MASTER THESIS

SciLab Program for the Calculation of Lightning EM-Fields using different Return Stroke Model

by

Ing. Andreas F. Dvořak BSc.

Thesis Submitted to the Faculty of
Electrical Engineering and Information Technology
In Partial Fulfillment
of the Requirements for the Degree
Master of Science
In
Power Engineering
Technical University of Vienna

supervision by
ao.Univ.-Prof.i.R. Dipl.-Ing. Dr.techn. Wolfgang Hadrian
Univ.Lektor Dipl.-Ing. Dr.techn. Gerhard Diendorfer

Vienna, 03.06.2014

Contents

Acknowledgment

I want to thank my teachers at the TGM in Vienna for awaken my interest in electrical engineering. I want to thank all the people at the Technical University of Vienna for channel my interest to energy technology. I want to thank also my supervisors for center my interest on lightnings. Especially I want to thank them for their patience.
I am also much obliged to all my friends that companion me. Friends I knew for ages. Friends that shared a part of my journey. But everybody a friend I can always count on. You all make my day.

Dedicated to the Past: Heidi and Helmut: In grateful and loving memory to my late parents. Died too young but live forever on in my heart.

Dedicated to the Present: Eva. Love of my life. My days would be dark without you. My heart would be crying without you. My life would be empty without you. I love you more than I can put into words.

Dedicated to the Future: Victoria, David and Jacob. You are the sunshine on rainy days. You are the light in dark times. You are the pride of my life. I love you.

Abstract

The computation of the electromagnetic (EM) fields of a return stroke is important to compare, evaluate and enhance the engineering models. Due the increasing costs of software packages rise compare to the funds for the academic research reduce it is economically interesting to take a closer look at open source software.

Within the scope of engineering and scientific numerical computation SCILAB ([Scilab-Enterprises, 2014]) is such a package. SCILAB is released as cross-platform open source application under the CeCILL license ([CEA-CNRS-INRIA, 2013]).

For that reason four engineering models of the current distribution along the lightning return stroke channel and the resulting electrical and magnetically fields are programmed in SCILAB and compared with published results of other computations.

Additionally the measured currents of lightning discharges are recorded as stroke to an elevated object for obvious reasons. Therefore also the strike to a tall object is implemented in SCILAB, the fields are calculated and compared with published data.

Chapter 1

Introduction

When the connection between the leader and ground is established the return stroke current starts to propagate the channel upwards. This is the most visible effect due to a lightning stroke. It is also the process that produces most of the damage, on the one hand through the current itself and on the other hand through the remote electromagnetic fields.

There are a few aspects to be discussed: the current at the striking point, the channel, the current along the channel being based on the engineering models and the remote fields.

1.1 Current at the Striking Point

The current at the channel-base is either measured or approximated by analytic functions. Two of the most widely used functions are the double-exponential function (equation 1.1) and the so called Heidler function from [Heidler, 1985] (equation 1.2).

$$I(0,t) = I_0 \cdot (e^{-\frac{t}{T_1}} - e^{-\frac{t}{T_2}}) \tag{1.1}$$

$$I(0,t) = \frac{I_0}{\eta} \cdot \frac{(\frac{t}{T_1})^n}{(\frac{t}{T_1})^n + 1} \cdot e^{-\frac{t}{T_2}} \quad \text{with} \quad \eta = e^{\frac{T_1}{T_2} \cdot (n\frac{T_2}{T_1})^{\frac{1}{n}}} \tag{1.2}$$

Sometimes also the sum of functions is used to approximate typically measured lightning current waveforms. So [Nucci et al., 1990] used a sum of a double-exponential and a Heidler function (see Figure 1.1) and [Diendorfer and Uman, 1990] used two Heidler functions to calculated the shapes of the undisturbed current (see Figure 1.2 and Figure 1.3). The values used to calculate the current $I(0,t)$ are shown in Table 1.1.

[Thottappillil et al., 1997] showed that instead of the current also the line charge density is a potential source for the field computation.

Table 1.1: Values used to calculate the undisturbed current waveforms used by [Nucci et al., 1990] and [Diendorfer and Uman, 1990].

	I_0 / kA	T_1 / µs	T_2 / µs	n
Current N1	−7.5	100	6	–
[Nucci et al., 1990]	−9.9	0.072	5	2
Current D1	−13	0.15	3	2
[Diendorfer and Uman, 1990]	−7	5	50	2
Current D2	−28	0.3	6	2
[Diendorfer and Uman, 1990]	−16	10	50	2

Note that the current $I(0,t)$ at the striking point is independent of the return stroke model used to calculate the current $I(z',t)$ along the channel and the remote fields. So a measured current combined with the measured fields are a strong basis for the validation of a model.

1.2 Simplifications of the Lightning Channel

For the computation of the fields caused by return strokes in engineering models there is normally a straight vertical channel assumed. In fact the real channel is known to be tortuous and branched. The effects of tortuosity and channel branches on the radiated fields are studied theoretically (see [Rakov and Uman, 2007]).

There is also a perfectly conducting ground assumed. So the boundary conditions of a perfect conductor require that the horizontal electrical field component and the vertical magnetic filed component are equal to zero at the ground. This simplifies the computation of the fields (section 1.4).

In most cases the return-front speed is also assumed as constant within the range of $1 \cdot 10^8 \frac{\text{m}}{\text{s}}$ to $2 \cdot 10^8 \frac{\text{m}}{\text{s}}$ [Rakov, 2007].

1.3 Current along the Channel based on Return Stroke Engineering Models

The following generalized equation for the current along the channel is given by [Rakov, 1997]:

$$I(z',t) = u(t - \frac{z'}{v_f}) \cdot P(z') \cdot I(0, t - \frac{z'}{v}) \tag{1.3}$$

$u(t - \frac{z'}{v_f})$: Heavyside function, which expresses that the current in the

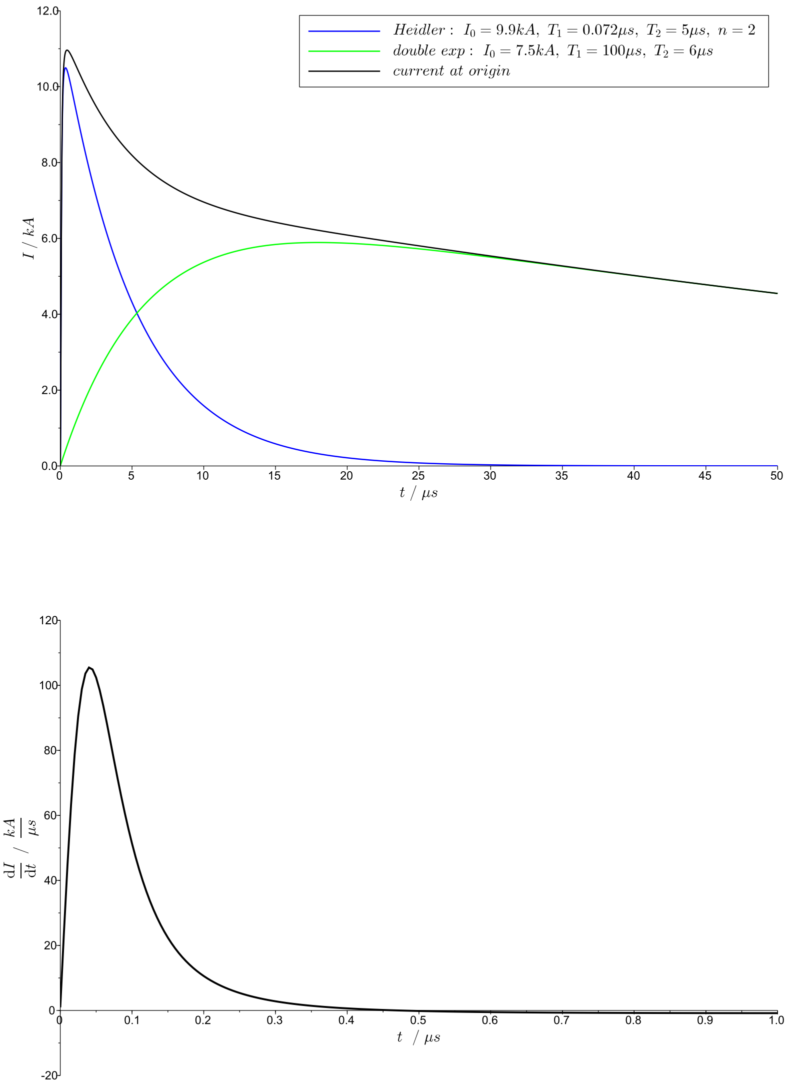


Figure 1.1: Undisturbed current N1 and its derivate used by [Nucci et al., 1990].

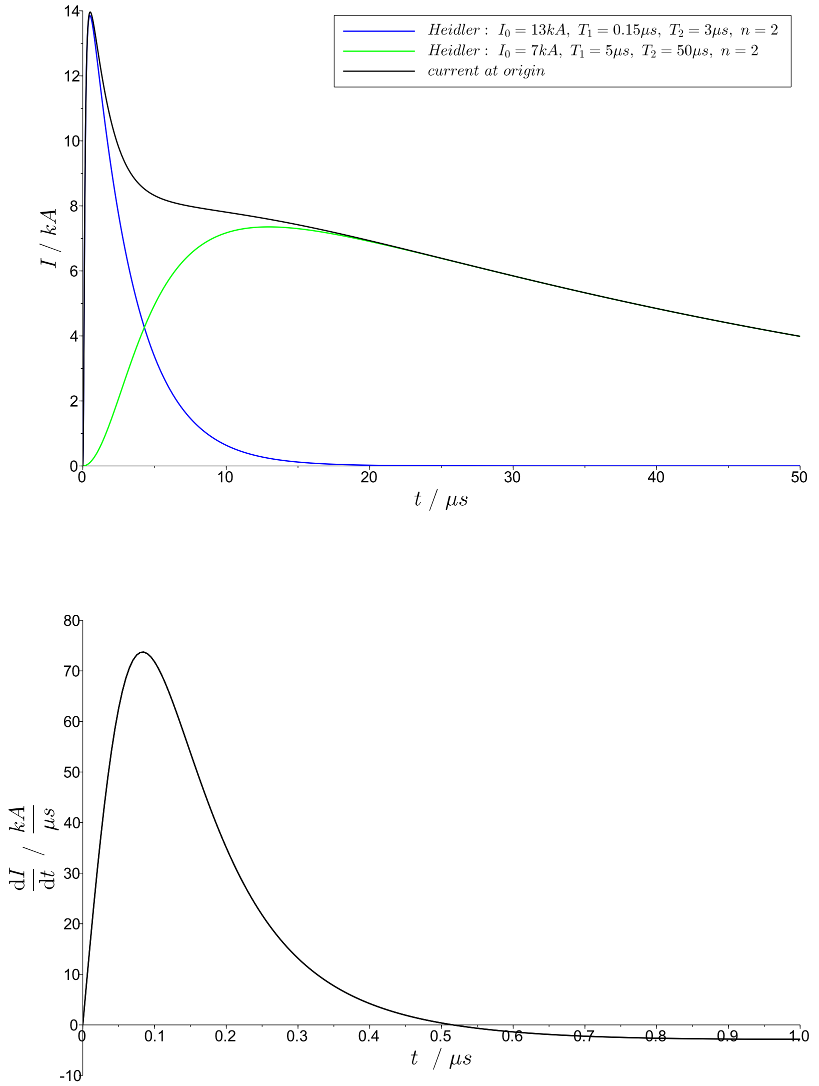


Figure 1.2: Undisturbed current D1 and its derivate used by [Diendorfer and Uman, 1990].

