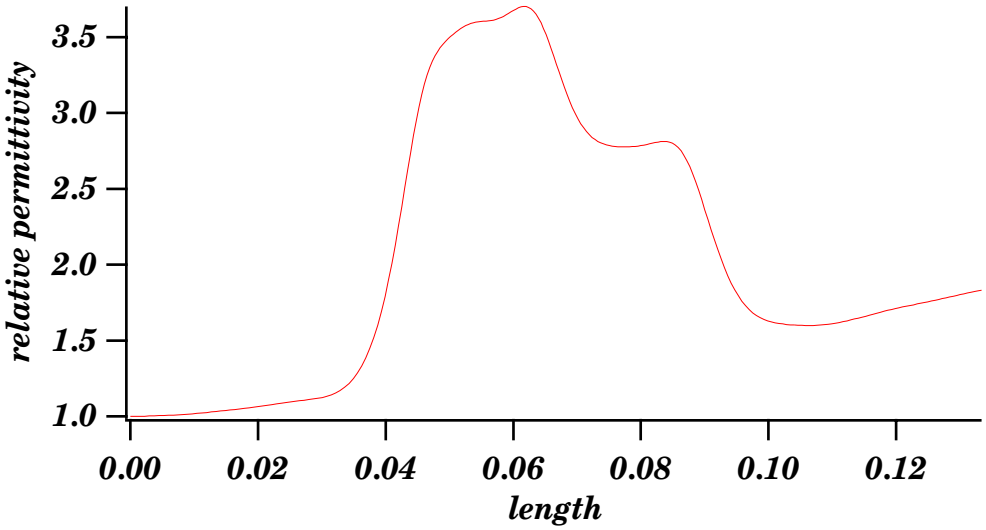


Figure 9 a: Error created by 2 % baseline offset.

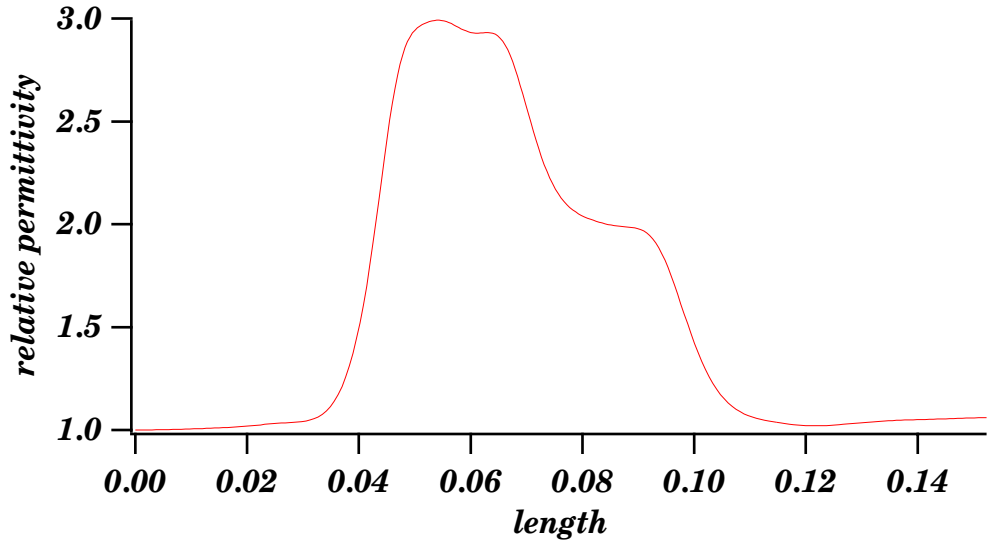


Figure 9 b: Corrected baseline error.

6.7 SNR of an A/D converter

A 10 bits converter in the TEK oscilloscope gives a SNR = 62 dB, which is more then enough compared to the SNR of the signal.

6.8 Amplitude linearity

All the measurements are made at the same sensitivity setting. Consequently, the absolute value of the amplitude error has no importance. The vertical linearity is investigated by measuring a number of identical waveforms which are offset by different amounts. This offset is then compensated by adding a constant to the stored waveform. No significant differences are detected when the processed waveforms are compared, as long as the input is kept under 250 mV.

6.9 Jitter and noise

Jitter is the random time uncertainty of a waveform point relative to a reference point. In a TDR application jitter is particularly critical because the information is extracted from time relations. The amount of jitter depends on the slope of the trig signal and the amount of noise in it. During the sampling of a signal with a slope different from zero, values randomly distributed around the signal value are recorded. This makes jitter difficult to distinguish from amplitude noise. However, the amount of noise due to jitter is largest at the large slope portions of the signal as opposed to the amplitude noise.

Jitter always limits the bandwidth of the system because the distribution of the noise around the signal is not symmetric. The mean value at points where the signal has negative second derivative is below the signal and where the second derivative is positive the mean value is above the signal. After averaging the original waveform will be reconstructed only for segments with constant slope. All other parts of the signal will be smoothed which results in loss of resolution. The distortion caused by this effect cannot be determined exactly because the probability distribution of the error depends on the waveform and is therefore unknown. It can be estimated, however, by measuring the noise at the most vertical part of the signal. Both kinds of noise can be removed from the signal by measuring it several times and then averaging. Generally, the larger the number of recordings averaged, the better the signal quality becomes. However, the process takes time and if the number of recordings is growing one must take into account the long-time drift of the oscilloscope. For the set-up reported in this paper, 100 times is about optimum.

6.10 Sweep errors

Sweep errors consist of the absolute error and the linearity error. Both can be calibrated by using a stable sinewave and using the zero crossing points as time markers. The frequency of the wave must be high in order to obtain a large number of calibration points. Unfortunately, due to the design of the TEK oscilloscopes trigger circuit, the highest possible frequency is about 900 MHz which is not sufficient. This problem is solved by using a chain of frequency multipliers connected to a very stable 500 MHz source which is also used as a trigger. Alternatively, a high Q-value resonator is used, triggered by short pulses with repetition frequency of 1 MHz.

6.11 Bandwidth

Many kind of errors can be normalized or filtered out. The price for these improvements in signal quality is generally a loss of effective bandwidth. Thus, one should try to start with as large bandwidth as possible even when the required resolution does not motivate this. In this set-up all the components were specified for operation to at least 18 GHz. The resulting bandwidth of the system is 12 GHz. This corresponds to a resolution of about 7 mm for permittivity values in the range 2-4 where the measurements are made. A 20 mm long sample with constant permittivity showed a clean flat portion of permittivity in the middle of the profile.

6.12 Residual ringing

The ringing, which mainly originates from the generator, can be divided into two parts. Some ringing is created in the step generating parts, and cannot be affected, and some is created in the following, pulse shaping components, and can be delayed with respect to the main pulse, see Figure 10. This makes it possible to create a "quiet zone" where the first ringing has died out and the second has not arrived yet.

The sample should be placed in this zone. This zone limits the maximum length of the sample to about 90mm.

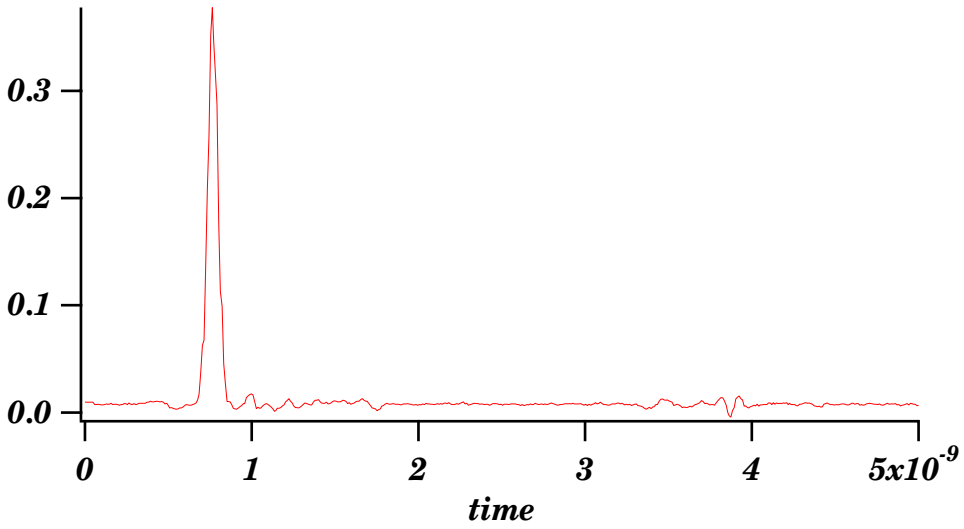


Figure 10: Residual ringing of the incident waveform.

7 Results

The general scheme for processing the recorded data is illustrated in Figures 11 a–11 j. These figures show the step by step procedure to obtain the inversion of the permittivity profile. The recorded incident and reflected waveforms, Figures 11 a and b, are first transformed into the frequency plane, Figures 11 c and d, using the transformation described in Section 4. Figures 11 e and f show the deconvolved reflection kernel in the frequency domain with different normalizations. The time domain behavior of the kernel from Figure 11 e is showed in Figures 11 g. Finally, in Figures 11 i and j the reconstructions corresponding to Figures 11 g and h, respectively, are showed illustrating the trade off between the resolution and amplitude accuracy when different windows are used.

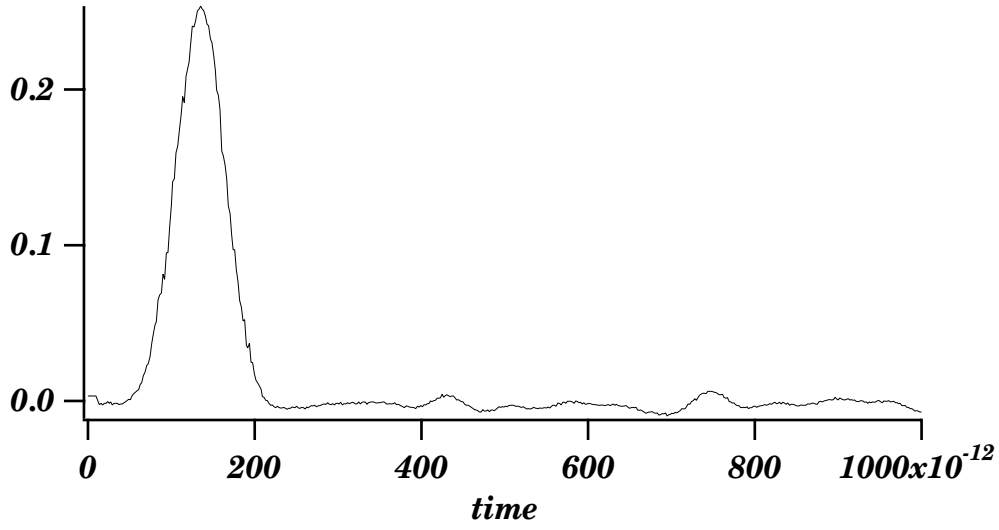


Figure 11 a: The incident waveform.

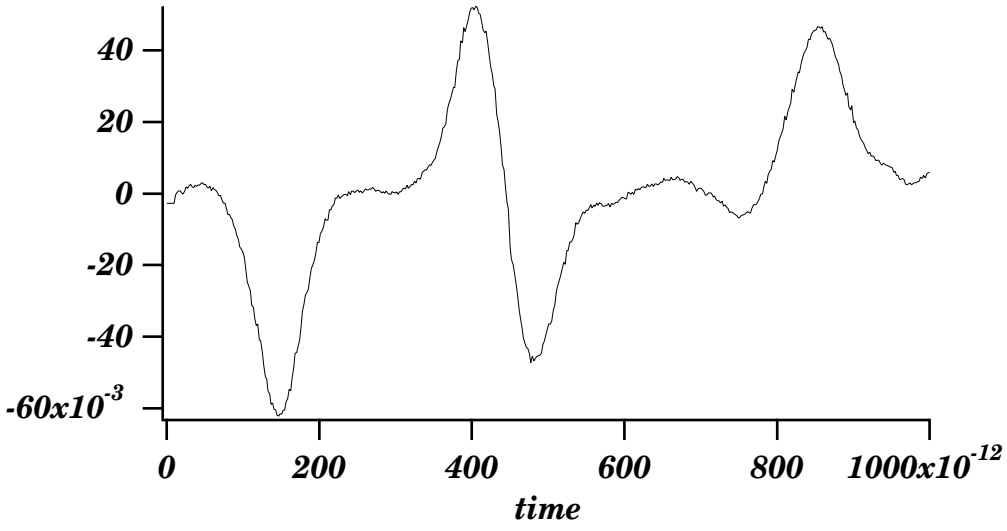


Figure 11 b: The reflected waveform.