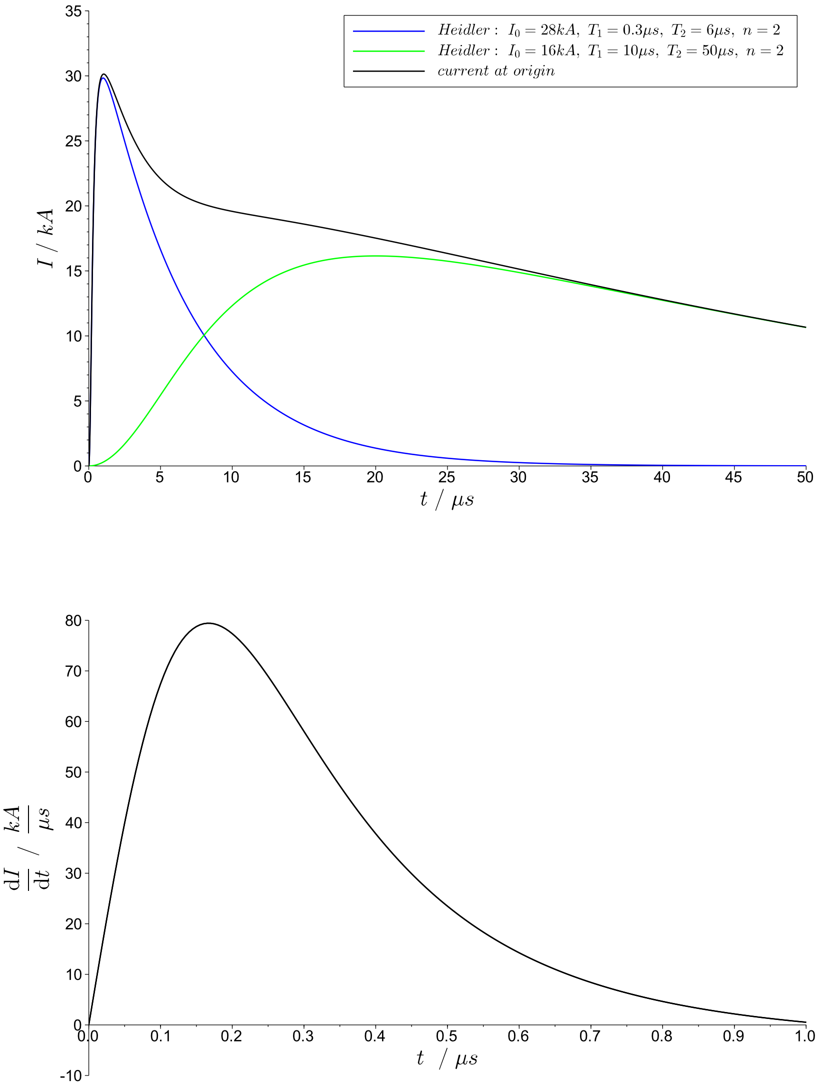


Figure 1.3: Undisturbed current D2 and its derivate used by [Diendorfer and Uman, 1990].

channel is equal zero at the height z' until the current wave reaches this height

$P(z')$: height-dependent current attenuation factor ([Rakov and Dulzon, 1991]), which describes the modification of the current as a function of the height z'

v_f: speed of the upward propagating retrun-stroke-front

v: current-wave propagation speed

$I(0, t - \frac{z'}{v})$: current at the channel-base which propagates with the speed v

So the basic information to compute the current distribution along the channel according to an engineering model is known. $P(z')$ and v are model specific parameters which are listed in Table 1.2.

Table 1.2: Model specific parameters for generalized current equation.

model	$P(z')$	v
TL	1	v_f
MTLE	$e^{-\frac{z'}{\lambda}}$	v_f
TCS	1	$-c$
DU	1	$-c$

Generally a model is a mathematical construct of the real world with simplifications. The simplifications are necessary due to the complexity of the processes, the huge number of parameters and in most cases also the lack of knowledge about both the process and the parameters. So they have effects on all aspects of the engineering models.

The models are designed to fit as good as possible to experimentally observed characteristics of the described process. The typical characteristics for the return stroke models are the measured fields, both at near and at far distances. The typical benchmark for the models is the comparison of the measured with the calculated fields.

There are two different types of return stroke models. The one which are based on the transmission line model, TL ,[Uman and McLain, 1969] (see following section 1.3.1 and 1.3.2), and the other one based on the travelling current source model, TCS ([Heidler, 1985], described more detailed in section 1.3.3 and 1.3.4 in this document).

1.3.1 Transmission Line (TL) Model

The widely used transmission line model assumes a perfect conducting channel with a current source at the ground which injects the current at the channel-base. Then the current wave propagates upward with constant speed v_f. Therefore the current-wave speed is positive and equal to the return-front speed v.

The idea behind the transmission line model is very simple and easy to calculate. When the connection between the cloud and the ground is established, the currentflow starts at the channel-base and the current wave propagates upward with the speed v_f to the cloud. Lightning current starts with a fast rising front to a peak value which is caused by the high charge density at the front section of the descending leader connected to ground. The peak is followed by a slower decrease because the amount of available charges decreases. So it can be seen as a current source at the channel-base which injects the current $I(0,t)$ in the channel (see Figure 1.4).

Though the channel is seen as a perfect conductor there is neither a decrease of the current level according to the heigth nor an attenuation according to the propagation. So the current at any height z' at any time t is given by

$$I(z',t) = I(0, t - \frac{z'}{v_f}) \tag{1.4}$$

The current distribution along the channel is illustrated in Figure 1.7a.

1.3.2 Modified Transmission Line Model (MTLE) with Exponential Decay

The MTLE model modifies the TL model so that there is still no distortion but an attenuation with exponential decrease according to the height z' as given in equation 1.5. There is also the so called MTLL model, which assumes a linear decrease, but this model is not discussed here. The parameter for the exponential decay is λ which is normally assumed to be about 2000m.

A decrease is visually noticeable and physical explanation is a discharge along the lightning channel.

$$I(z',t) = e^{-\frac{z'}{\lambda}} \cdot I(0, t - \frac{z'}{v_f}) \tag{1.5}$$

The current distribution along the channel for the MTLE model is illustrated in Figure 1.7b.

1.3.3 Traveling Current Source (TCS) Model

[Heidler, 1985] postulated a new engineering model based on the charge deposited along the leader. When the connection to the ground is established,

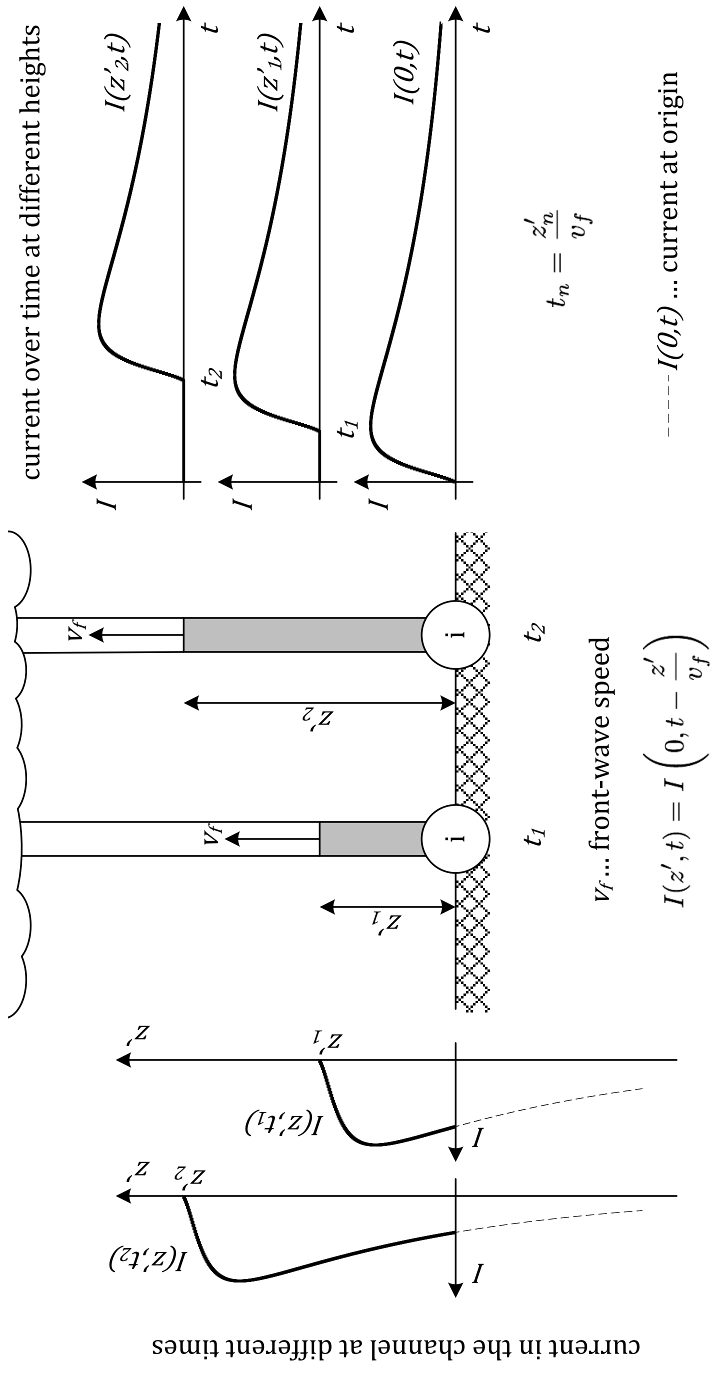


Figure 1.4: TL model: current along the channel at different times and heights.

these charges are the source for the current. So it resembles a current-source moving upward with the speed v_f (see Figure 1.5). The injected current-wave propagates downward with the speed of light c.

The current along the RS channel $I(z', t)$ is

$$I(z', t) = u(t - \frac{z'}{v_f}) \cdot I(0, t + \frac{z'}{c}) \tag{1.6}$$

With the arrival of the return stroke front at a particular height the charges movement starts immediately. This causes a current discontinuity at the return stroke front so that the field calculation needs to be extended (see section 1.4).

The current distribution along the channel for the TCS model is illustrated in Figure 1.7c.

1.3.4 Diendorfer-Uman (DU) Model

Along a leader channel there are two charged areas, the highly ionized leader core with its head and a corona sheath. [Diendorfer and Uman, 1990] used two current components to improve the TCS model. The first one is the short time but high-peak breakdown-current from the highly ionized areas (leader and channel core) and the second one is the slow rising and decaying corona-current representing the curved resulting from the collection of charges in the corona sheath (see Figure 1.2 and 1.3). Exponential discharge with the time constant τ_B and τ_C is assumed for the breakdown and corona sheath respectively. Both are computed with equation 1.7. The current in the channel is the sum of both components.

As a result of the assumed exponential discharge of these areas the discontinuity at the front is substituted with a slope (second term in Eqn. 1.7) with different time constants for corona sheath and breakdown current component, because the charges in the corona need a longer time to move to the channel and effect as a current compared to the time the charges in the leader core need to accelerate.

$$I(z', t) = I(0, t + \frac{z'}{c}) + e^{-(t - \frac{z'}{v_f})\tau_D^{-1}} \cdot I(0, \frac{z'}{v_f} + \frac{z'}{c}) \tag{1.7}$$

Where the discharge time constant $\tau_D = \tau_B$ and $\tau_D = \tau_C$ for the breakdown and corona sheath area, respectively.

The current distribution along the channel for the DU model is illustrated in Figure 1.7d.

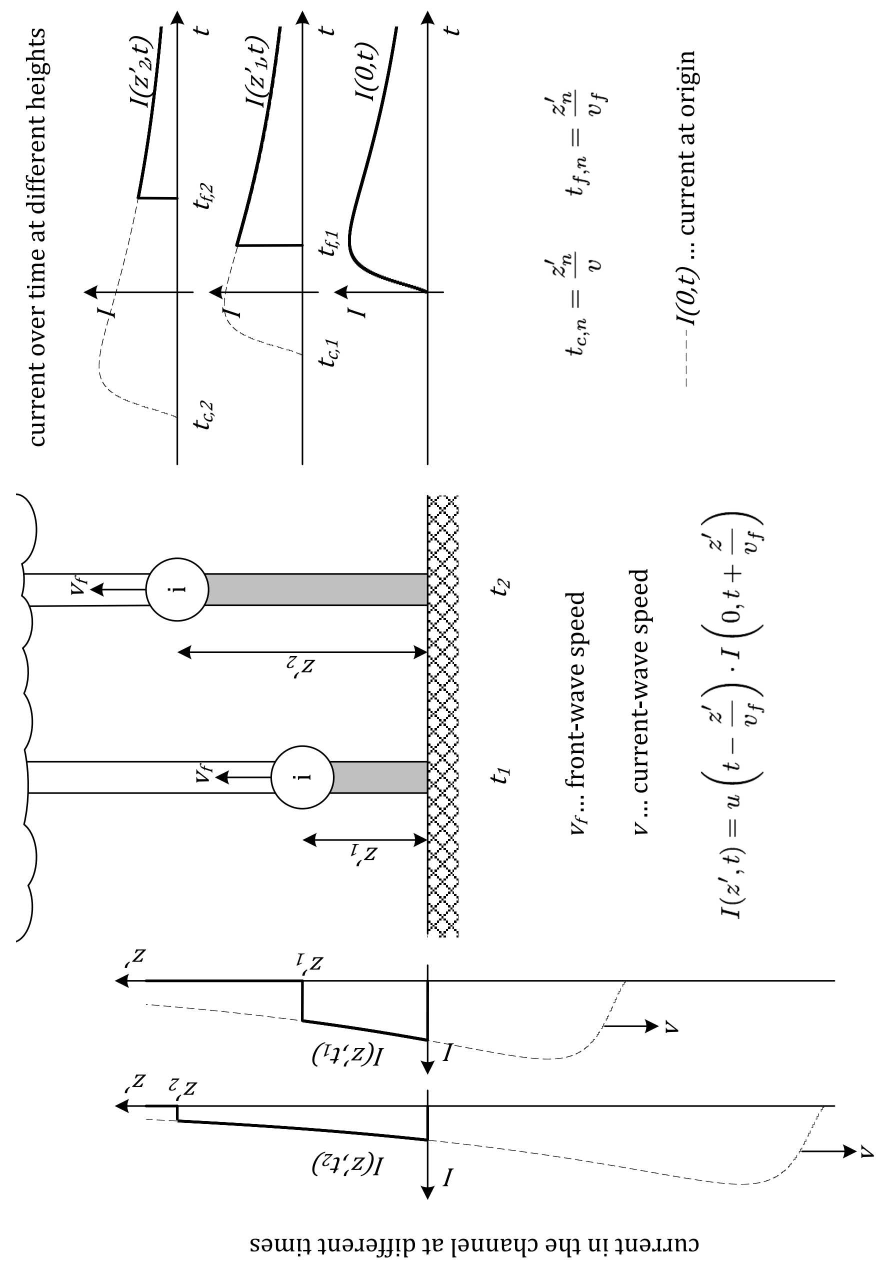


Figure 1.5: TCS model: current along the channel at different times and heights.

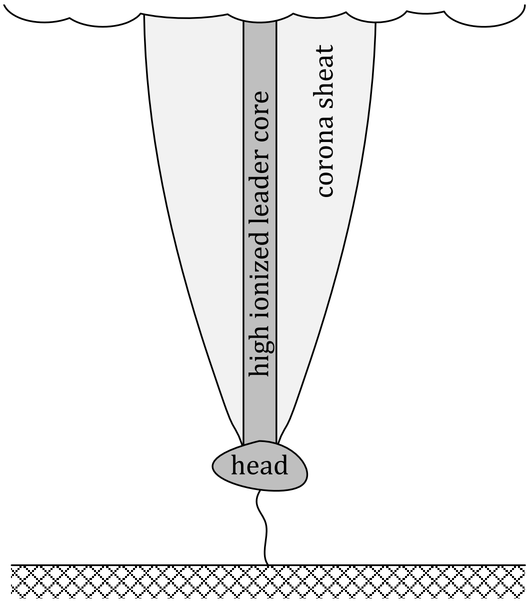
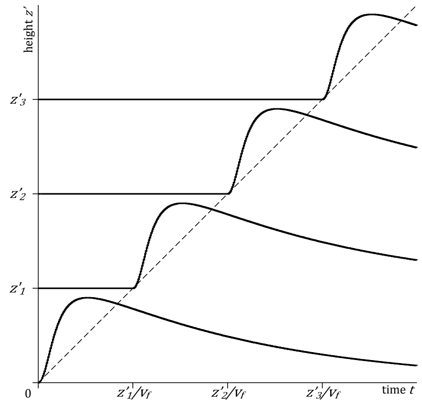


Figure 1.6: Charged areas as base for the DU-model.

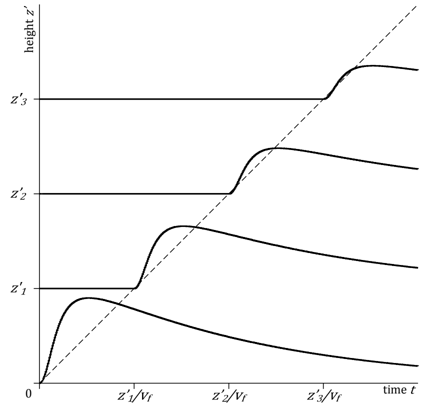
1.4 Electric and Magnetic Field Calculation

Calculating the fields of a return stroke is fundamental in comparing the engineering models to each other and to the measured fields as well as calculating the induced voltages appearing on electric power or telecommunication lines. The assumption made is the perfectly conducting ground and the methods chosen are the approximation of the channel through fragmentation in small electric dipoles (Figure 1.8) and the method of image charge.

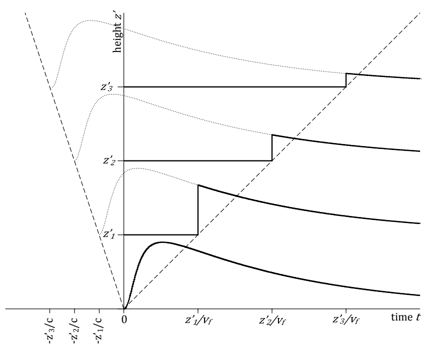
With those simplifications and the field point at ground level $z = 0$ and the distance r from the striking point the equations for the vertical electric field $E_z(r,t)$ and the magnetic field $B_\phi(r,t)$ are given by [Thottappillil et al., 1997]:



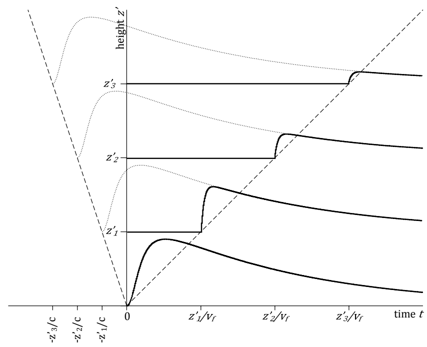
(a) TL model.



(b) MTLE model.



(c) TCS model.



(d) DU model.

Figure 1.7: Current in the channel at different heights for four engineering models.

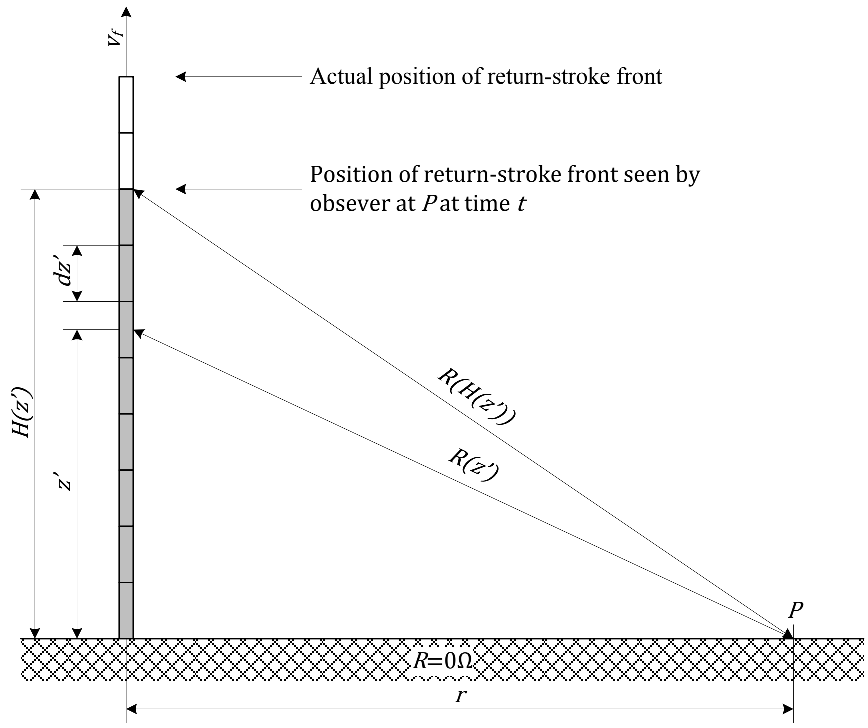


Figure 1.8: Geometry for computation of the remote fields. Adapted from [Thottappillil et al., 1997].

$$E_z(r,t) = \frac{1}{2\pi\varepsilon_0} \int_0^{H(t)} \left[\frac{2z'^2 - r^2}{R^5(z')} \int_{\frac{z'}{v_f}+\frac{R(z')}{c}}^{t} I\left(z', \tau - \frac{R(z')}{c}\right) \mathrm{d}\tau \right.$$
$$+ \frac{2z'^2 - r^2}{c\,R^4(z')}\; I\left(z', t - \frac{R(z')}{c}\right)$$
$$\left. - \frac{r^2}{c^2\,R^3(z')} \frac{\partial I\left(z', t - \frac{R(z')}{c}\right)}{\partial t} \right] \mathrm{d}z'$$
$$- \frac{1}{2\pi\varepsilon_0} \frac{r^2}{c^2\,R^3(H(t))}\; I\left(H(t), \frac{H(t)}{v_f}\right) \frac{\mathrm{d}H(t)}{\mathrm{d}t} \tag{1.8}$$

$$B_\phi(r,t) = \frac{\mu_0}{2\pi} \int_0^{H(t)} \left[\frac{r}{R^3(z')} \, I\left(z', t - \frac{R(z')}{c}\right) \right.$$

$$\left. + \frac{r}{c\,R^2(z')} \frac{\partial I\left(z', t - \frac{R(z')}{c}\right)}{\partial t} \right] \mathrm{d}z'$$

$$+ \frac{\mu_0}{2\pi} \frac{r}{c\,R^2(H(t))} \, I\left(H(t), \frac{H(t)}{v_f}\right) \frac{\mathrm{d}H(t)}{\mathrm{d}t} \qquad (1.9)$$

The terms in 1.8 and 1.9 which are proportional to the current $I(t)$ are referred to as induction components, the terms proportional to the current derivates $\frac{\partial I}{\partial t}$ as radiation components. The term in 1.8 proportinal to $Q = \int I(t)\,\mathrm{d}t$ is referred to as the electrostatic component.

The last term in both equations is only used if there is a current discontinuity at the return stroke current front. The TCS model (1.3.3) is the only considered model which implicates such a discontinuity.

1.5 Lightning Strike to a Tall Object

To study a lightning strike to a tall grounded object is very important. On one hand it is the only possibility (expect of triggered strokes) for direct measurements of the current of a return stroke and on the other hand it is relevant for lightning protection.

[Baba and Rakov, 2005] derived the expression for the current $I(z',t)$ as a sum of the injected wave and reflected respectively transmitted waves. Along the tall object ($0 \leq z' \leq h$) with a configuration shown in Figure 1.9 the current distribution is given by

$$I(z',t) = \frac{1-\varrho_{top}}{2} \sum_{n=0}^{\infty} \left[\varrho_{bot}^n \, \varrho_{top}^n \, I_{SC}\left(h, t - \frac{h-z'}{c} - \frac{2nh}{c}\right) \right.$$

$$\left. + \varrho_{bot}^n \, \varrho_{top}^n \, I_{SC}\left(h, t - \frac{h+z'}{c} - \frac{2nh}{c}\right) \right] \text{ for } 0 < z \leq h \quad (1.10)$$

In the channel ($z' \geq h$) there are only the injected wave and the transmitted waves across the object-channel-junction.

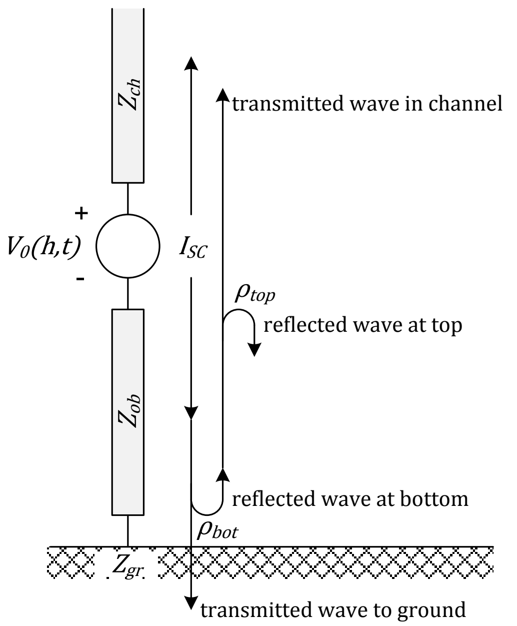


Figure 1.9: Lightning strike to a tall object. Object and lightning channel are represented by lossless transmission lines connected in series with a lumped voltage source. Adapted from [Baba and Rakov, 2005].

$$I(z',t) = \frac{1-\varrho_{top}}{2}\left[I_{SC}\left(h, t - \frac{z'-h}{v}\right) + \sum_{n=1}^{\infty} \varrho_{bot}^{n}\, \varrho_{top}^{n-1}(1+\varrho_{top})\, I_{SC}\left(h, t - \frac{z'-h}{v} - \frac{2nh}{c}\right)\right] \text{ for } z \geq h \quad (1.11)$$

cohere

$I_{SC}(h,t) = \frac{V_0(h,t)}{Z_{CH}}$ is the lightning current that would be measured at an ideal grounded ($Z_{OB} = Z_{GR} = 0$) object with negligible height.