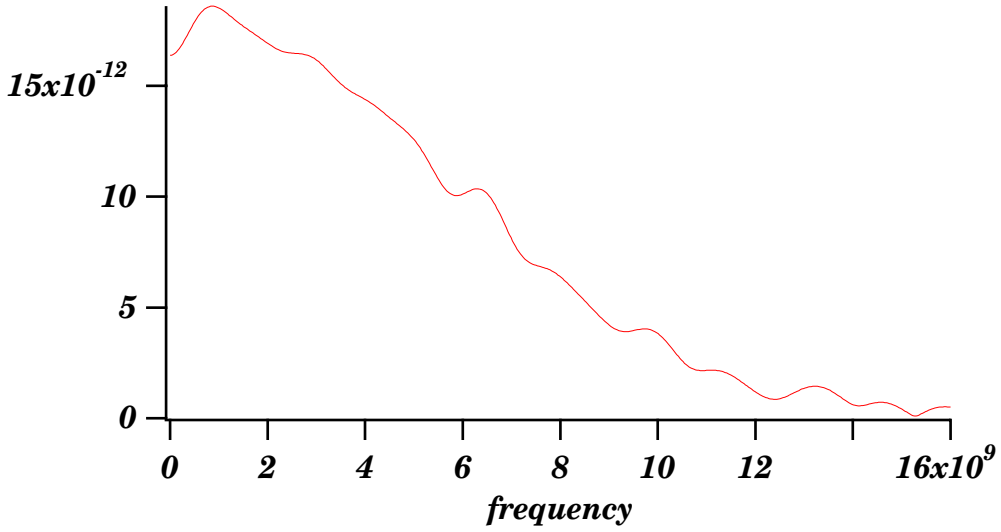


Figure 11 c: The DFT of the incident waveform.

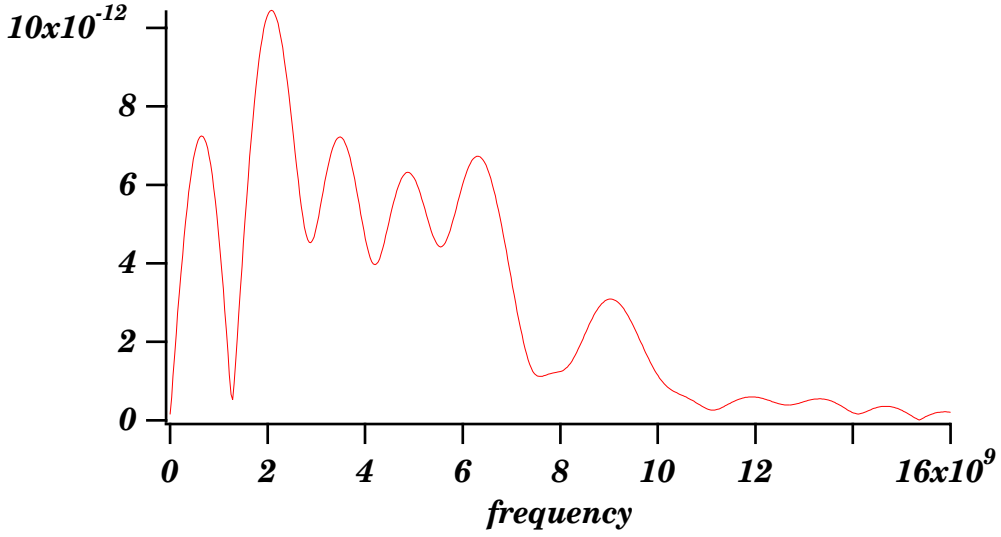


Figure 11 d: The DFT of the reflected waveform.

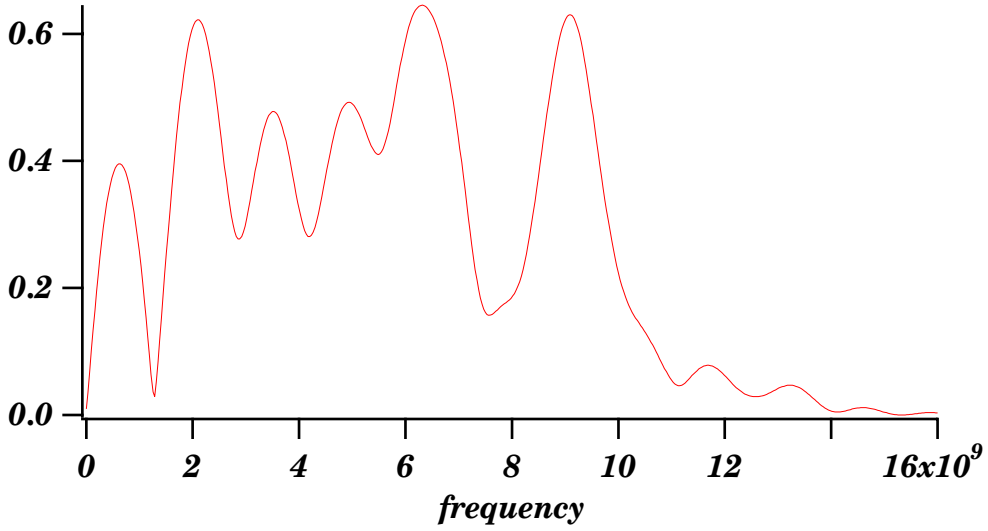


Figure 11 e: Deconvolved reflected waveform with optimum normalization.

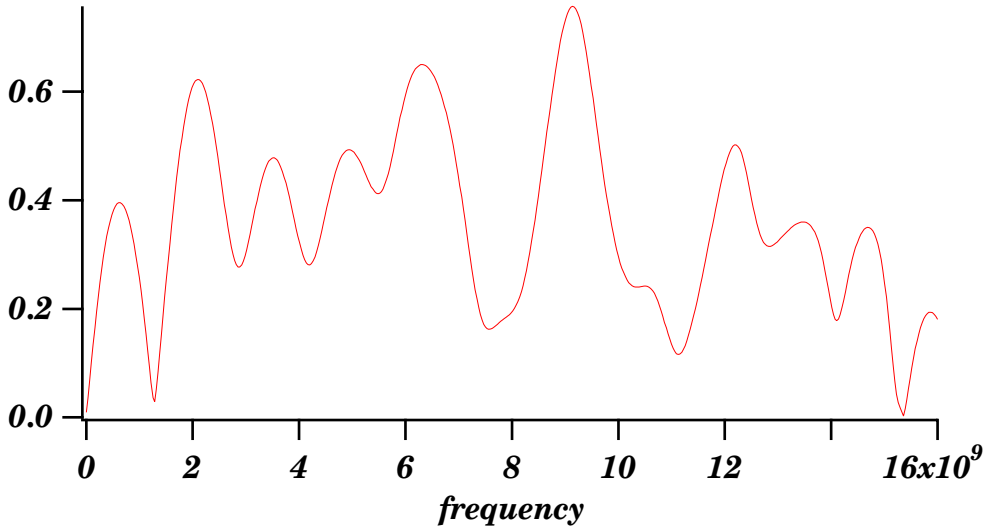


Figure 11 f: Same as Figure 11 e, but with insufficient normalization.

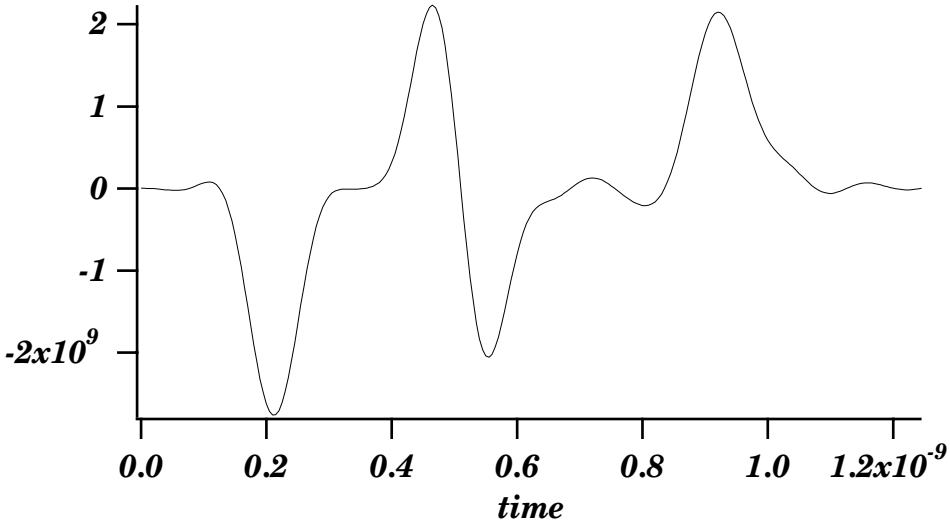


Figure 11 g: Deconvolved reflected waveform with optimum windowing.

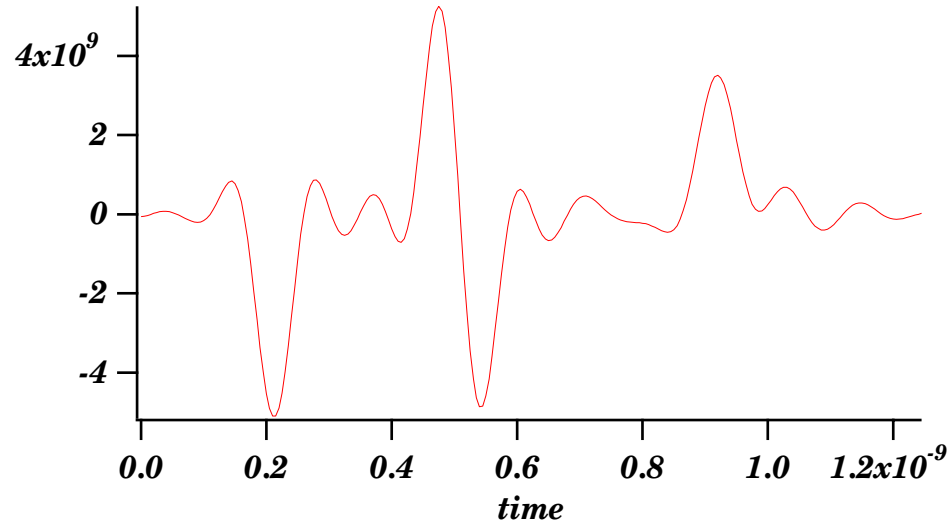


Figure 11 h: As Figure 11 g, but with rectangular windowing.

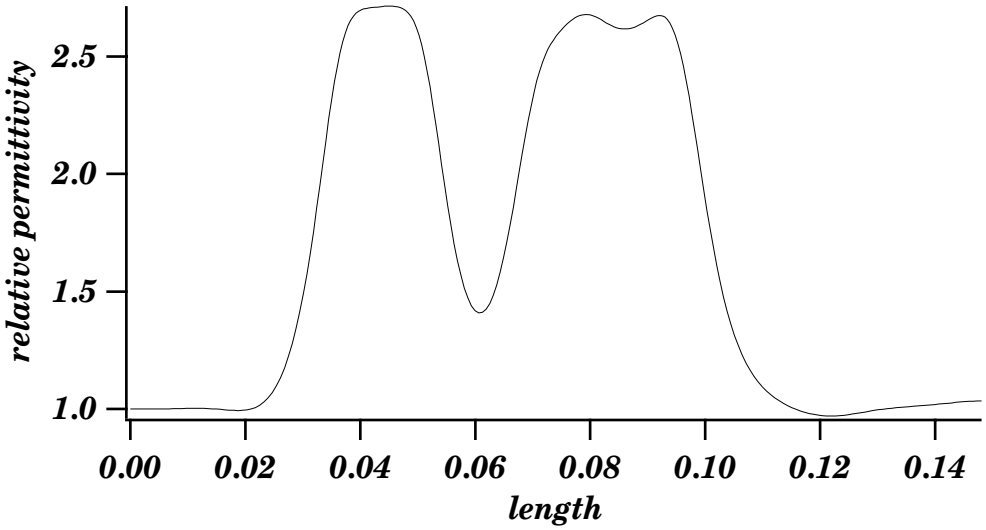


Figure 11 i: Permittivity profile reconstruction from waveform in Figure 11 g.

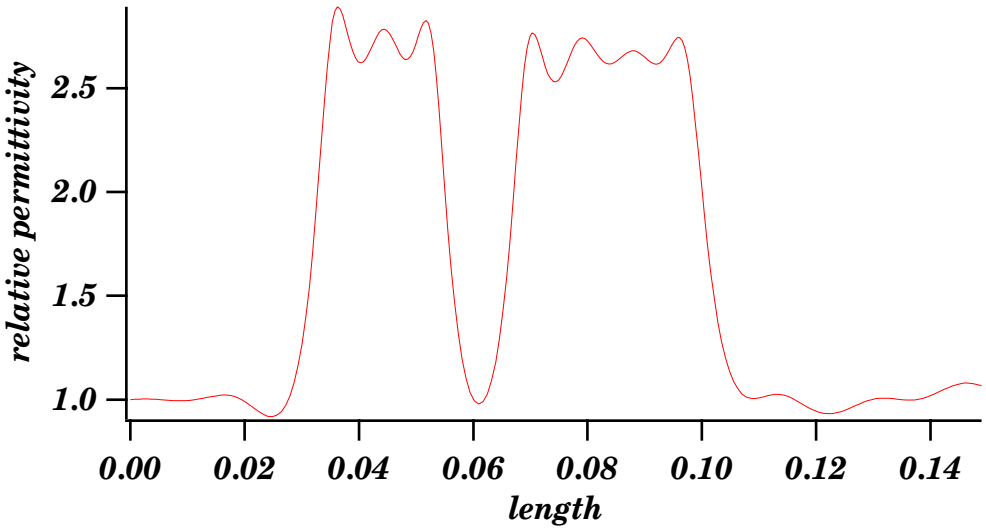


Figure 11 j: Permittivity profile reconstruction from waveform in Figure 11 h.

Two additional sets of figures illustrate the reconstruction of permittivity profiles which are inhomogeneous, see Figures 12–13 and 15–16. The original samples are depicted in Figures 14 and 17. In both the cases the samples are turned round and new measurements with these reflected profiles are made for comparison. The two reconstructions should be mirror images of each other. From these comparisons it

is possible to conclude that the accuracy decreases slightly when the length of the sample increases. This is due to the losses. If the losses are *a priori* known, then these losses can be compensated for. No such compensation is made here.

The resolution of the measurement system is also clearly shown. A comparison of the length of the samples (measures with a sliding calliper) with the length obtained from the reconstructions of the permittivity profiles shows an error of 2%. This is indeed a very sensitive test because the length of the medium is calculated from the reconstructed permittivity.

The last set of reconstructions, see Figures 15–17, also illustrates the resolution of the system in a different way. (The reconstruction depicted in Figure 15 has already been given in Figure 11 i, but it is repeated here to facilitate the comparison.) Between the two dielectric samples (of different lengths, but the same permittivity) there is an air-gap. The length of this air-gap can be varied. The smallest possible air-gap that still gives a fair reconstruction of the vacuum value of the permittivity is a measure of the resolution of the system. This is illustrated in Figures 15–16.

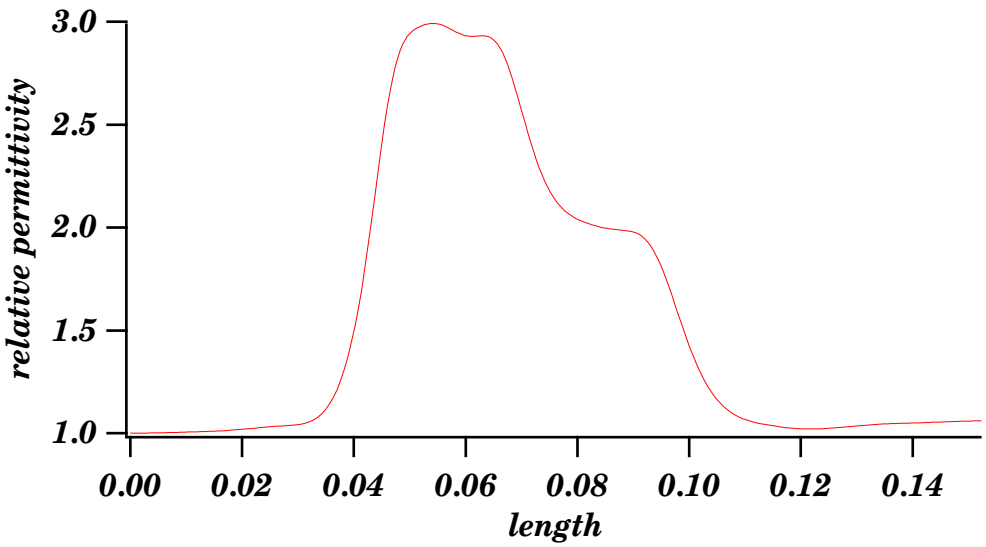


Figure 12: Reconstruction of a composite sample.

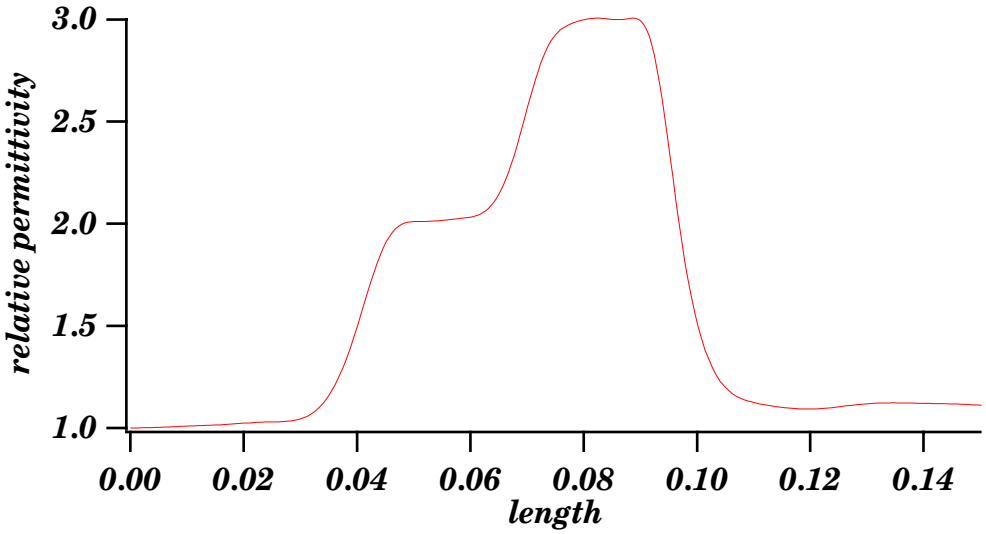


Figure 13: Reconstruction of a composite sample.

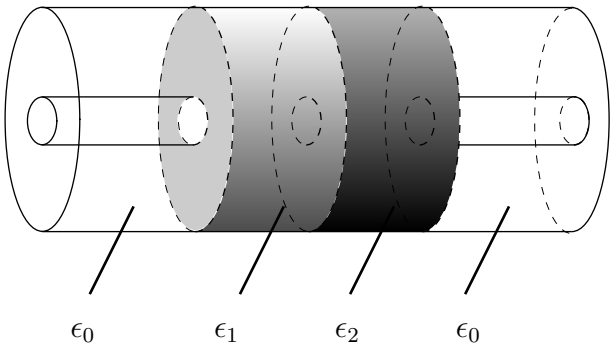


Figure 14: The original sample of Figure 13.

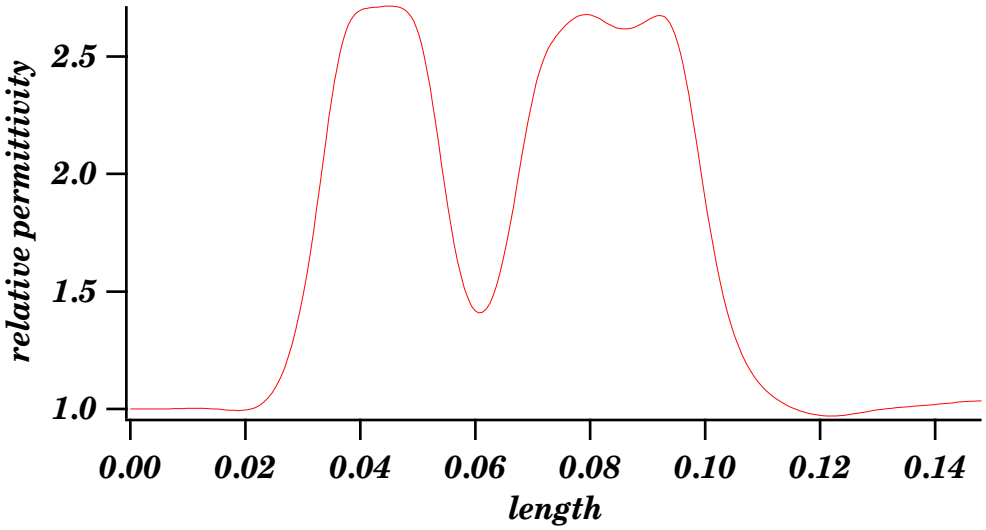


Figure 15: Reconstruction of a composite sample.

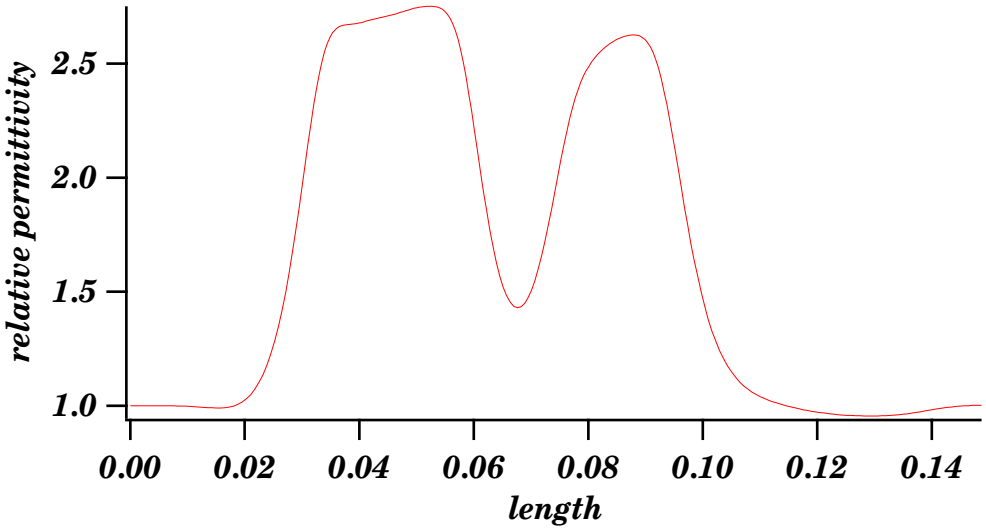


Figure 16: Reconstruction of a composite sample.

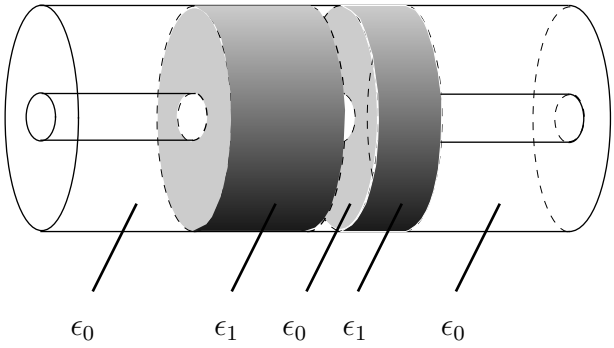


Figure 17: The original sample of Figure 16.

References

[1] Beezley R.S. and R.J. Krueger, "An electromagnetic inverse problem for dispersive media," *J. Math. Phys.*, **26**(2), 317–325 (1985).

[2] Corones J.P., M.E. Davison and R.J. Krueger, "Wave splittings, invariant imbedding and inverse scattering," in *Inverse Optics, Proc. SPIE*, Vol. 413, Ed. A.J. Devaney, (SPIE Bellingham, WA 1983), pp. 102–106.

[3] Kristensson G. and R.J. Krueger, "Direct and inverse scattering in the time domain for a dissipative wave equation. Part 1: Scattering operators," *J. Math. Phys.*, **27**(6), 1667–1682 (1986).