$\varrho_{bop} = \frac{Z_{OB}-Z_{GR}}{Z_{OB}+Z_{GR}}$: reflection coefficient at the bottom of the tall object for downward propagating waves in the tall object.

$\varrho_{top} = \frac{Z_{OB}-Z_{CH}}{Z_{OB}+Z_{CH}}$: reflection coefficient at the top of the tall object for upward propagating waves in the tall object.

Z_{OB} : characteristic impedance of the tall object

Z_{GR} : characteristic impedance of the ground

Z_{CH} : characteristic impedance of the lightning channel

[Pavanello et al., 2007] used a different set of equations, based on the concept of the undisturbed current (1.12).

$$\begin{aligned} I(z',t) = \Bigg[& i_0\left(h, t - \frac{z-h}{v}\right) - \varrho_{top}\, i_0\left(h, t - \frac{z-h}{c}\right) \\ & + (1-\varrho_{top})(1+\varrho_{bot}) \sum_{n=0}^{\infty} \left[\varrho_{bot}^{n+1}\, \varrho_{top}^{n}\, i_0\left(h, t - \frac{z+h}{c} - \frac{2nh}{c}\right) \right] \Bigg] \\ & \cdot u\left(t - \frac{z-h}{v}\right) \quad \text{for } 0 < z \leq h \end{aligned} \tag{1.12}$$

$$\begin{aligned} I(z',t) = (1-\varrho_{top}) & \sum_{n=0}^{\infty} \left[\varrho_{top}^{n} \varrho_{bot}^{n}\, i_0\left(h, t - \frac{h-z}{c} - \frac{2nh}{c}\right) \right. \\ & \left. + \varrho_{top}^{n} \varrho_{bot}^{n+1}\, i_0\left(h, t - \frac{h+z}{c} - \frac{2nh}{c}\right) \right] \\ & \cdot u\left(t - \frac{h+z}{c} - \frac{2nh}{c}\right) \quad \text{for } 0 < z \leq h \end{aligned} \tag{1.13}$$

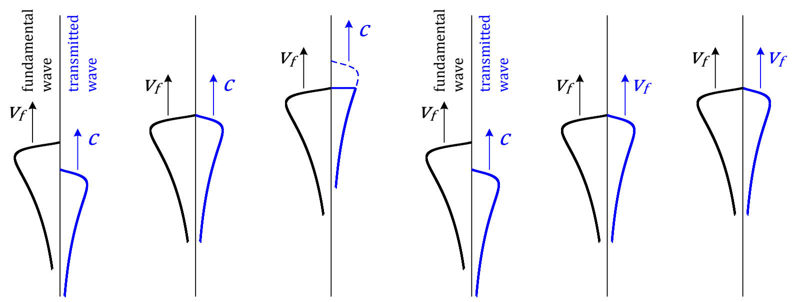
Because the return stroke front propagates with $v_f \approx \frac{1}{3} \ldots \frac{2}{3} \cdot c$, but the transmitted waves propagates in the highly conductive channel with the speed of light c there is the question what will happen when these waves reach the return stroke front.

[Baba and Rakov, 2005] used implicitly and [Pavanello et al., 2007] used explicitly the Heavyside function to cut off the transmitted respectively reflected waves. The problem here is the occurrence of a discontinuity which results in a discontinuity component of the field. (see Figure 1.10a).

Another possibility is that the transmitted waves change their propagation speed to v_f when they arrive at the front (see Figure 1.10b). This is explainable through the energy exchange between the charges although the assumed abrupt change of the speed for the complete wave is implausible. It has to be more of a compression of the wave shape through the continuous speed change when it approaches the front.

The results of the computation of the fields at a distance of 100km and 5km with the set of assumptions [Pavanello et al., 2007] used (confer 4.2) and the neglect of the discontinuity is plotted in Figures 1.11 to 1.16 .

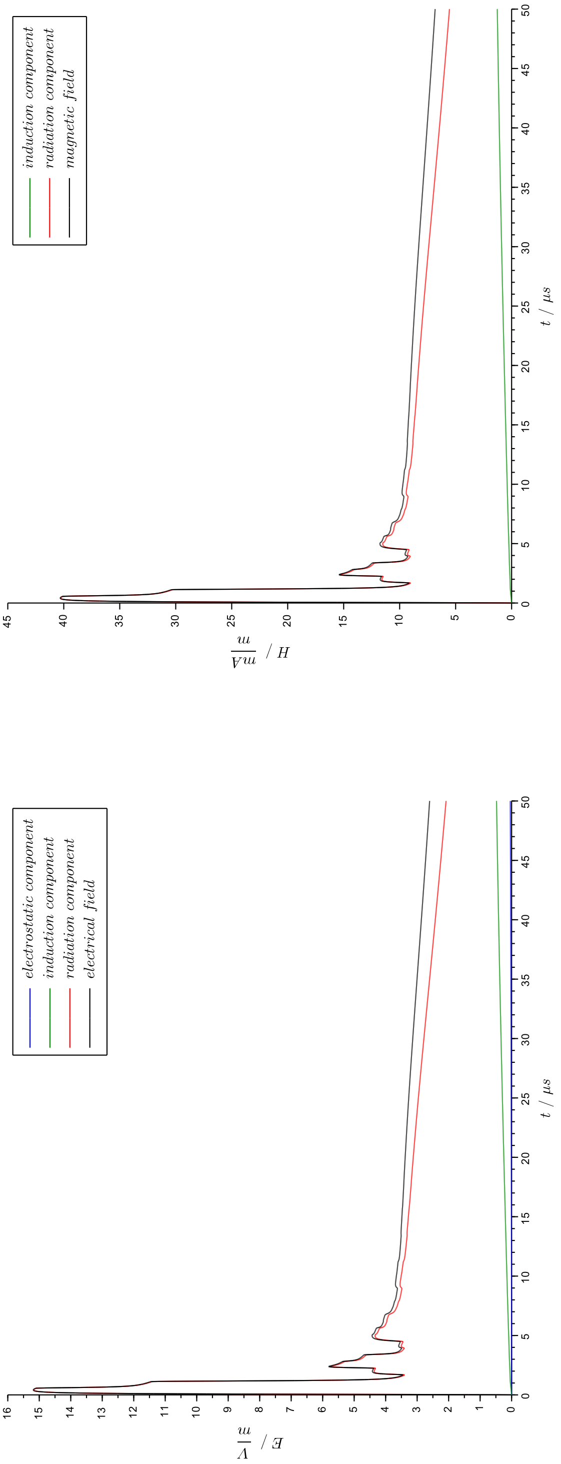
The comparison in figures 1.17 and 1.18 shows that there is no major difference between the change of the propagation speed and the neglection of the discontinuity, but the considering of the discontinuity changes a lot.



(a) Transmitted wave with cut-off thru Heavyside function.

(b) Transmitted wave with abrupt speed change at the return stroke front.

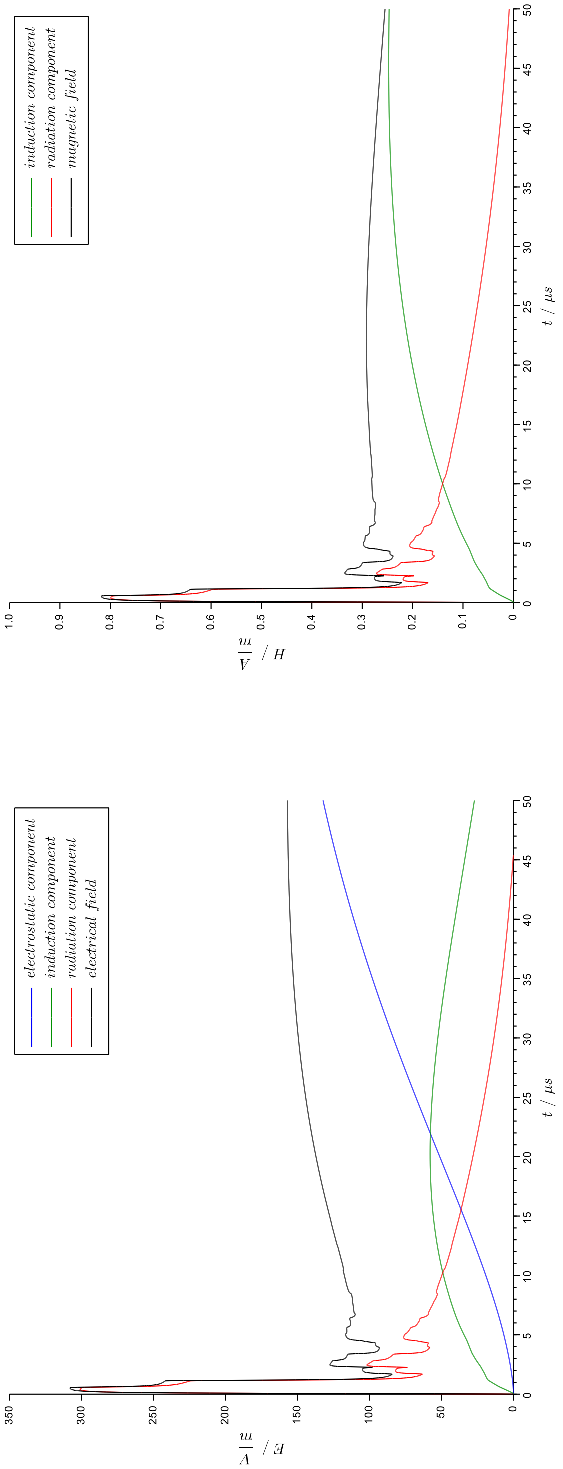
Figure 1.10: Comparison of the process used by [Baba and Rakov, 2005] and in this thesis when a transmitted wave arrives at the return stroke front.



(a) Electrical field

(b) Magnetic field

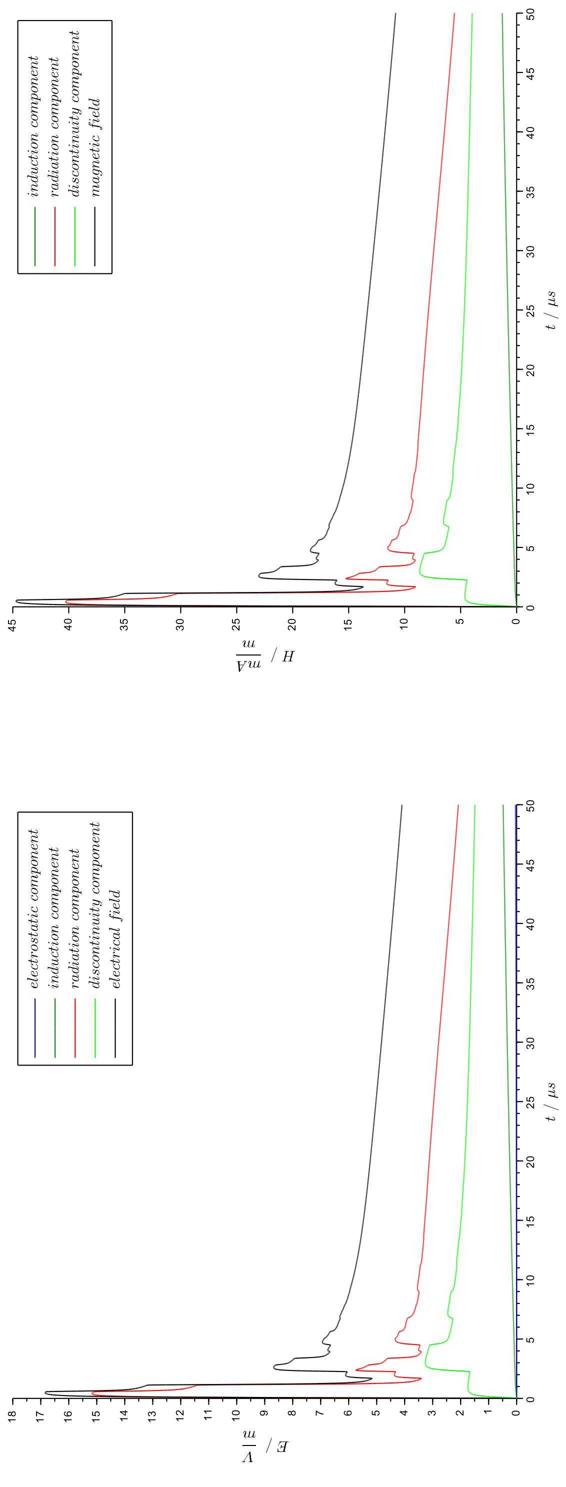
Figure 1.11: Fields in 100km distance in case of neglecting the discontinuity.