[4] Kristensson G. and R.J. Krueger, "Direct and inverse scattering in the time domain for a dissipative wave equation. Part 2: Simultaneous reconstruction of dissipation and phase velocity profiles," *J. Math. Phys.*, **27**(6), 1683–1693 (1986).

[5] Kristensson G. and R.J. Krueger, "Direct and inverse scattering in the time domain for a dissipative wave equation. Part 3: Scattering operators in the presence of a phase velocity mismatch," *J. Math. Phys.*, **28**(2), 360–370 (1987).

[6] Kristensson G. and R.J. Krueger, "Direct and inverse scattering in the time domain for a dissipative wave equation. Part 4: Use of phase velocity mismatches to simplify inversions," *Inverse Problems*, **5**(3), 375–388 (1989).

[7] Krueger R.J. and R. L. Ochs, Jr., "A Green's Function Approach to the Determination of Internal Fields," *Wave Motion*, **11**, 525–543 (1989).

[8] Nahman N.S. and M.E. Guillaume, "Deconvolution of Time Domain waveforms in the Presence of Noise," NBS Technical note 1047, October 1981.

[9] Rabiner L.R. and R.W. Schafer, "The Chirp z-Transform Algorithm," *IEEE Trans. Audio Electroacoust*, AU-**17**, 86–92 (1969).

[10] Rothwell E.J. and W. Sun, "Time Domain Deconvolution of Transient Radar Data," *IEEE Trans. Antennas Propagat.*, AP-**38**, 470–475 (1990).