(a) Electrical field

(b) Magnetic field

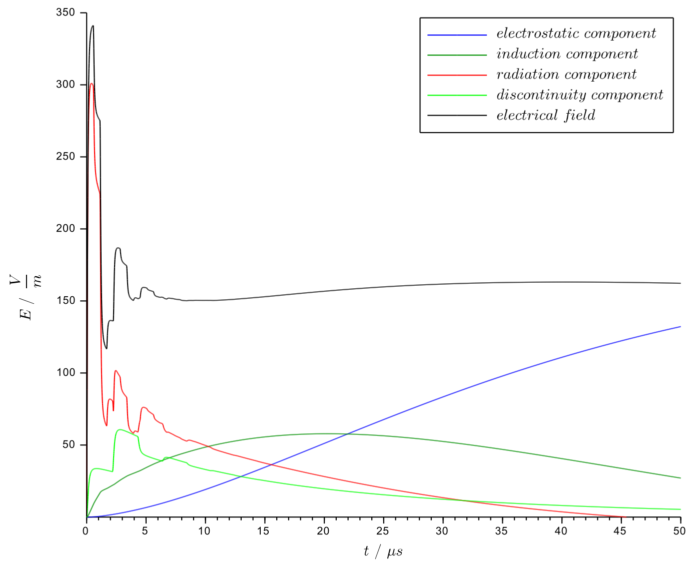
Figure 1.12: Fields in 5km distance in case of neglecting the discontinuity.



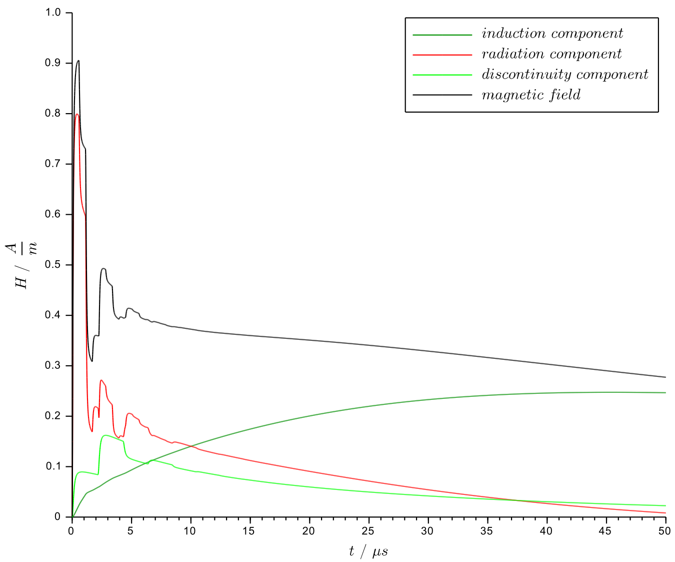
(a) Electrical field

(b) Magnetic field

Figure 1.13: Fields in 100km distance in case of considering the discontinuity.

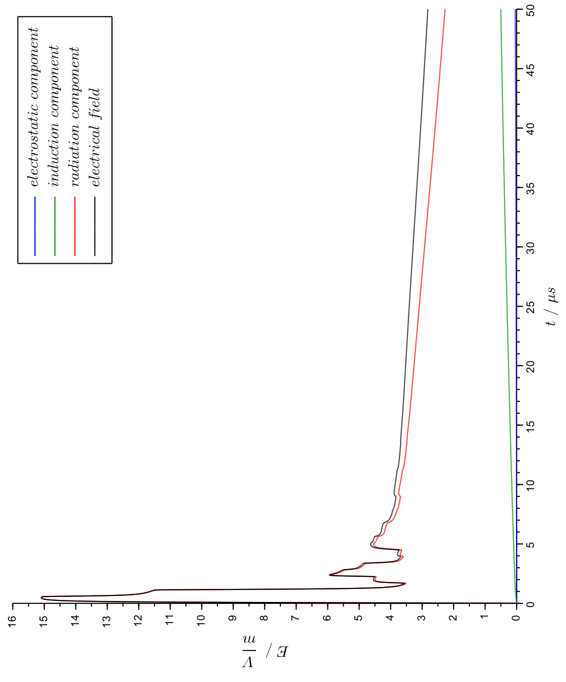


(a) Electrical field

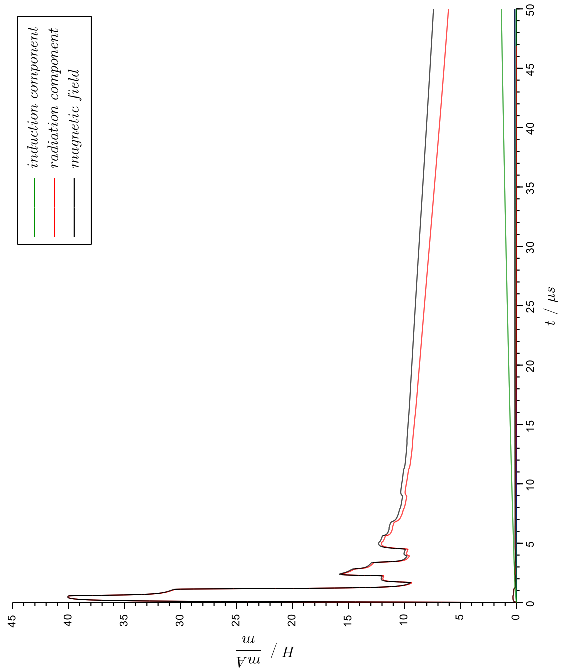


(b) Magnetic field

Figure 1.14: Fields in 5km distance in case of considering the discontinuity.

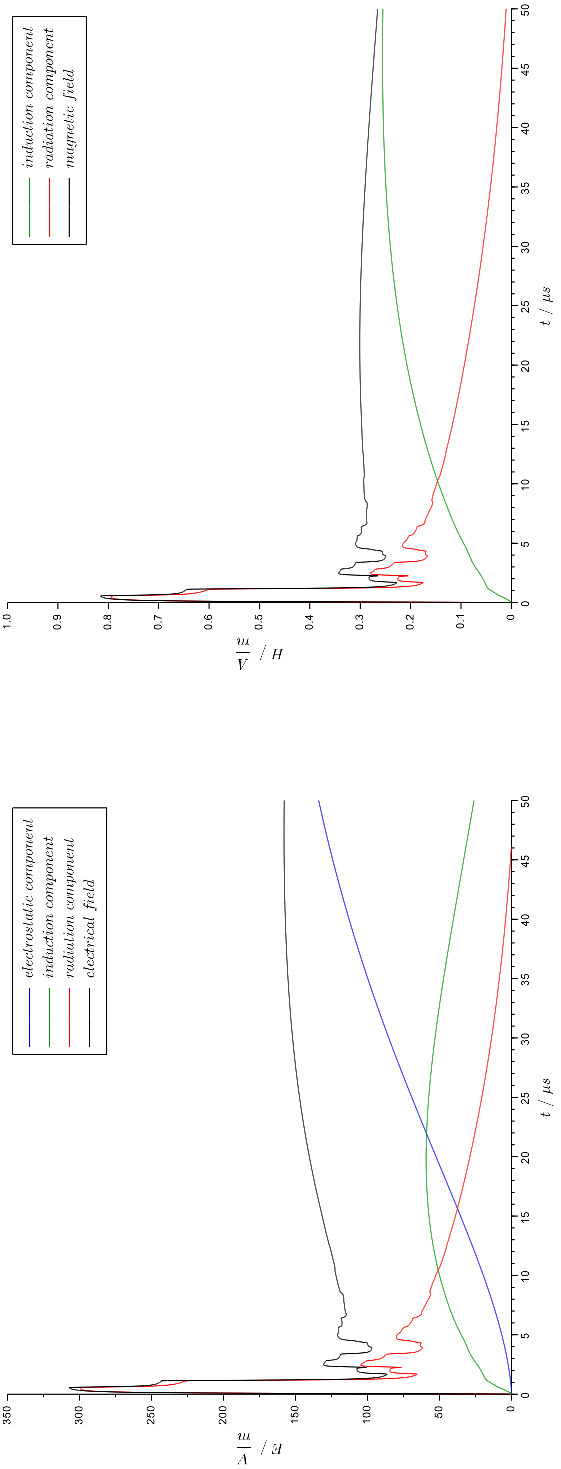


(a) Electrical field



(b) Magnetic field

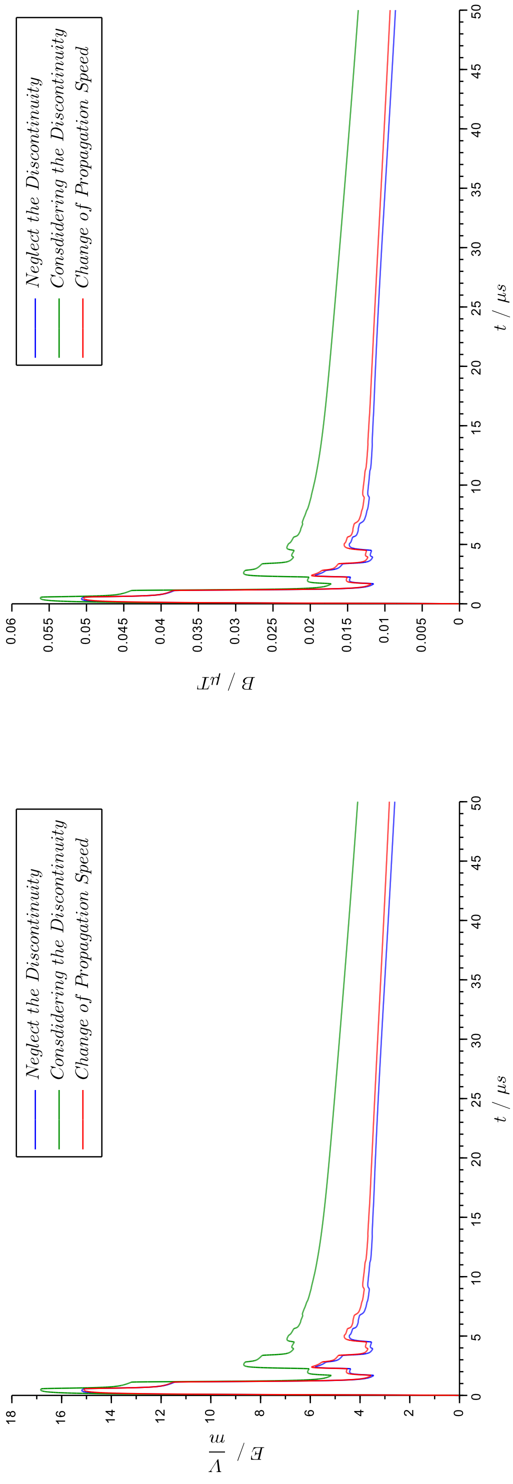
Figure 1.15: Fields in 100km distance in case of changing the propagation speed.