Appendix A Computer programs

DEBUGGED USING DDT

```
C PROGRAM DECONV
C
C
      IMPLICIT NONE
C
      INTEGER       NMAX
      INTEGER       IMAX
      PARAMETER     (NMAX=8192,IMAX=400)
C
      REAL          W(1:NMAX),TR(1:NMAX),TI(1:NMAX)
      REAL          AMP(1:NMAX),DB(1:NMAX)
      REAL          FYR(1:NMAX),FYI(1:NMAX)
      REAL          FR(1:NMAX),FI(1:NMAX),T_INC,TZ
      REAL          F_INC,T_INC1,T_INC2,T_ZERO,F_ZERO,LAMBDA
      REAL          F_CUTOFF
      REAL          T_FRONT,T_REAR,DATA(1:NMAX),E(0:IMAX)
      INTEGER       L,L2,NF,NR,KR,KI,KRF,KIF
      INTEGER       N,N2,M,MM,I,NT
      INTEGER       ITER,K,KK,KKK
      INTEGER       SYM,XWIN,YWIN,FWIN,N_HAN
      INTEGER       BA_K,BA_P,BA_R
      INTEGER       K_STEP,K_ZERO,P_STEP,P_ZERO
      CHARACTER*2   FPR,IDX,IDY
      CHARACTER*4   ID
C
      COMMON /INTID/ N_HAN
C
      CALL SCREENCLEAR
C
      N_HAN=25
C
C Indata
C
1000  FORMAT(2A)
2000  FORMAT(4A)
3000  FORMAT(I4)
4000  FORMAT(I3)
      WRITE(6,*) ' '
      WRITE(6,*) '          <><><> PROGRAM DECONVOLUTION <><><>'
      WRITE(6,*) '          <><><>    SUPER-RESOLUTION   <><><>'
```

```
WRITE(6,*) ' '
WRITE(6,*) ' Enter ID of the kernel.'
READ(5,1000) IDY
WRITE(6,*) ' '
WRITE(6,*) ' KERNEL WINDOW. Choose 0 or 7! '
WRITE(6,*) ' '
CALL MENUE
READ(5,3000) YWIN
WRITE(6,*) ' '
WRITE(6,*) ' Baseline adjustment on kernel data? Y=1, N=0'
READ(5,3000) BA_K
WRITE(6,*) ' '
WRITE(6,*) ' Enter size of zero padding on kernel data.'
READ(5,3000) K_ZERO
WRITE(6,*) ' '
WRITE(6,*) ' Enter change in sampling rate. 0,1,2,...?'
READ(5,3000) K_STEP
WRITE(6,*) ' '
WRITE(6,*) ' Enter ID of the input pulse.'
READ(5,1000) IDX
WRITE(6,*) ' '
WRITE(6,*) ' PULSE WINDOW. Choose 0 or 7! '
WRITE(6,*) ' '
CALL MENUE
READ(5,3000) XWIN
WRITE(6,*) ' '
WRITE(6,*) ' Baseline adjustment on pulse data? Y=1, N=0'
READ(5,3000) BA_P
WRITE(6,*) ' '
WRITE(6,*) ' Enter size of zero padding on pulse data.'
READ(5,3000) P_ZERO
WRITE(6,*) ' '
WRITE(6,*) ' Enter change in sampling rate. 0,1,2,...?'
READ(5,3000) P_STEP
WRITE(6,*) ' '
WRITE(6,*) ' Enter filter parameter lambda. Try 5e-33.'
READ(5,*) LAMBDA
WRITE(6,*) ' '
WRITE(6,*) ' Enter time zero in ps. Try -200 ps.'
READ(5,*) T_ZERO
T_ZERO=T_ZERO*1E-12
WRITE(6,*) ' '
WRITE(6,*) ' FREQUENCY WINDOW'
WRITE(6,*) ' '
CALL MENUE
```

```
      READ(5,3000) FWIN
      WRITE(6,*) ' '
      WRITE(6,*) ' Enter cutoff frequency in GHz. Try 16 GHZ.'
      READ(5,*) F_CUTOFF
      F_CUTOFF=F_CUTOFF*1E9
      WRITE(6,*) ' '
      WRITE(6,*) ' Enter number of frequency points M.'
      READ(5,3000) M
      WRITE(6,*) ' '
      WRITE(6,*) ' Enter time increment for output kernel in ps. '
      READ(5,*) T_INC
      T_INC=T_INC*1E-12
      WRITE(6,*) ' '
      WRITE(6,*) ' Enter number of time points.'
      READ(5,3000) NT
      WRITE(6,*) ' '
      WRITE(6,*) ' Enter time zero for output kernel in ps.'
      READ(5,*) TZ
      TZ=TZ*1E-12
      WRITE(6,*) ' '
      WRITE(6,*) ' Enter number of iterations. 0,1,2,...'
      READ(5,3000) ITER
      WRITE(6,*) ' '
C
C End indata
C
      F_INC=F_CUTOFF/FLOAT(M-1)
      T_FRONT=0.0
      T_REAR=0.0
      NF=0
      NR=0
C
      IF (ITER.GT.1) THEN
        CALL CHOOSE_L(NT,L2)
        L=NT
        L2=L*2
        F_INC=1.0/(T_INC*FLOAT(L))
        M=INT(F_CUTOFF/F_INC)
        IF (L.LT.(2*M)) THEN
          WRITE(6,*) ' L too small! L= ',L,' M= ',M
          WRITE(6,*) ' '
          CALL EXIT
        END IF
        WRITE(6,*) 'Enter object front time in ps.'
        READ(5,*) T_FRONT
```

```
        T_FRONT=T_FRONT*1E-12
        WRITE(6,*) ' '
        WRITE(6,*) 'Enter object rear time in ps.'
        READ(5,*) T_REAR
        T_REAR=T_REAR*1E-12
        WRITE(6,*) ' '
        NF=INT(T_FRONT/T_INC)+1
        NR=INT(T_REAR/T_INC)+2
         DO K=1,L2
           DATA(K)=0.0
         END DO
      END IF
C
      FPR='hi'
      ID=IDY//IDX
      CALL OOUTFN(FPR,ID,9)
C
      ID=IDY//IDX
      WRITE(9,*) '   HISTORY FILE for tk',ID
      WRITE(9,*) ' '
      WRITE(9,*) ' PROGRAM DECONV.'
      WRITE(9,*) ' '
      WRITE(9,*) 'N_HAN=',N_HAN,'  PARAMETER=',NMAX
      WRITE(9,*) ' '
      WRITE(9,*) ' '
      ID=IDY//'00'
      WRITE(9,*) 'KERNEL tk',ID
      WRITE(9,*) ' '
      WRITE(9,*) 'KERNEL WINDOW           =',YWIN
      WRITE(9,*) 'Baseline adjustment     =',BA_K,'   (Y=1,N=0)'
      WRITE(9,*) 'Size of zero padding    =',K_ZERO
      WRITE(9,*) 'Change in sampling rate =',K_STEP
      WRITE(9,*) ' '
      WRITE(9,*) ' '
      WRITE(9,*) 'PULSE tx',IDX
      WRITE(9,*) ' '
      WRITE(9,*) 'PULSE WINDOW            =',XWIN
      WRITE(9,*) 'Baseline adjustment     =',BA_P,'  (Y=1,N=0)'
      WRITE(9,*) 'Size of zero padding    =',P_ZERO
      WRITE(9,*) 'Change in sampling rate =',P_STEP
      WRITE(9,*) ' '
      WRITE(9,*) ' '
      WRITE(9,*) 'DECONVOLUTION PARAMETER'
      WRITE(9,*) ' '
      WRITE(9,*) 'Filter parameter lambda =',LAMBDA
```

```
      WRITE(9,*) 'Pulse time zero in ps   =',T_ZERO*1E12
      WRITE(9,*) ' '
      WRITE(9,*) ' '
      WRITE(9,*) 'OUTPUT KERNEL PARAMETER'
      WRITE(9,*) ' '
      WRITE(9,*) 'Kernel time zero in ps     =',TZ*1E12
      WRITE(9,*) 'Time increment in ps       =',T_INC*1E12
      WRITE(9,*) 'Number of time points      =',NT
      WRITE(9,*) 'Time trace in ps           =',NT*T_INC*1E12
      WRITE(9,*) ' '
C
      IF (ITER.GT.1) THEN
      WRITE(9,*) 'Front object time in ps    =',T_FRONT*1E12
      WRITE(9,*) 'Rear object time in ps     =',T_REAR*1E12
      WRITE(9,*) 'NF =',NF,' NR =',NR
      WRITE(9,*) 'Object size in ps          =',
     +(NR-NF)*T_INC*1E12
      WRITE(9,*) 'New freq. cutoff in GHz    =',
     +L/2*F_INC*1E-9
      END IF
C
      WRITE(9,*) 'FREQUENCY WINDOW           =',FWIN
      WRITE(9,*) 'Cutoff Frequency in GHz    =',F_CUTOFF*1E-9
      WRITE(9,*) 'Number of freq. points (M) =',M
      WRITE(9,*) 'Frequency increment in MHz =',F_INC*1E-6
      WRITE(9,*) 'Number of iterations       =',ITER
      WRITE(9,*) ' '
C
C KERNEL
C
      CALL SCREENCLEAR
C
      WRITE(6,*) ' Running KERNEL ......'
      WRITE(6,*) ' '
      FPR='tk'
      ID=IDY//'00'
      CALL READ_FILE(W,N2,FPR,ID)
      CALL SORT(TR,TI,T_INC1,W,N2)
      N=N2/2
      IF (BA_K.EQ.1) CALL BASELINE(TR,N)
      IF (K_STEP.NE.0) CALL S_RATE(TR,TI,N,T_INC1,K_STEP)
      IF (K_ZERO.NE.0) CALL ZERO_PAD(TR,TI,N,K_ZERO)
      SYM=0
      CALL WINDOW(TR,TI,N,YWIN,SYM)
C
```

```
      CALL CZT(TR,TI,N,M,0.0,T_INC1,0.0,F_INC,-1)
C
       DO K=1,M
         FYR(K)=TR(K)
         FYI(K)=TI(K)
       END DO
      CALL AMP_DB(FYR,FYI,AMP,DB,M)
      FPR='fk'
      CALL WRITE5_FILE(F_INC,AMP,DB,FYR,FYI,M,FPR,ID)
C
C End KERNEL
C
C PULSE
C
      WRITE(6,*) ' Running PULSE ......'
      WRITE(6,*) ' '
      FPR='tx'
      ID=IDX
      CALL READ_FILE(W,N2,FPR,ID)
      CALL SORT(TR,TI,T_INC2,W,N2)
      N=N2/2
      IF (BA_P.EQ.1) CALL BASELINE(TR,N)
      IF (P_STEP.NE.0) CALL S_RATE(TR,TI,N,T_INC2,P_STEP)
      IF (P_ZERO.NE.0) CALL ZERO_PAD(TR,TI,N,P_ZERO)
      SYM=0
      CALL WINDOW(TR,TI,N,XWIN,SYM)
C
      CALL CZT(TR,TI,N,M,T_ZERO,T_INC2,0.0,F_INC,-1)
C
      CALL AMP_DB(TR,TI,AMP,DB,M)
      FPR='fx'
      CALL WRITE5_FILE(F_INC,AMP,DB,TR,TI,M,FPR,ID)
C
C End PULSE
C
      WRITE(6,*) ' Running DECON ......'
      WRITE(6,*) ' '
      CALL DECON(TR,TI,FYR,FYI,M,FR,FI,LAMBDA)
      CALL AMP_DB(FR,FI,AMP,DB,M)
      FPR='fr'
      ID=IDY//IDX
      CALL WRITE5_FILE(F_INC,AMP,DB,FR,FI,M,FPR,ID)
C
      IF (ITER.GT.1) GOTO 200
C
```

```
      WRITE(6,*) 'M= ',M,' F_CUTOFF in GHZ =',F_CUTOFF*1E-9
      WRITE(6,*) ' '
      WRITE(6,*) ' Enter new value on M, less than previous one.'
      READ(5,3000) M
      F_CUTOFF=(M-1)*F_INC
      WRITE(9,*) 'New value on M, less than previous one   =',M
      WRITE(9,*) 'New frequency cutoff in GHZ              =',
     +F_CUTOFF*1E-9
      WRITE(6,*) ' '
      WRITE(6,*) 'New frequency cutoff in GHZ =',F_CUTOFF*1E-9
      WRITE(9,*) ' '
      WRITE(6,*) ' '
C
      WRITE(6,*) ' Running ICZT ......'
      WRITE(6,*) ' '
C
      K=1
      KK=M
      DO KKK=1,M-1
        TR(K)=FR(KK)
        TI(K)=-FI(KK)
        K=K+1
        KK=KK-1
      END DO
      TR(M)=FR(1)
      TI(M)=0.0
      K=M+1
      KK=2
      DO KKK=1,M-2
        TR(K)=FR(KK)
        TI(K)=FI(KK)
        K=K+1
        KK=KK+1
      END DO
C
      F_ZERO=-F_CUTOFF
      MM=2*M-2
      SYM=0
      CALL WINDOW(TR,TI,MM,FWIN,SYM)
C
      CALL CZT(TR,TI,MM,NT,TZ,T_INC,F_ZERO,F_INC,1)
C
      FPR='tk'
      CALL AMP_DB(TR,TI,AMP,DB,NT)
      CALL WRITE5_FILE(T_INC,TR,TI,AMP,DB,NT,FPR,ID)
```

```
C
      GOTO 999
C
200   CONTINUE
C
      WRITE(6,*) ' Running ITER ',ITER,' times  ......'
      WRITE(6,*) ' '
C
      E(0)=0.0
      DATA(1)=FR(1)
      DATA(2)=FI(1)
      KR=3
      KI=4
      KRF=L2-1
      KIF=L2
      DO K=2,M
        DATA(KR)=FR(K)
        DATA(KI)=FI(K)
        DATA(KRF)=FR(K)
        DATA(KIF)=-FI(K)
        KR=KR+2
        KI=KI+2
        KRF=KRF-2
        KIF=KIF-2
      END DO
C
      DO I=1,ITER
C
      WRITE(6,*) 'ITER=',I
C
      CALL FOUR1(DATA,L,1)
C
      DO K=1,L2
        DATA(K)=DATA(K)*F_INC
      END DO
C
      E(I)=0.0
      DO K=1,2*NF-2
        E(I)=E(I)+DATA(K)*DATA(K)
      END DO
      DO K=2*NR+1,L2
        E(I)=E(I)+DATA(K)*DATA(K)
      END DO
C
      IF (I.EQ.ITER) GOTO 300
```

```
C
      DO K=1,2*NF-2
        DATA(K)=0.0
      END DO
      DO K=2*NR+1,L2
        DATA(K)=0.0
      END DO
C
C     DO K=2*NF,2*NR,2
C       DATA(K)=0.0
C     END DO
C
      CALL FOUR1(DATA,L,-1)
C
      DO K=1,L2
        DATA(K)=DATA(K)*T_INC
      END DO
C
      DATA(1)=FR(1)
      DATA(2)=FI(1)
      KR=3
      KI=4
      KRF=L2-1
      KIF=L2
      DO K=2,M
        DATA(KR)=FR(K)
        DATA(KI)=FI(K)
        DATA(KRF)=FR(K)
        DATA(KIF)=-FI(K)
        KR=KR+2
        KI=KI+2
        KRF=KRF-2
        KIF=KIF-2
      END DO
C
      IF (I.EQ.(ITER-1)) THEN
         KR=1
         KI=2
         MM=L/2+1
        DO K=1,L
          FR(K)=DATA(KR)
          FI(K)=DATA(KI)
          KR=KR+2
          KI=KI+2
        END DO
```

```
          KK=MM
          DO K=1,L/2
            TR(K)=FR(KK)
            TI(K)=FI(KK)
            KK=KK+1
          END DO
          KK=MM
          DO K=1,L/2
            TR(KK)=FR(K)
            TI(KK)=FI(K)
            KK=KK+1
          END DO

          SYM=0
          FPR='fi'
          ID=IDY//IDX
          CALL AMP_DB(FR,FI,AMP,DB,MM)
          CALL WRITE5_FILE(F_INC,AMP,DB,FR,FI,MM,FPR,ID)
          CALL WINDOW(TR,TI,L,FWIN,SYM)
          KR=1
          KI=2
          DO K=L/2+1,L
            DATA(KR)=TR(K)
            DATA(KI)=TI(K)
            KR=KR+2
            KI=KI+2
          END DO
          DO K=1,L/2
            DATA(KR)=TR(K)
            DATA(KI)=TI(K)
            KR=KR+2
            KI=KI+2
          END DO
          FPR='fu'
          ID='0000'
          CALL WRITE3_FILE(F_INC,DATA,L,FPR,ID)
        END IF
C
        END DO
C
300     CONTINUE
        FPR='tk'
        ID=IDY//IDX
        CALL WRITE3_FILE(T_INC,DATA,L2,FPR,ID)
        FPR='en'
```

```
      T_INC1=1.0
      K=ITER+1
      CALL WRITE2_FILE(T_INC1,E,K,FPR,ID)
C
999   CONTINUE
      WRITE(9,*) ' '
      WRITE(9,*) ' '
      WRITE(9,*) ' '
      WRITE(9,*) '   File used in this program.  File created in',
     +' this program'
      WRITE(9,*) ' '
      WRITE(9,*) '   tk',IDY//'00','                    hi',
     +IDY//IDX
      WRITE(9,*) '   tx',IDX,'                        fk',
     +IDY//'00'
      WRITE(9,*) '                                  fx',IDX
      WRITE(9,*) '                                  fr',
     +IDY//IDX
      WRITE(9,*) '                                  tk',
     +IDY//IDX
      WRITE(9,*) '                                  en',
     +IDY//IDX
      CLOSE(9)
      WRITE(6,*) ' '
      WRITE(6,*) ' READY! Read file hi',ID
      WRITE(6,*) ' '
C
      END
```

```
      SUBROUTINE FOUR1(DATA,NN,ISIGN)
C
C
C    NN MUST be an integer power of 2 (this is not checked for!).
C
C*********************************************
C
C  Replaces DATA by its discrete Fourier transform, if ISIGN
C  is input as 1; or replaces DATA by NN times its inverse
C  discrete Fourier transform, if ISIGN is input as -1.
C  DATA is a complex array of length NN or, equivalently, a
C  real array of length 2*NN.
C
C*********************************************
C
C
      REAL*8     WR,WI,WPR,WPI,WTEMP,THETA
      DIMENSION  DATA(2*NN)
      N=2*NN
      J=1
      DO I=1,N,2
        IF (J.GT.I) THEN
          TEMPR=DATA(J)
          TEMPI=DATA(J+1)
          DATA(J)=DATA(I)
          DATA(J+1)=DATA(I+1)
          DATA(I)=TEMPR
          DATA(I+1)=TEMPI
       END IF
      M=N/2
1       IF ((M.GE.2).AND.(J.GT.M)) THEN
          J=J-M
          M=M/2
         GOTO 1
        END IF
      J=J+M
      END DO
      MMAX=2
2     IF (N.GT.MMAX) THEN
        ISTEP=2*MMAX
        THETA=6.28318530717959D0/(ISIGN*MMAX)
        WPR=-2.D0*DSIN(0.5D0*THETA)**2
        WPI=DSIN(THETA)
        WR=1.D0
        WI=0.D0
```

```
DO M=1,MMAX,2
  DO I=M,N,ISTEP
    J=I+MMAX
    TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
    TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
    DATA(J)=DATA(I)-TEMPR
    DATA(J+1)=DATA(I+1)-TEMPI
    DATA(I)=DATA(I)+TEMPR
    DATA(I+1)=DATA(I+1)+TEMPI
  END DO
 WTEMP=WR
 WR=WR*WPR-WI*WPI+WR
 WI=WI*WPR+WTEMP*WPI+WI
END DO
MMAX=ISTEP
GOTO 2
END IF
RETURN
END
```

```
      SUBROUTINE MENUE
C
       COMMON /INTID/ N_HAN
       INTEGER    N_HAN
C
       WRITE(6,*) '        WINDOW MENUE'
       WRITE(6,*) ' '
       WRITE(6,*) ' '
       WRITE(6,*) ' 0   RECTANGULAR     WEIGHTING'
       WRITE(6,*) ' 1   HANNING/TUKEY   WEIGHTING'
       WRITE(6,*) ' 2   HAMMING         WEIGHTING'
       WRITE(6,*) ' 3   BLACKMAN        WEIGHTING'
       WRITE(6,*) ' 4   KAISER-BESSEL   WEIGHTING'
       WRITE(6,*) ' 5   BLACKMAN-HARRIS WEIGHTING'
       WRITE(6,*) ' 6   FLAT TOP        WEIGHTING'
       WRITE(6,*) ' 7  ',N_HAN,' points leading and trailing
      +HANNING'
       WRITE(6,*) ' '
       WRITE(6,*) ' YOUR CHOICE ? '
       WRITE(6,*) ' '
       RETURN
       END
```

```
      SUBROUTINE WINDOW(XREAL,XIMAG,N,W,SYM)
C
      IMPLICIT NONE
C
      REAL       XREAL(0:*),XIMAG(0:*),WEIGHT
      REAL       PI,PI2,AR,ARG,ARG2,ARG3,ARG4
      REAL       A0,A1,A2,A3,A4
      INTEGER    N,N_HAN,W,SYM,I
      COMMON /INTID/ N_HAN
C
      PI=4.0*ATAN(1.0)
      PI2=8.0*ATAN(1.0)
      A0=0.0
      A1=0.0
      A2=0.0
      A3=0.0
      A4=0.0
C
      IF (W.EQ.0) THEN
        WRITE(6,*) ' RECTANGULAR WEIGHTING'
        WRITE(6,*) ' '
        RETURN
      ELSEIF (W.EQ.1) THEN
        WRITE(6,*) ' HANNING/TUKEY WEIGHTING'
        WRITE(6,*) ' '
        A0=0.5
        A1=0.5
      ELSEIF (W.EQ.2) THEN
        WRITE(6,*) ' HAMMING WEIGHTING'
        WRITE(6,*) ' '
        A0=0.54
        A1=0.46
      ELSEIF (W.EQ.3) THEN
        WRITE(6,*) ' BLACKMAN WEIGHTING'
        WRITE(6,*) ' '
        A0=0.42
        A1=0.5
        A2=0.08
      ELSEIF (W.EQ.4) THEN
        WRITE(6,*) ' KAISER-BESSEL WEIGHTING'
        WRITE(6,*) ' '
        A0=0.402083
        A1=0.498583
        A2=0.098108
        A3=0.001226
```

```
      ELSEIF (W.EQ.5) THEN
        WRITE(6,*) ' BLACKMAN-HARRIS'
        WRITE(6,*) ' '
        A0=0.35875
        A1=0.48829
        A2=0.14128
        A3=0.01168
      ELSEIF (W.EQ.6) THEN
        WRITE(6,*) ' FLAT TOP WEIGHTING'
        WRITE(6,*) ' '
        A0=0.215508
        A1=0.415930
        A2=0.278005
        A3=0.083617
        A4=0.006939
      ELSEIF (W.EQ.7) THEN
        WRITE(6,*)  N_HAN, ' points leading and trailing HANNING'
        WRITE(6,*) ' '
        GOTO 20
      ELSE
        WRITE(6,*)  'No such window. W= ',W
        WRITE(6,*) ' '
        CALL EXIT
      ENDIF
C
      IF (SYM.EQ.0) THEN
        ARG=PI2/FLOAT(N)
        ARG2=2.0*ARG
        ARG3=3.0*ARG
        ARG4=4.0*ARG
       DO I=0,N-1
         WEIGHT=A0-A1*COS(ARG*I)+A2*COS(ARG2*I)
         WEIGHT=WEIGHT-A3*COS(ARG3*I)+A4*COS(ARG4*I)
         XREAL(I)=XREAL(I)*WEIGHT
         XIMAG(I)=XIMAG(I)*WEIGHT
       ENDDO
C
      RETURN
C
      ELSEIF (SYM.EQ.1) THEN
        ARG=PI/FLOAT(N-1)
        ARG2=2.0*ARG
        ARG3=3.0*ARG
        ARG4=4.0*ARG
       DO I=0,N-1
```

```
          WEIGHT=A0+A1*COS(ARG*I)+A2*COS(ARG2*I)
          WEIGHT=WEIGHT+A3*COS(ARG3*I)+A4*COS(ARG4*I)
          XREAL(I)=XREAL(I)*WEIGHT
          XIMAG(I)=XIMAG(I)*WEIGHT
         END DO
C
      RETURN
      ELSE
       WRITE(6,*) 'Error in SYM. SYM= ',SYM
       WRITE(6,*) ' '
       CALL EXIT
      END IF
20    IF (SYM.NE.0) THEN
       WRITE(6,*) 'Error in SYM. SYM= ',SYM
       WRITE(6,*) ' '
       CALL EXIT
      ELSE
      ARG=PI2/FLOAT(2*N_HAN-1)
       DO I=0,N_HAN-1
          WEIGHT=0.5-0.5*COS(ARG*I)
          XREAL(I)=XREAL(I)*WEIGHT
          XIMAG(I)=XIMAG(I)*WEIGHT
          XREAL(N-1-I)=XREAL(N-1-I)*WEIGHT
          XIMAG(N-1-I)=XIMAG(N-1-I)*WEIGHT
       END DO
C
      RETURN
      END IF
C
      END
```

```
      SUBROUTINE CZT(XR,XI,N,M,T_ZERO,T_INC,F_ZERO,F_INC,ISN)
C
C*********************************************
C
C     CZT: ISN=-1   TIME TO FREQUENCY
C    ICZT: ISN=+1   FREQUENCY TO TIME
C
C   Ref.: "The Chirp z-Transform Algorithm". IEEE Trans. Audio C
C           Electroacoust.,vol. AU-17,pp.86-92,June 1969.
C         "Conversion of Frequency-Domain Data to the Time
C          Domain". Proc. IEEE,vol.74,no.1,January 1986.
C
C*********************************************
      IMPLICIT NONE
C
      INTEGER      NMAX
      PARAMETER    (NMAX=8191)
C
      REAL         XR(0:NMAX),XI(0:NMAX)
      REAL         T_ZERO,T_INC,F_ZERO,F_INC
      REAL         TR,TI
      REAL         RV(0:NMAX),IV(0:NMAX)
      REAL         PI,PI2,ARG,ALFA,BETA,GAMMA,DELTA,NORM,C,S
      INTEGER      N,M,L,NU,I,ISN
C
      IF (ISN.EQ.1) THEN
        S=T_ZERO
        T_ZERO=F_ZERO
        F_ZERO=-S
        S=T_INC
        T_INC=F_INC
        F_INC=-S
      END IF
C
      PI=4.0*ATAN(1.0)
      PI2=8.0*ATAN(1.0)
      ALFA=F_INC*T_INC*PI
      BETA=F_ZERO*T_INC*PI2
      GAMMA=F_ZERO*T_ZERO*PI2
      DELTA=F_INC*T_ZERO*PI2
C
C Step 1
C
      L=N+M-1
      CALL CHOOSE_L(L,NU)
```

```
C
C End Step 1
C
C Step 2
C
      DO I=0,N-1
         ARG=ALFA*I*I+BETA*I
         C=COS(ARG)
         S=SIN(ARG)
         TR=C*XR(I)+S*XI(I)
         TI=C*XI(I)-S*XR(I)
         XR(I)=TR
         XI(I)=TI
      END DO
C
      DO I=N,L-1
         XR(I)=0.0
         XI(I)=0.0
      END DO
C
C End Step 2
C
C Step 3
C
      CALL DFT(XR,XI,L,NU,-1)
C
C End Step 3
C
C Step 4
C
      RV(0)=1.0
      IV(0)=0.0
C
      IF (M.GE.N) THEN

         DO I=1,M-1
           ARG=ALFA*I*I
           RV(I)=COS(ARG)
           IV(I)=SIN(ARG)
         END DO

         DO I=1,N-1
           RV(L-I)=RV(I)
           IV(L-I)=IV(I)
         END DO
```

```
      ELSEIF (L.GE.(2*N)) THEN

         DO I=1,N-1
           ARG=ALFA*I*I
           RV(I)=COS(ARG)
           IV(I)=SIN(ARG)
           RV(L-I)=RV(I)
           IV(L-I)=IV(I)
         END DO

       ELSE

         DO I=1,L/2-1
           ARG=ALFA*I*I
           RV(I)=COS(ARG)
           IV(I)=SIN(ARG)
           RV(L-I)=RV(I)
           IV(L-I)=IV(I)
         END DO

         DO I=L-N+1,L/2
           ARG=ALFA*(L-I)*(L-I)
           RV(I)=COS(ARG)
           IV(I)=SIN(ARG)
         END DO

       END IF
C
      IF (L.NE.(M+N-1)) THEN
C
         DO I=M,L-N
           RV(I)=0.0
           IV(I)=0.0
         END DO
C
      END IF
C
C End Step 4
C
C Step 5
C
      CALL DFT(RV,IV,L,NU,-1)
C
C End Step 5
```

```
C
C Step 6
C
      DO I=0,L-1
       TR=RV(I)*XR(I)-IV(I)*XI(I)
       TI=IV(I)*XR(I)+RV(I)*XI(I)
       XR(I)=TR
       XI(I)=TI
      END DO
C
C End Step 6
C
C Step 7
C
      CALL DFT(XR,XI,L,NU,1)
C
C End Step 7
C
C Step 8
C
      NORM=T_INC/FLOAT(L)
C
      DO I=0,M-1
       ARG=ALFA*I*I+GAMMA+DELTA*I
       C=COS(ARG)
       S=SIN(ARG)
       TR=C*XR(I)+S*XI(I)
       TI=C*XI(I)-S*XR(I)
       XR(I)=TR*NORM
       XI(I)=TI*NORM
      END DO
C
C End Step 8
C
      IF (ISN.EQ.1) THEN
        S=T_ZERO
        T_ZERO=-F_ZERO
        F_ZERO=S
        S=T_INC
        T_INC=-F_INC
        F_INC=S
      END IF
C
      RETURN
      END
```

```
      SUBROUTINE DECON(XR,XI,YR,YI,N,ZR,ZI,LAMBDA)
C
      IMPLICIT NONE
C
      REAL         XR(1:N),XI(1:N),YR(1:N),YI(1:N)
      REAL         ZR(1:N),ZI(1:N)
      REAL         DEN
      REAL         LAMBDA
      INTEGER      N,I
C
      DO I=1,N
        DEN=1.0/(XR(I)*XR(I)+XI(I)*XI(I)+LAMBDA*I*I*I*I)
        ZR(I)=(XR(I)*YR(I)+XI(I)*YI(I))*DEN
        ZI(I)=(XR(I)*YI(I)-XI(I)*YR(I))*DEN
      END DO
C
      RETURN
      END
```

```
      SUBROUTINE DFT(XREAL,XIMAG,N,NU,ISN)
C
C
C*********************************************
C
C    Ref.: E. Oran Brigham, "The Fast Fourier Transform"
C           Prentice-Hall, Inc. 1974
C
C      DFT: ISN=-1  (TIME TO FREQUENCY)
C     IDFT: ISN=+1  (FREQUENCY TO TIME)
C
C*********************************************
C
      IMPLICIT NONE
C
      INTEGER     NMAX
      PARAMETER   (NMAX=8192)
C