(a) Electrical field

(b) Magnetic field

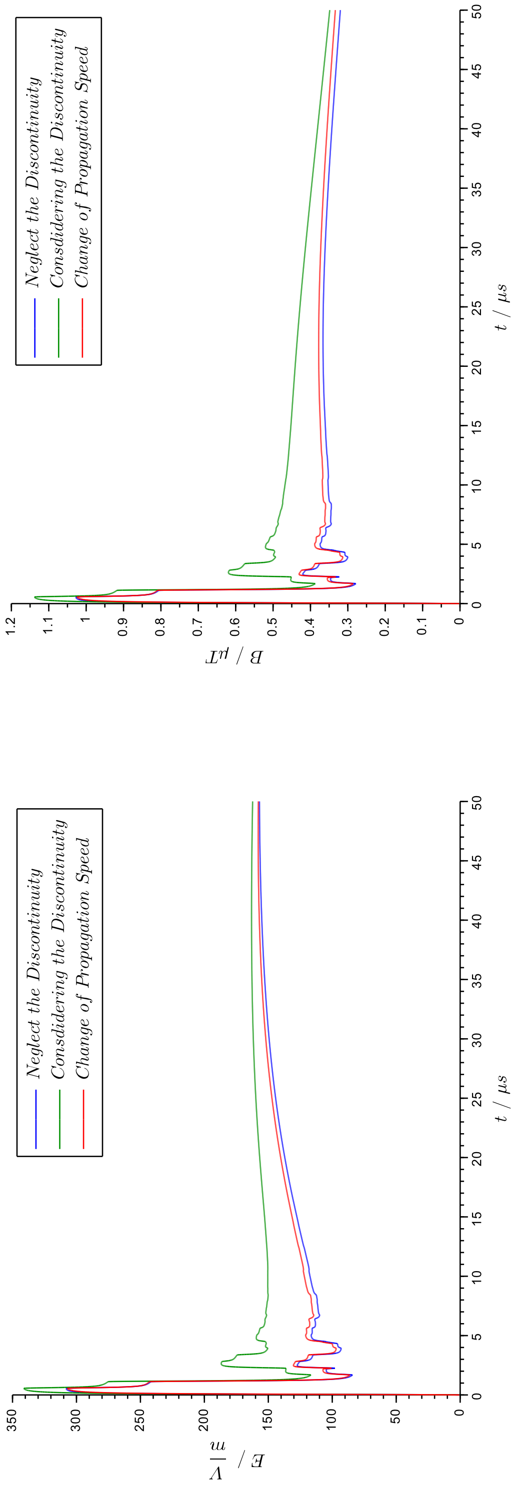
Figure 1.16: Fields in 5km distance in case of changing the propagation speed.



(a) Electrical field

(b) Magnetic field

Figure 1.17: Comparison of the total fields in 100km distance in all three case.



(a) Electrical field

(b) Magnetic field

Figure 1.18: Comparison of the total fields in 5km distance in all three case.

Chapter 2

Lightning Strike to the Ground

The program calculates the remote fields in case of a lightning strike to a perfectly conducting ground. The used return stroke models are TL, MTLE, TCS and DU. The field calculation is possible for one return stroke model with all components as well as for all four return stroke models for comparison. The user can define the input parameter via keyboard or file. The results are shown as plots and are storable as plain text file.

2.1 Basics

To calculate the transient fields a few parameter must be defined. First of all the time-vector is to be constructed. Then the channel have to be discretized. Further on there are a few other parameters to be defined.

2.1.1 Current and the Time-Vector

The time-vector is responsible for the calculation-time and for the accuracy of the computation. Unfortunately, like all other numerical solutions, the two are opposed to each other.

The calculation-time depends on the number of time-steps. The best results are, based on a MacBook Pro with 4GB, in the range 1000 to 5000 steps.

The major challenge is the accuracy of the numerical calculation of the fields. The shortest relevant time-window is found in the current derivates and therefore the radiation field component needs detailed attention. Recall the currents in section 1.1. [Diendorfer and Uman, 1990] used a 'slow' current with a relevant level of current derivative in a time-window from 0 to approx. 0.2μs (refer to Figure 1.3), on the other hand [Nucci et al., 1990] used a 'fast' current with a time-window from 0 to approx. 0.1μs (see Figure

1.1). The accuracy depends on the number of time-steps in that window. The best results were achieved with ≥ 10 steps in this time-window.

$t_0 = 0$s is always assumed as start-time. Up for definition is the end-time of calculation t_e as well as the number of time-steps n.

After calculating the channel-vector with Equation 2.5 it is necessary to extend the time-vector to calculate the current $I(0,t)$ because the TCS and DU models need a longer current due the movement of the current source with $\frac{z'}{c}$ (Figure 1.5).

The current at the point of origin $I(0,t)$ is the sum of up to 5 current components, which are either corresponding to the Heidler-function (1.2) or to the double exponential function (1.1).

The Scilab function to build the currents and the vectors is `calc_par` in `'func_par.sci'`.

2.1.2 Discretization of the Channel and the Channel-Vector

The easiest case to calculate the fields, which is the goal, would be if the field of the return-front at a height

$$z' = n \cdot \Delta z$$

effects at the field point P at a time

$$t = t_B + n \cdot \Delta t$$

in which $t = t_B$ is the runtime of the field from the striking point to the point where the field is calculated. Thus there would be a minimum of interpolation which means the computation would be as accurate as possible. As result of the difference in the speed of the return-front wave in the cannel v_f and the propagation speed of its fields this is not possible with equidistant channel-steps, a height-vector is to be build.

A consideration at the runtime of the current in the cannel t_I and of the field in the air t_F combined with the time-steps of the calculation solves this problem (Figure 2.1). The runtime equation 2.1 has to be valid for all time-steps $n \cdot \Delta t$ and all heights z'_n.

$$\mathbf{t_C} + \mathbf{t_A} - t_B = \mathbf{t} \tag{2.1}$$

With the notation introduced in Figure 1.8 together with the time-vector

$$\mathbf{t} = (0 \quad \Delta t \quad 2\Delta t \ \cdots \ n\Delta t) \tag{2.2}$$

and the height-vector of the channel

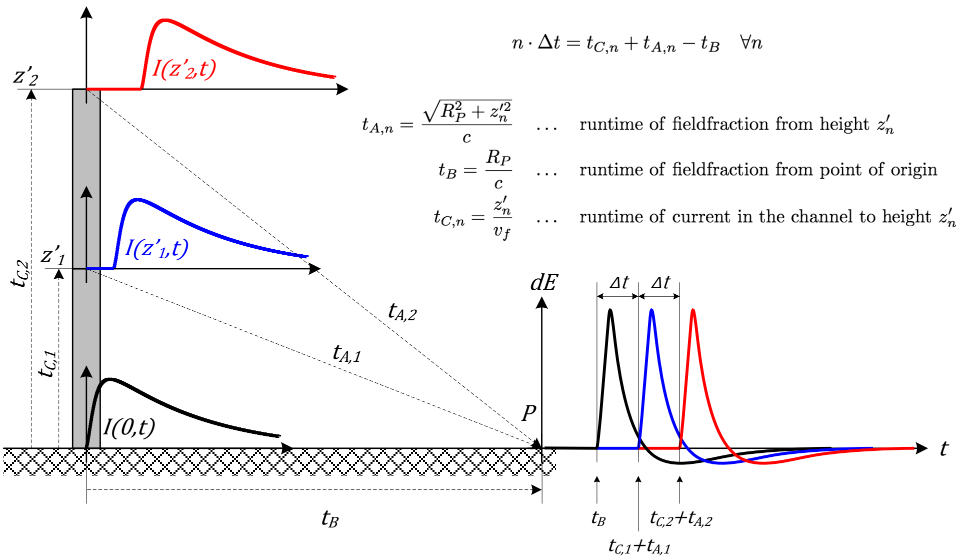


Figure 2.1: Method of discretization of the channel.

$$\mathbf{z}' = (0 \quad z_1' \quad z_2' \cdots z_n') \tag{2.3}$$

the runtime equation reveals

$$\mathbf{t} = \frac{\mathbf{z}'}{v_f} + \frac{\sqrt{R_P^2 + \mathbf{z}'^2}}{c} - \frac{R_P}{c} \tag{2.4}$$

which is a quadratic equation in $\mathbf{z}'$

$$\mathbf{z}'^2 \cdot \left[\left(\frac{c}{v_f}\right)^2 - 1\right] - \mathbf{z}' \cdot 2\frac{c}{v_f}(R_p + c\mathbf{t}) + \left[(R_p + c\mathbf{t})^2 - R_p^2\right] = 0 \tag{2.5}$$

and easy to solve.

Notice that there is no verification of the input data. If e.g. the front wave speed is too high or the period of calculation is too long the channel will be unrealistically long.

2.1.3 Other Parameter

Other parameter are the upward-propagating return stroke front speed v_f, the distance r of the point where the remote field is calculated (see Figure 1.8).

2.2 Current Distribution and Field Calculation

The current distribution as a function with the arguments time t and height z' would be a rectangular array with the dimension of the number of time-steps. This would be very memory intensive and so this array is not calculated explicitly. The current at a height z' is calculated separately to compute the field. Tests also showed the the calculation time is shorter with the use of this method.

2.2.1 Current at Height z'

The current is calculated with the argument time at fixed height according to the used model. Step by step the array of the required data is build to compute the field component.

TL Model

Through the discretization of the channel with consideration of the length of a time-step the number of time-steps is equal to the number of channel segments. So the computation of the current $I(z', t)$ is only a time shifting of $I(0, t)$. In the listing `j` is the height-index and `ca` is the number of channel segments and time-steps.

```
Izt=[zeros(1,(j-1)) I0t(1:(ca-j+1))]
```

MTLE Model

The MTLE model is the enhancement of the TL model with a height dependent factor $e^{-\frac{z'}{\lambda}}$ (see 1.3.2). In the listing `cv(j)` is the height at the height-index `j` and `la` is the parameter of the current decay λ.

```
Izt=[zeros(1,(j-1)) I0t(1:(ca-j+1))*exp(-cv(j)/la)]
```

TCS Model

As a result of the movement of the current source and of the return stroke front the current starts at time $t = \frac{z'_n}{v_f}$ at the heigh z' with the function $I(0, t - \frac{z'_n}{v_f} - \frac{z'_n}{c})$ (Figure 1.5). So firstly the current is calculated and then the time shift has do be done.

```
tsstart=cv(j)/vf+cv(j)/c
tsend=tsstart+(ca-j)*tv(2)
ts=tsstart:tv(2):tsend
Izt=interp(ts,tv,I0t,dI0t,"by_zero")
Izt=[zeros(1,(j-1)) Izt(1:(ca-j+1))]
```

DU Model

The DU model is an enhanced version of the TCS model with an exponential discharge time constant which affects and avoids the discontinuity at the return stroke front. Additionally the current $I(z', t)$ may be a sum of two currents (see 1.3.4).

```
tsstart=cv(j)/vf+cv(j)/c
tsend=tsstart+(ca-j)*tv(2)
ts=tsstart:tv(2):tsend
if I0tb==zeros(I0tb) then
   Izt=interp(ts,tv,I0t,dI0t,"by_zero")
   dfb=exp(-tv(1:length(ts))/taub)
   Izt=Izt-Izt(1).*dfb
else
   Iztb=interp(ts,tv,I0tb,dI0tb,"by_zero")
   dfb=exp(-tv(1:length(ts))/taub)
   Iztc=interp(ts,tv,I0tc,dI0tc,"by_zero")
   dfc=exp(-tv(1:length(ts))/tauc)
   Izt=Iztb-Iztb(1).*dfb+Iztc-Iztc(1).*dfc
end
Izt=[zeros(1,(j-1)) Izt(1:(ca-j+1))]
```

2.2.2 Field Calculation

The field calculation is split in the four components referred to as electrostatic, induction, radiation and discontinuity field component. This is useful in order to minimize the memory size needed and to visualize the influence of different parameter like distance, current gradient, current value and so on. It is also possible to skip the calculation of components with less influence to reduce the computation time with minimal change of the source code.

The main module for the calculation is `'field.sci'`. It loads the functions, computes the field components and superposes them.