      REAL        XREAL(1:NMAX),XIMAG(1:NMAX)
      INTEGER     N,NU,ISN
      REAL        TREAL,TIMAG,P,ARG,C,S,PI2
      INTEGER     N2,NU1,I,L,K,K1,K1N2,IBITR
C
      PI2=8.0*ATAN(1.0)
      ISN=-ISN
      N2=N/2
      NU1=NU-1
      K=0
      DO 100 L=1,NU
 102  DO 101 I=1,N2
      P=IBITR(K/2**NU1,NU)
      ARG=PI2*P/FLOAT(N)
      C=COS(ARG)
      S=SIN(ARG)*ISN
      K1=K+1
      K1N2=K1+N2
      TREAL=XREAL(K1N2)*C+XIMAG(K1N2)*S
      TIMAG=XIMAG(K1N2)*C-XREAL(K1N2)*S
      XREAL(K1N2)=XREAL(K1)-TREAL
      XIMAG(K1N2)=XIMAG(K1)-TIMAG
      XREAL(K1)=XREAL(K1)+TREAL
      XIMAG(K1)=XIMAG(K1)+TIMAG
 101  K=K+1
      K=K+N2
      IF(K.LT.N) GOTO 102
```

```
      K=0
      NU1=NU1-1
100   N2=N2/2
      DO 103 K=1,N
      I=IBITR(K-1,NU)+1
      IF(I.LE.K) GOTO 103
      TREAL=XREAL(K)
      TIMAG=XIMAG(K)
      XREAL(K)=XREAL(I)
      XIMAG(K)=XIMAG(I)
      XREAL(I)=TREAL
      XIMAG(I)=TIMAG
103   CONTINUE
      ISN=-ISN
      RETURN
      END
```

```
      INTEGER FUNCTION IBITR(J,NU)
C
      IMPLICIT NONE
C
      INTEGER     J,NU,I,J1,J2
C
      J1=J
      IBITR=0
      DO 200 I=1,NU
      J2=J1/2
      IBITR=IBITR*2+(J1-2*J2)
 200  J1=J2
      RETURN
      END
```

```
      SUBROUTINE CHOOSE_L(L,NU)
C
      IMPLICIT NONE
C
      INTEGER  L,NU
C
      IF (L.GT.8192) THEN
       WRITE(6,*) 'L is too big. L= ' , L
       CALL EXIT
      END IF
      IF (L.GT.4096) THEN
       L=8192
       NU=13
       RETURN
      END IF
      IF (L.GT.2048) THEN
       L=4096
       NU=12
       RETURN
      END IF
      IF (L.GT.1024) THEN
       L=2048
       NU=11
       RETURN
      END IF
      IF (L.GT.512) THEN
       L=1024
       NU=10
       RETURN
      END IF
      IF (L.GT.256) THEN
       L=512
       NU=9
       RETURN
      END IF
      IF (L.GT.128) THEN
       L=256
       NU=8
       RETURN
      END IF
      IF (L.GT.64) THEN
       L=128
       NU=7
       RETURN
      END IF
```

```
IF (L.GT.32) THEN
 L=64
 NU=6
 RETURN
END IF
IF (L.GT.16) THEN
 L=32
 NU=5
 RETURN
END IF
IF (L.GT.8) THEN
 L=16
 NU=4
 RETURN
END IF
IF (L.GT.4) THEN
 L=8
 NU=3
 RETURN
END IF
IF (L.GT.2) THEN
 L=4
 NU=2
 RETURN
END IF
IF (L.GT.1) THEN
 L=2
 NU=1
 RETURN
END IF
WRITE(6,*) 'L is too small. L= ' , L
CALL EXIT
END
```

```
      SUBROUTINE SCREENCLEAR
C
      IMPLICIT NONE
C
      WRITE(6,*) CHAR(27)//'H'//CHAR(27)//'J'
      RETURN
      END
```

```
      SUBROUTINE READ_FILE(W,N,FPR,ID)
C
C     This subroutine reads the data used in this program
C
      IMPLICIT NONE
C
      REAL          W(0:*)
      INTEGER       N,I,IOS
      CHARACTER*2   FPR
      CHARACTER*4   ID
      WRITE(6,*) ' '
C
      CALL OINFN(FPR,ID,20)
C
1000  FORMAT(4E13.5,4E13.5)
      I = 0
      READ(20,1000,IOSTAT=IOS) W(I),W(I+1)
10    IF(IOS.EQ.0) THEN
 I=I+2
         READ(20,1000,IOSTAT=IOS) W(I),W(I+1)
 GO TO 10
      END IF
      CLOSE(20)
      N=I
      RETURN
      END
```

```
      SUBROUTINE OINFN(FPR,ID,LUN)
C
      IMPLICIT NONE
C
      CHARACTER*6   FLNAME
      CHARACTER*4   ID
      CHARACTER*2   FPR
      INTEGER       LUN
C
      FLNAME=FPR//ID
      OPEN (UNIT=LUN,FILE=FLNAME,ERR=150,STATUS='OLD')
      GO TO 20
150   WRITE(6,*) 'Error in trying to open the input file:',
      +FLNAME
      CALL EXIT
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SORT(XREAL,XIMAG,T_INC,W,N2)
C
      IMPLICIT NONE
C
      REAL         XREAL(0:*),XIMAG(0:*),W(0:*),T_INC
      INTEGER      N2,I
C
      T_INC=W(2)
      DO I=1,N2-1,2
       XREAL((I-1)/2)=W(I)
       XIMAG((I-1)/2)=0.0
      END DO
C
      RETURN
      END
```

```
      SUBROUTINE S_RATE(XR,XI,N,T_INC,STEP)
C
C     Change of sampling rate.
C
      IMPLICIT NONE
C
      REAL       XR(1:*),XI(1:*),T_INC
      INTEGER    N,STEP,I,K
C
      K=0
      DO I=1,N,STEP
        K=K+1
        XR(K)=XR(I)
        XI(K)=XI(I)
      END DO
C
      T_INC=T_INC*STEP
      N=K
C
      RETURN
      END
```

```
      SUBROUTINE ZERO_PAD(XR,XI,N,NMAX)
C
      IMPLICIT NONE
C
      REAL        XR(1:*),XI(1:*)
      INTEGER     N,NMAX,I
C
      DO I=N+1,NMAX
        XR(I)=0.0
        XI(I)=0.0
      END DO
C
      N=NMAX
      RETURN
      END
```

```
      SUBROUTINE BASELINE(W,N)
C
C     Adjust for non-zero baseline.

C
      IMPLICIT NONE
C
      REAL         W(0:N-1),K
      INTEGER      N,I
C
      K=0.0
      DO I=0,24
        K=K+W(I)
      END DO
      DO I=N-25,N-1
        K=K+W(I)
      END DO
      K=K/50.0
      WRITE(6,*) ' Enter baseline offset. Try ',K
      WRITE(6,*) ' '
      READ(5,*) K
C
      DO I=0,N-1
        W(I)=W(I)-K
      END DO
C
      RETURN
      END
```

```
      SUBROUTINE AMP_PHASE(RE,IM,AMP,PHASE,N)
C
      IMPLICIT NONE
C
      REAL       RE(1:N),IM(1:N),AMP(1:N),PHASE(1:N)
      INTEGER    N,I
C
      DO I=1,N
        AMP(I)=SQRT(RE(I)*RE(I)+IM(I)*IM(I))
        IF (RE(I).LE.(573.0*IM(I))) THEN
         PHASE(I)=2.0*ATAN(1.0)
        ELSE
         PHASE(I)=ATAN(IM(I)/RE(I))
        END IF
      END DO
C
      RETURN
      END
```

```
      SUBROUTINE AMP_DB(RE,IM,AMP,DB,N)
C
      IMPLICIT NONE
C
      REAL       RE(1:N),IM(1:N),AMP(1:N),DB(1:N)
      INTEGER    N,I
C
      DO I=1,N
        AMP(I)=SQRT(RE(I)*RE(I)+IM(I)*IM(I))
        IF (AMP(I).LE.1E-36) THEN
         DB(I)=-720.0
        ELSE
         DB(I)=20.0*LOG10(AMP(I))
        END IF
      END DO
C
      RETURN
      END
```

```
      SUBROUTINE WRITE2_FILE(INC,Y,N,FPR,ID)
C
      IMPLICIT NONE
C
      REAL           Y(0:*),INC
      INTEGER        N,I
      CHARACTER*2    FPR
      CHARACTER*4    ID
C
1000  FORMAT(4E13.5,4E13.5)
C
      CALL OOUTFN(FPR,ID,20)
C
      DO I=0,N-1
         WRITE(20,1000)  INC*I,Y(I)
      END DO
C
      CLOSE(20)
C
      RETURN
      END
```

```
      SUBROUTINE WRITE3_FILE(INC,Y,N,FPR,ID)
C
      IMPLICIT NONE
C
      REAL           Y(0:*),INC
      INTEGER        N,I,K
      CHARACTER*2    FPR
      CHARACTER*4    ID
C
1000  FORMAT(4E13.5,4E13.5,4E13.5)
C
      CALL OOUTFN(FPR,ID,20)
C
      K=0
      DO I=0,N-2,2
         WRITE(20,1000)  INC*K,Y(I),Y(I+1)
         K=K+1
      END DO
C
      CLOSE(20)
C
      RETURN
      END
```

```
      SUBROUTINE WRITE5_FILE(INC,Y1,Y2,Y3,Y4,N,FPR,ID)
C
C     This subroutine writes data to the file FPR&ID
C            N .LE. *
C
      IMPLICIT NONE
C
      REAL          Y1(0:*),Y2(0:*),Y3(0:*),Y4(0:*),INC
      INTEGER       N,I
      CHARACTER*2   FPR
      CHARACTER*4   ID
C
1000  FORMAT(4E13.5,4E13.5,4E13.5,4E13.5,4E13.5)
C
      CALL OOUTFN(FPR,ID,20)
C
      DO I=0,N-1
         WRITE(20,1000) INC*I,Y1(I),Y2(I),Y3(I),Y4(I)
      END DO
C
      CLOSE(20)
C
      RETURN
      END
```

```
      SUBROUTINE OOUTFN(FPR,ID,LUN)
C
      IMPLICIT NONE
C
      CHARACTER*6   FLNAME
      CHARACTER*4   ID
      CHARACTER*2   FPR
      INTEGER       LUN
C
      FLNAME=FPR//ID
      OPEN (UNIT=LUN,FILE=FLNAME,ERR=150)
      RETURN
150   WRITE(6,*)'Error in trying to open the output file:',
      +FLNAME
      RETURN
      END
```

```
C Program INV
C
C*************************************************************C
C                                                             C
C     This program solves the inverse problem, assuming that the C
C     conductivity, SIG(X), is given.  The Riccati equation is   C
C     used to reconstruct the coefficients A and B.  Then the    C
C     relative permittivity, EPSR, is computed as a function of  C
C     depth, Z.                                               C
C                                                             C
C     (SL=l*c_0)                                              C
C                                                             C
C*************************************************************C
C
C
      IMPLICIT NONE
      REAL         RPLUS(0:4096),EPSRZ,SL
      INTEGER      N
      CHARACTER*4  ID
C
      CALL READ_DATA(N,RPLUS,EPSRZ,SL,ID)
      CALL RSOLVE(N,EPSRZ,SL,RPLUS,ID)
      STOP
      END
```

```
      SUBROUTINE READ_DATA(N,RPLUS,EPSRZ,SL,ID)
C
C     This subroutine reads the data used in this program
C
      IMPLICIT NONE
      REAL          RPLUS(0:4096),T,EPSRZ,SL,SIG1,L1,L
      INTEGER       N,I,IOS
      CHARACTER*2   FID
      CHARACTER*4   ID
      COMMON /SIGMA/ SIG1
C
1000  FORMAT(2A)
1001  FORMAT(2E13.5)
1002  FORMAT(E13.5)
C
      CALL SCREENCLEAR
      WRITE(6,*) '   <><><> Solve the inverse problem for',
     +          ' EPS(Z) <><><>'
      WRITE(6,*) ' '
      WRITE(6,*) 'Enter the input file id'
      WRITE(6,*) '2-digit # for synthetic data'
      WRITE(6,*) '4-digit # for experimental data'
      READ(5,1000) ID
C
      IF(LGE(ID,'0100')) THEN
         FID(1:1)=ID(1:1)
         FID(2:2)=ID(2:2)
         CALL OINFN(FID,'ce',20)
      ELSE
         CALL OINFN(ID,'cs',20)
      ENDIF
C
      READ(20,1002) EPSRZ
      READ(20,1002) L1
      READ(20,1002) SIG1
C     READ(20,1002) G1
      CALL CLOSE(20)
C
      CALL OINFN(ID,'tk',20)
C
      I = 0
      READ(20,1001,IOSTAT=IOS) T,RPLUS(I)
10    IF(IOS.EQ.0) THEN
         I=I+1
         READ(20,1001,IOSTAT=IOS) T,RPLUS(I)
```

```
         GO TO 10
      END IF
      N = I-1
      CALL CLOSE(20)
C
      L=T/2.
      DO I=0,N
         RPLUS(I)=RPLUS(I)*L
      END DO
C
      SL=L*3.E8
      RETURN
      END
```

```
      SUBROUTINE RSOLVE(N,EPSRZ,SL,ROLD,ID)
C
C     This subroutine inverts the reflection data assuming
C     the conductivity SIG known.
C
      IMPLICIT NONE
      REAL RNEW(0:4096),ROLD(0:4096)
      REAL Z(0:4096),EPSR(0:4096),A(0:4096),B(0:4096)
      REAL CONV(0:4096),E(0:4096)
      REAL C,C1,C2,C3,C4,C5,SUM,SUM2
      REAL CON,HSL,H,SIGN,RHS,AC,AN,RELER,EPSRZ,SL,SIG
      INTEGER I,J,K,N,JMAX
      CHARACTER*2 ID
C
1000  FORMAT(6E13.5)
C
      H = 1./N
      HSL = H*SL/SQRT(EPSRZ)
      CON = -SL/(EPSRZ*3.*8.854E-4)
C
      CALL OOUTFN(ID,'po',22)
C
      E(0) = 1.
      Z(0) = 0.
      EPSR(0) = EPSRZ/(E(0)**2)
      B(0) = CON*SIG(0.)
      A(0) = -4.*ROLD(0)+B(0)
      WRITE(22,1000) Z(0),EPSR(0),SIG(0.),0.,A(0),B(0)
C
C     Compute the convolution product of the RPLUS data
C
      CONV(0) = 0.
      DO J = 1,N-1
         SUM = 0.
         DO K = 1,J
            SUM = SUM + ROLD(K)*ROLD(J+1-K)
        END DO
        CONV(J) = SUM
      END DO
C
      DO  I = 1,N
C
C     Compute A(I) using Newton's method
C
         SIGN = SIG(Z(I-1) + HSL*E(I-1))
```

```
      C = CON*SIGN*(E(I-1)**2)
      C1 = C*EXP(-H*A(I-1))
      RHS = ROLD(1)*(1-H*B(I-1)/2.-H*H*ROLD(0)*(A(I-1)
     +               + B(I-1))/2.)
      C5 = C1*EXP(-H*A(I-1))
      AC = C5 - 4.*RHS/(1.+H*C5/2.)
      DO J = 1,10
         C2 = C1*EXP(-H*AC)
         AN = AC - (4.*RHS + (AN-C2)*(1.+H*C2/2.))/
     +          ((1.+H*C2)*(1.+H*C2/2.) - (AN - C2)*(H*H*C2)/2.)
         IF (AN.EQ.0.) RELER = ABS(AN-AC)
         IF (AN.NE.0.) RELER = ABS((AN-AC)/AN)
         IF (RELER.LT..5E-5)  GO TO 20
         AC = AN
      END DO
      WRITE(6,*) I,RELER
20    A(I) = AN
      B(I) = C1*EXP(-H*AN)
      RNEW(0) = -.25*(AN-B(I))
      E(I) = E(I-1)*EXP(-H*(A(I-1)+AN)/2.)
      Z(I) = Z(I-1) + HSL*(E(I)+E(I-1))/2.
C
      EPSR(I)= EPSRZ/(E(I)**2)
      WRITE(22,1000) Z(I), EPSR(I), SIG(Z(I)),H*I,A(I),B(I)
C
      IF (I.LT.N)  THEN
C
C     Compute RNEW array
C
         JMAX = N - I
         C1 = -H*H*(AN+B(I))/2.
         C2 = 1./(1.+H*B(I)/2. - C1*RNEW(0))
         C3 = -H*H*(A(I-1) + B(I-1))/2.
         C4 = 1. - H*B(I-1)/2.
         DO J = 1,JMAX
            SUM = CONV(J) + ROLD(0)*ROLD(J+1)
            SUM2 = 0.
            IF (J.GT.1)  THEN
               DO K = 1,J-1
                  SUM2 = SUM2 + RNEW(K)*RNEW(J-K)
               END DO
               CONV(J-1) = SUM2
            END IF
            RNEW(J) = C2*(C4*ROLD(J+1) + C3*SUM + C1*SUM2)
         END DO
```

```
C
            DO J = 0,JMAX
               ROLD(J) = RNEW(J)
            END DO
         END IF
      END DO
C
      CALL CLOSE(22)
      RETURN
      END
```

```
      FUNCTION SIG(X)
C
C     SIG(X) is the assumed conductivity
C
      IMPLICIT NONE
      REAL SIG,SIG1,X
      COMMON /SIGMA/ SIG1
C
      SIG = SIG1
      RETURN
      END
```

```
C Program INVGREEN
C*****************************************************************C
C                                                                 C
C     This program solves the inverse problem, assuming that the  C
C     conductivity, SIG(X), is given.  The Green function is      C
C     used to reconstruct the coefficients A and B.  Then the     C
C     relative permittivity, EPSR, is computed as a function of   C
C     depth, z.                                                   C
C     The normalization of time is SL/c.                          C
C                                                                 C
C     Input:                                                      C
C       1. File k##                                               C
C               N+1 values of Rplus(s)                            C
C                                                                 C
C       2. File a##                                               C
C               Relative permittivity at z = 0, SL, constant SIG  C
C                                                                 C
C     Output:                                                     C
C          File o##                                               C
C               N+1 values of z, epsr(z), sig(z), x(z), A(x), B(x)C
C                                                                 C
C*****************************************************************C
      IMPLICIT NONE
      REAL    RPLUS(0:1024)
      INTEGER N
C
      CALL READ_DATAG(N,RPLUS)
      CALL RSOLVE(N,RPLUS)
      STOP
      END
```

```
      SUBROUTINE READ_DATAG(N,RPLUS)
C*****************************************************************
C     This subroutine reads the data used in this program.
C     The factor CON is -1/eps_0/eps_r.
C     The factor HSL is l*h*c(0).
C     EPSRZ is the value of the relative permittivity at z=0.
C     SIG1 is the assumed constant value of the conductivity.
C*****************************************************************
      IMPLICIT NONE
      REAL               RPLUS(0:1024),EPSRZ,HSL,SIG1,X,CON,SL
      INTEGER            N,I,IOS
      CHARACTER          FID*2
      COMMON /SIGMA/     SIG1
      COMMON /CONSTANTS/ EPSRZ,HSL,CON
      COMMON /FILEID/    FID
C
1000  FORMAT(A)
C
      CALL SCREENCLEAR
      WRITE(6,*) '<><><> SOLVE THE INVERSE PROBLEM FOR',
     +           ' EPS(Z)   <><><>'
      WRITE(6,*) "<><><> THE GREEN FUNCTIONS TECHNIQUE',
     +           ' IS USED <><><>"
      WRITE(6,*) ' '
      WRITE(6,*) 'Enter the input ID #, a two digit number'
      READ(5,1000) FID
      CALL OINFN('k',FID,20)
      I=0
      READ(20,*,IOSTAT=IOS) X,RPLUS(I)
10    IF(IOS.EQ.0) THEN
         I=I+1
         READ(20,*,IOSTAT=IOS) X,RPLUS(I)
         GO TO 10
      END IF
      CALL CLOSE(20)
      N=I-1
      CALL OINFN('a',FID,20)
      READ(20,*) X,X
      READ(20,*) EPSRZ,SL,SIG1
      CALL CLOSE(20)
C
      HSL=SL/SQRT(EPSRZ)/N
      CON=-SL/(EPSRZ*3.*8.854E-4)
      RETURN
      END
```

```
      SUBROUTINE RSOLVE(N,R)
C*****************************************************************
C     This subroutine inverts the reflection data assuming
C     the conductivity SIG known.
C     The Green functions approach is used.
C*****************************************************************
      IMPLICIT NONE
      REAL            R(0:1024),G1(0:1024),G2(0:1024)
      REAL            H,A,B,AM,AP,Z,E
      INTEGER         N,I
      CHARACTER       FID*2
      COMMON /FILEID/ FID
C
      H=1./N
      CALL OOUTFN('o',FID,22)
      CALL INITIATE(N,R)
      DO I=1,N
         CALL NEWAB(I,H,A,B)
         CALL GREEN(I,N,A,B,AM,AP,G1,G2)
         CALL PROFILE(I,H,A,B,Z,E)
         CALL UPDATE(I,N,A,B,AM,AP,Z,E,G1,G2)
      END DO
      CALL CLOSE(22)
      RETURN
      END
```

```
      SUBROUTINE INITIATE(N,R)
C     ************************************************************
C     This subroutine initiates the inversion algorithm.
C     The field E(z)=c(0)/c(z).
C     The variable ATT=integral of B from 0 to x.
C     ************************************************************
      IMPLICIT NONE
      REAL               R(0:N),G1(0:1024),G2(0:1024)
      REAL               A,B,EPSRZ,CON,SIG,ATT,AP,AM,Z,E,HSL
      INTEGER            N,I
      COMMON /OLD/       ATT,A,B,AP,AM,Z,E
      COMMON /CONSTANTS/ EPSRZ,HSL,CON
      COMMON /OLDG/      G1,G2
C
1000  FORMAT(6E13.5)
C
      DO I=0,N
        G1(I)=0
        G2(I)=R(I)
      END DO
      Z=0.
      E=1.
      B=CON*SIG(0.)
      A=-4*R(0)+B
      AP=(A+B)/2.
      AM=(A-B)/2.
      ATT=1.
      WRITE(22,1000) Z,EPSRZ,SIG(0.),0.,A,B
      RETURN
      END
```

```
      SUBROUTINE NEWAB(I,H,A,B)
C*************************************************************
C     This subroutine computes the new A and B values at the new
C     station in terms of the old values of AP, AM, G1 and G2
C     at time s=x-h and s=x+h.
C*************************************************************
      IMPLICIT NONE
      REAL               G1OLD(0:1024),G2OLD(0:1024)
      REAL               A,B,APOLD,AMOLD,AOLD,BOLD,ZOLD,EOLD,ATT
      REAL               EPSRZ,HSL,CON,SIG,C1,F1,F2,F3
      REAL               AC,FA,FD,H,RELER
      INTEGER            I,J
      COMMON /OLD/       ATT,AOLD,BOLD,APOLD,AMOLD,ZOLD,EOLD
      COMMON /CONSTANTS/ EPSRZ,HSL,CON
      COMMON /OLDG/      G1OLD,G2OLD
C
      C1=CON*SIG(ZOLD+HSL*EOLD)*EOLD**2*EXP(-H*AOLD)
      B=C1*EXP(-H*AOLD)
      F1=G2OLD(1)+.5*H*AMOLD*G1OLD(1)
      F2=H*G1OLD(0)-.0625*H*H*(AOLD*AOLD-BOLD*BOLD)
      F3=-4*F1/ATT*EXP(-.5*H*BOLD)
      AC=B+F3*EXP(-.5*H*B)/(1+F2)
      DO J=1,10
         B=C1*EXP(-H*AC)
         FA=(AC-B)*(1-.0625*H*H*(AC*AC-B*B)+F2)-F3*EXP(-.5*H*B)
         FD=(1+H*B-.5*H*H*B*(AC-B))*(1-.0625*H*H*(AC*AC-B*B)+F2)
     &       -.125*H*H*(AC-B)*(AC+H*B*B)
         A=AC-FA/FD
         IF (A.EQ.0.) RELER=ABS(A-AC)
         IF (A.NE.0.) RELER=ABS((A-AC)/A)
         IF (RELER.LT..5E-5) GOTO 1
         AC=A
      END DO
      WRITE(6,*) I,RELER
      RETURN
1     B=C1*EXP(-H*A)
      RETURN
      END
```

```
      SUBROUTINE GREEN(I,N,A,B,AM,AP,G1,G2)
C*************************************************************
C     This subroutine computes the Green functions G1 and G2 at the
C     new station I.
C*************************************************************
      IMPLICIT NONE
      REAL               G1(0:1024),G2(0:1024)
      REAL               G1OLD(0:1024),G2OLD(0:1024)
      REAL               A,B,AP,AM
      REAL               ATT,AOLD,BOLD,APOLD,AMOLD,ZOLD,EOLD
      REAL               H,C1,C2,C3
      INTEGER            N,I,J
      COMMON /OLD/       ATT,AOLD,BOLD,APOLD,AMOLD,ZOLD,EOLD
      COMMON /OLDG/      G1OLD,G2OLD
C
      H=1./N
      ATT=ATT*EXP(H*(BOLD+B)/2)
      AP=(A+B)/2/ATT
      AM=(A-B)/2*ATT
C
      DO J=0,N-I
         C1=1-.25*H*H*AP*AM
         C2=G1OLD(J)+.5*H*APOLD*G2OLD(J)+.5*H*AP*G2OLD(J+1)
         C3=.25*H*H*AP*AMOLD*G1OLD(J+1)
         G1(J)=(C2+C3)/C1
         G2(J)=G2OLD(J+1)+.5*H*(AM*G1(J)+AMOLD*G1OLD(J+1))
      END DO
      RETURN
      END
```

```
      SUBROUTINE PROFILE(I,H,A,B,Z,E)
C     ********************************************************
C     This subroutine takes the computed values of A and B at
C     the position I and converts them to permittivity and
C     conductivity values.
C     The field E(z)=c(0)/c(z).
C     ********************************************************
      IMPLICIT NONE
      REAL               G1OLD(0:1024),G2OLD(0:1024)
      REAL               E,A,B,Z,H,HSL,EPSRZ,SIG,EPS,CON
      REAL               ATT,AOLD,BOLD,APOLD,AMOLD,ZOLD,EOLD
      INTEGER            I
      COMMON /OLD/       ATT,AOLD,BOLD,APOLD,AMOLD,ZOLD,EOLD
      COMMON /CONSTANTS/ EPSRZ,HSL,CON
      COMMON /OLDG/      G1OLD,G2OLD
C
1000  FORMAT(6E13.5)
C
      E=EOLD*EXP(-H*(AOLD+A)*.5)
      Z=ZOLD+HSL*.5*(EOLD+E)
      EPS=EPSRZ/E/E
      WRITE(22,1000) Z,EPS,SIG(Z),H*I,A,B
      RETURN
      END
```

```
      SUBROUTINE UPDATE(I,N,A,B,AM,AP,Z,E,G1,G2)
C*************************************************************
C     This subroutine updates the arrays and the variables.
C     The attenuation factor ATT is already updated in the subroutine
C     GREEN.
C*************************************************************
      IMPLICIT NONE
      REAL               G1OLD(0:1024),G2OLD(0:1024)
      REAL               G1(0:1024),G2(0:1024)
      REAL               A,B,AM,AP,E,Z
      REAL               ATT,AOLD,BOLD,APOLD,AMOLD,ZOLD,EOLD
      REAL               EPSRZ,HSL,CON
      INTEGER            N,I,J
      COMMON /OLD/       ATT,AOLD,BOLD,APOLD,AMOLD,ZOLD,EOLD
      COMMON /CONSTANTS/ EPSRZ,HSL,CON
      COMMON /OLDG/      G1OLD,G2OLD
C
      DO J=0,N-I
        G1OLD(J)=G1(J)
        G2OLD(J)=G2(J)
      END DO
      AOLD=A
      BOLD=B
      APOLD=AP
      AMOLD=AM
      ZOLD=Z
      EOLD=E
      RETURN
      END
```