Electrostatic Field Component

For the electrostatic field component at first the charge-equivalent is calculated with integration by trapezoidal interpolation over the time (loop-variable `i`). Step by step the integration is done for every height (loop-variable `j`). So an array of the amount of charge is build.

```
dFe(j,i)=dFe(j,(i-1))+tv(2)*(Izt(i-1)+Izt(i))/2
```

The last step is the multiplication with the factor $\frac{1}{2\pi\varepsilon_0} \cdot \frac{2z'^2 - r^2}{R^5(z')}$ from equation 1.8

```
efac=(2*cv^2-Rp^2)./(R^5*(2*%pi*eps0))
```

and the integration over the channel-height.

```
Ee(i)=inttrap(cv(1:i),(dFe(1:i,i)'.*efac(1:i)))
```

Induction Field Component

The data of the array simply consists of the values of the current.

```
dFi(j,:)=Izt
```

Then the multiplication with the factor $\frac{1}{2\pi\varepsilon_0} \cdot \frac{2z'^2 - r^2}{c\,R^4(z')}$ from equation 1.8 for the electric field component and the factor $\frac{\mu_0}{2\pi} \cdot \frac{r}{R^3(z')}$ from equation 1.9 for the magnetic field component

```
efac=(2*cv^2-Rp^2)./(c*R^4*(2*%pi*eps0))
bfac=(mu0*Rp)./(2*%pi*R^3)
```

and the integration over the channel-height is done.

```
Ei(i)=inttrap(cv(1:i),(dFi(1:i,i)'.*efac(1:i)))
Bi(i)=inttrap(cv(1:i),(dFi(1:i,i)'.*bfac(1:i)))
```

Radiation Field Component

For the computation of the radiation field component the array consists of the derivates of the current. Due the numerical calculation it is necessary to be sure that the initial value equals zero.

In case of the TCS model the setting of the start tangent to zero is replaced with the condition of a monotone derivate.

```
dIzt=splin(tv,Izt)
dIzt=splin(tv,Izt,"clamped",[0 dIzt(ta)])
dFr(j,:)=[zeros(1,j-1) dIzt(1:(ca-j+1))]
```

The last step is the multiplication with the factor $\frac{-1}{2\pi\varepsilon_0} \cdot \frac{r^2}{c^2\,R^3(z')}$ from equation 1.8 for the electric field component and the factor $\frac{\mu_0}{2\pi} \cdot \frac{r}{c\,R^2(z')}$ from equation 1.9 for the magnetic field component.

```
efac=(-1)*(Rp^2)./(c^2*R^3*(2*%pi*eps0))
bfac=(mu0*Rp)./(2*%pi*c*R^2)
```

and the integration over the channel-height.

```
Er(i)=inttrap(cv(1:i),(dFr(1:i,i)'.*efac(1:i)))
Br(i)=inttrap(cv(1:i),(dFr(1:i,i)'.*bfac(1:i)))
```

Discontinuity Field Component

A discontinuity occurs only in the TCS model and consists of three parts. The factor $\frac{-1}{2\pi\varepsilon_0} \frac{r^2}{c^2\,R^3(H(t))}$ from equation 1.8 for the electric field component and the factor $\frac{\mu_0}{2\pi} \frac{r}{c\,R^2(H(t))}$ from equation 1.9 for the magnetic field component,

```
efac=-(Rp^2)./(c^2*R^3*(2*%pi*eps0))
bfac=(mu0*Rp)./(2*%pi*c*R^2)
```

the value of the current at the discontinuity

```
tsstart=cv/vf+cv/c
Iht=interp(tsstart,tv,I0t,dI0t,"by_zero")
```

and the derivate of the height $\frac{\mathrm{d}H(t)}{\mathrm{d}t}$.

```
dH=splin(tv(1:ca),cv)
```

2.3 Using the Program

After starting Scilab the working directory has to be changed to the path where the modules are saved (File Browser). Further on the initial module has to be loaded and started (command `exec` or right-click and "Execute in Scilab") (Figure 2.2).

The menu is visually divided in 4 parts (Figure 2.2). They are (1) the definitions of the input parameters, (2) calculating the field of one model, (3) calculating the fields of all models and (4) additional point for user support.

2.3.1 Defining the Input Parameters

The input is possible by reading them from the file `def_ground.txt` (see Listing A.1.5) or via keyboard (Figure 2.3).

Following parameters have to be defined:

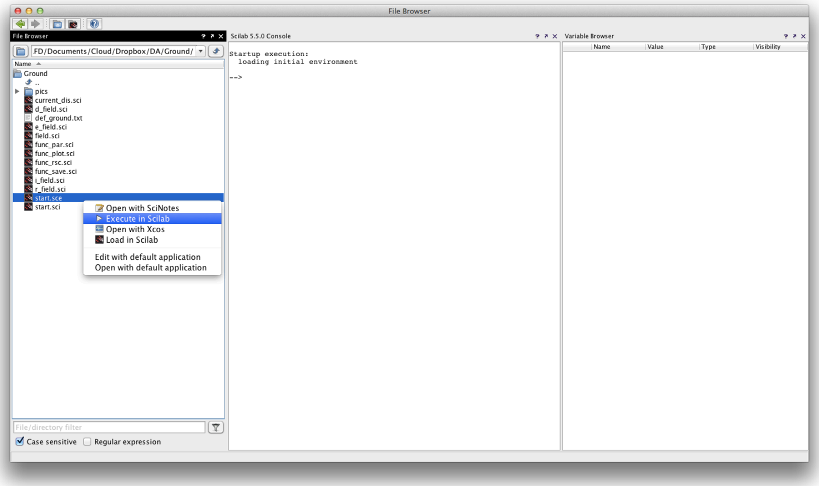
end-time: As a result that the start-time is always assumed at $t = 0$s, the end-time is equal to the period of time the field is calculated over. The unit is µs.

number of time-steps: Number of calculation-steps in the defined period of time. The ideal number is depended on the maximum of the current derivates, as discussed in section 2.1.1.

upward propagating return stroke front wave speed: Speed of the return stroke front in 10^8m/s.

distance of field point: Distance of the striking point to the point the field is calculated in km. It is r in Figure 1.8.

current definition: There are two pre-defined current-sets. One is used by [Nucci et al., 1990] (Figure 1.1), the other by [Diendorfer and Uman, 1990] (Figure 1.2). It is also possible to define a new set with up to 5 functions based on the double-exponential function (equation 1.1) and the so called Heidler function (equation 1.2). If required the current set can be plotted just as well as the derivative of the total current.



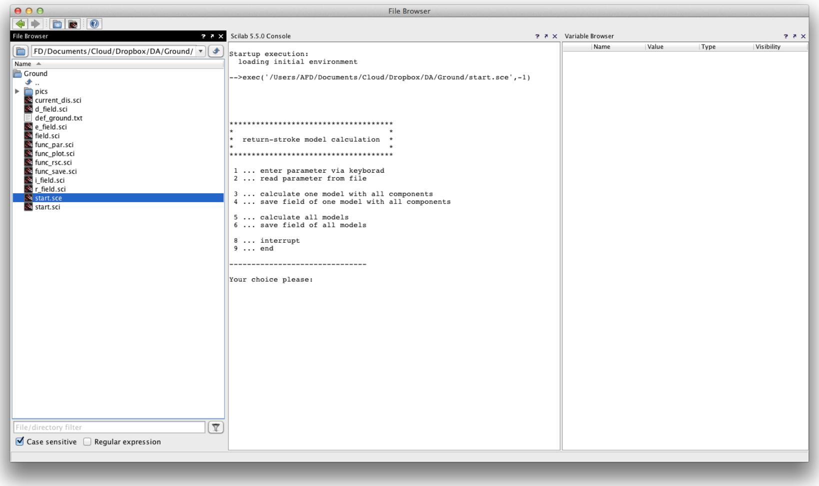


Figure 2.2: Screenshot: Start of the program to calculate the fields in case of a lightning strike to the ground.

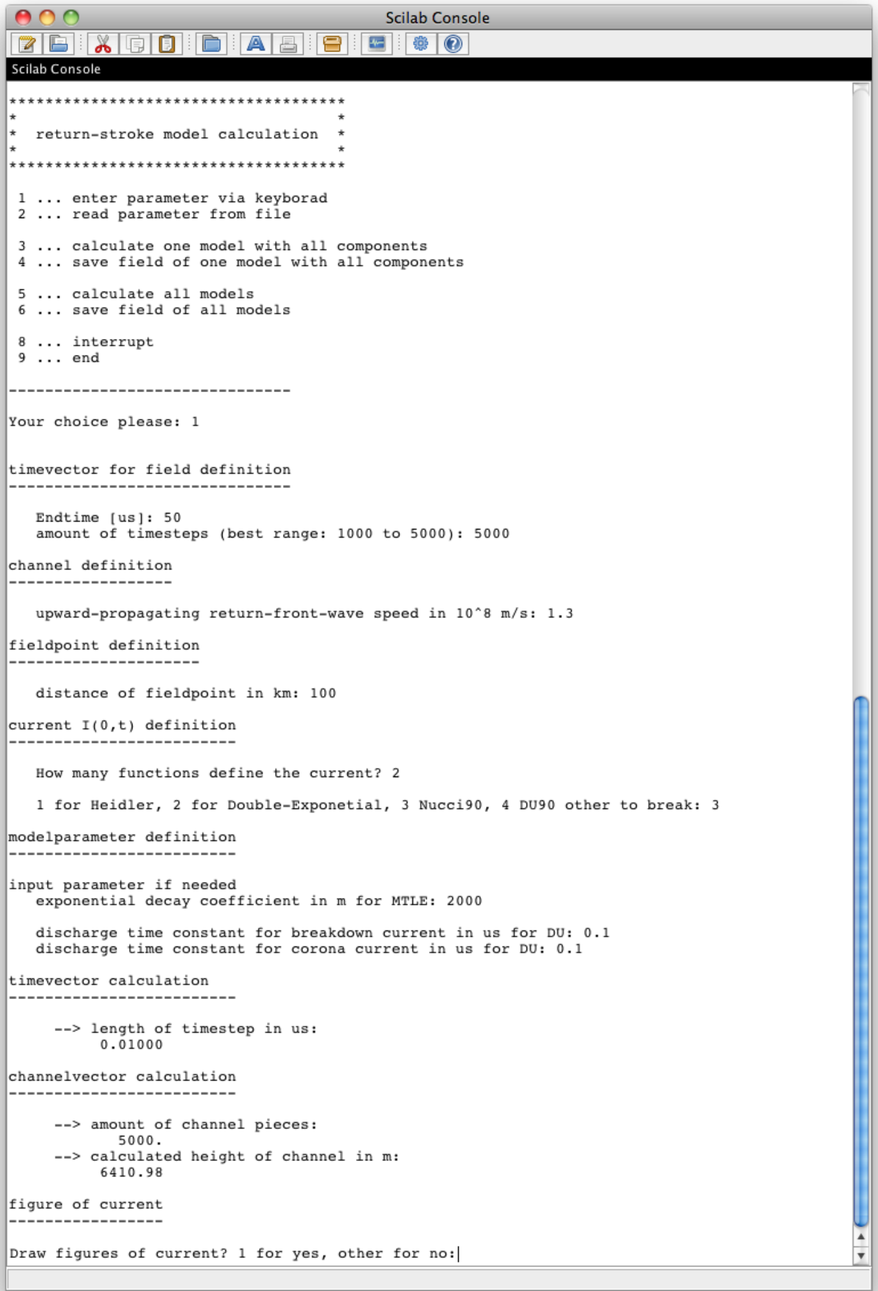


Figure 2.3: Screenshot: Defining the input parameters via keyboard.

model-parameter definition: The model specific parameters are the exponential decay λ in m for the MTLE model (section 1.3.2) and the time constants for the breakdown respectively the corona current in µs for the DU model (section 1.3.4). If not needed they are left empty.

The length of a time-step and the channel height are calculated based on these inputs and they are printed afterwards to validate them.

2.3.2 Calculating the Field for one Model

After defining the environment the calculation is possible. If calculating only one return stroke model with all field components, the model has to be chosen. The field components are calculated and if requested shown as a plot.

It is also possible to save the data as a plain text file (Figure 2.4). The header of the file contains the enironment and is followed by the data lines.

The header starts with the model type with the model-specific parameters (at least v_f) and the distance to the point the field is calculated. This is followed by the definition of the data lines. It starts with the order of the columns. Through the equidistance of the time-vector, it is only necessary to specify the start-time, which is the time of the arrival of the field at the field-point, the length of a time-step, which equals the unit of the x-axis, and the number of time-steps, which equals the length of the data-file. The last information in the header is the unity of the y-axis.

The data-lines follow the header. In Figure 2.5 a typical header followed by the first data-lines is shown.

2.3.3 Calculating the Field of all Models

To compare models it is necessary to plot and save them together. Therefore only the file-header changes (Figure 2.6) and the usage is the same as seen in section 2.3.2. If requested this part of the program plots a bunch of plots which include a comparison of all models as well as each single model with all its field components.

2.3.4 User Intervention

Sometimes it is important to provide direct access to the data for the user. So the menu-item 8 interrupts the program so that a data manipulation like saving, plotting or inspecting special data-areas is possible. Table 2.1 shows the important variables with a short description.

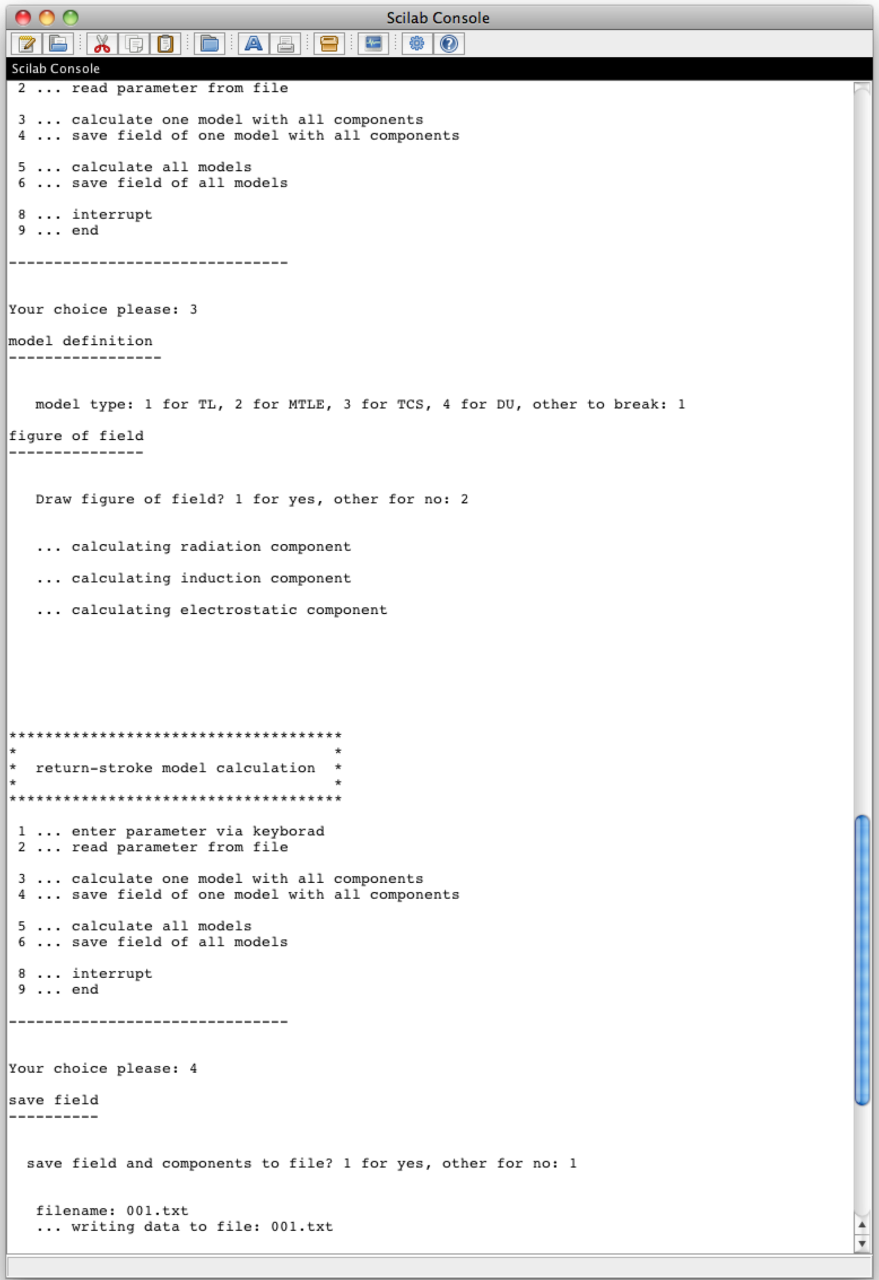
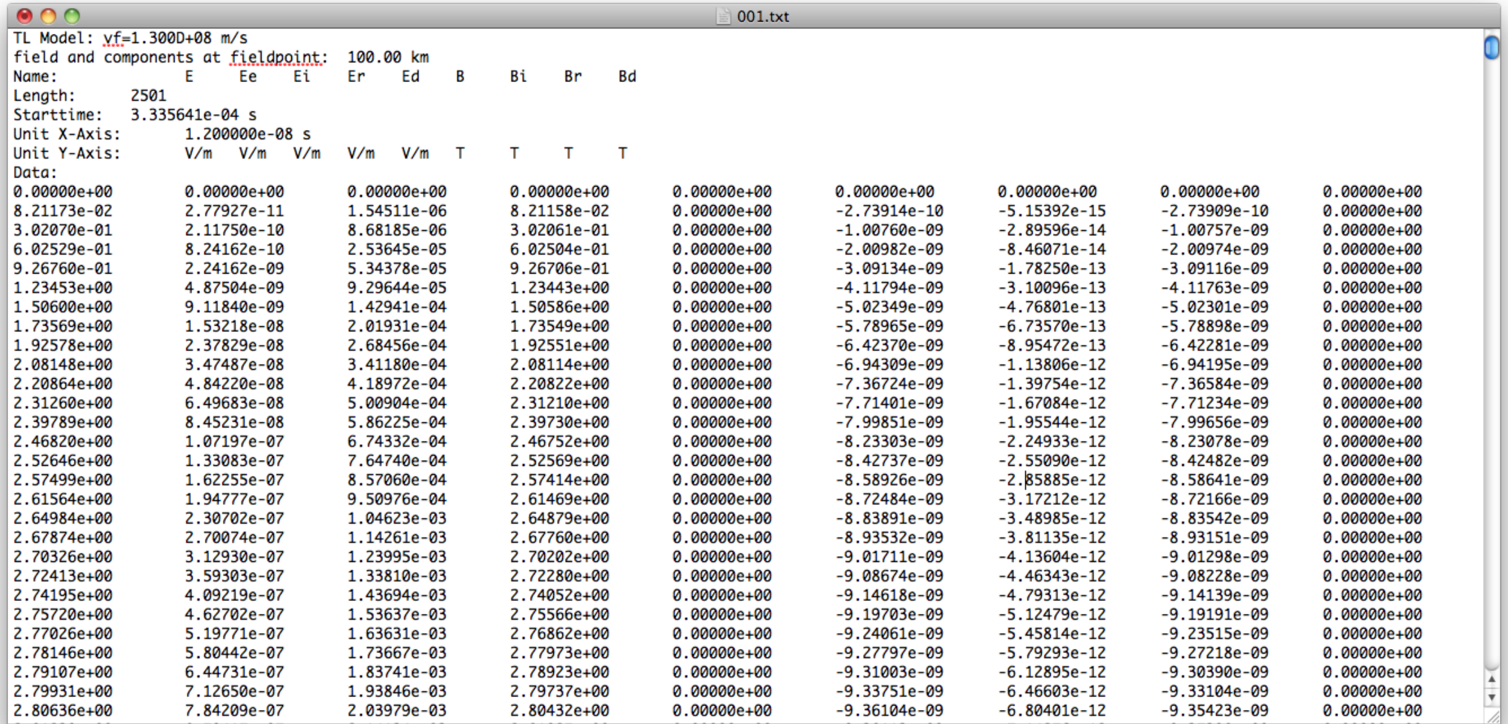
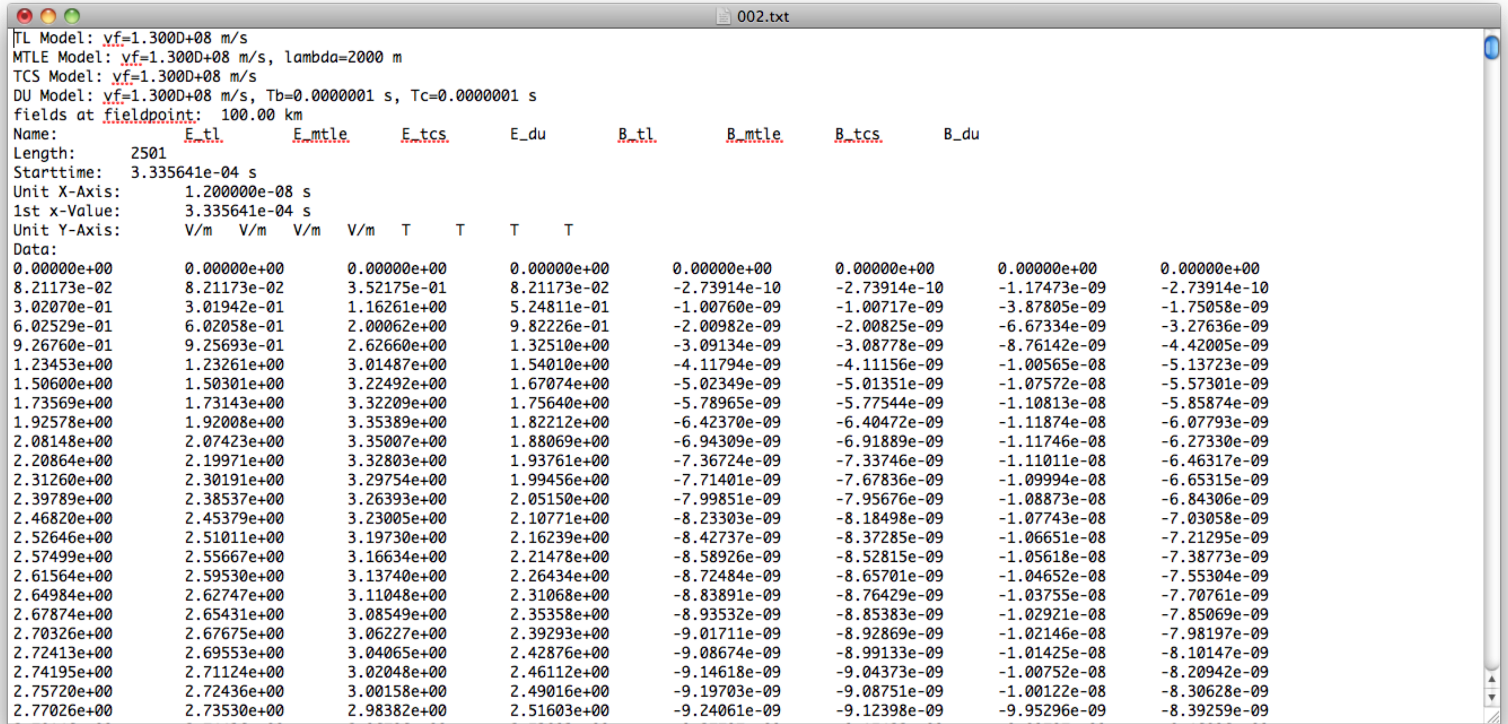


Figure 2.4: Screenshot: Calculation of one Model.



```
001.txt
TL Model: vf=1.300D+08 m/s
field and components at fieldpoint:  100.00 km
Name:           E     Ee    Ei    Er    Ed    B     Bi    Br    Bd
Length:      2501
Starttime:   3.335641e-04 s
Unit X-Axis:    1.200000e-08 s
Unit Y-Axis:    V/m   V/m   V/m   V/m   V/m   T     T     T     T
Data:
0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00
8.21173e-02   2.77927e-11   1.54511e-06   8.21158e-02   0.00000e+00   -2.73914e-10   -5.15392e-15   -2.73909e-10   0.00000e+00
3.02070e-01   2.11750e-10   8.68185e-06   3.02061e-01   0.00000e+00   -1.00760e-09   -2.89596e-14   -1.00757e-09   0.00000e+00
6.02529e-01   8.24162e-10   2.53645e-05   6.02504e-01   0.00000e+00   -2.00982e-09   -8.46071e-14   -2.00974e-09   0.00000e+00
9.26760e-01   2.24162e-09   5.34378e-05   9.26706e-01   0.00000e+00   -3.09134e-09   -1.78250e-13   -3.09116e-09   0.00000e+00
1.23453e+00   4.87504e-09   9.29644e-05   1.23443e+00   0.00000e+00   -4.11794e-09   -3.10096e-13   -4.11763e-09   0.00000e+00
1.50600e+00   9.11840e-09   1.42941e-04   1.50586e+00   0.00000e+00   -5.02349e-09   -4.76801e-13   -5.02301e-09   0.00000e+00
1.73569e+00   1.53218e-08   2.01931e-04   1.73549e+00   0.00000e+00   -5.78965e-09   -6.73570e-13   -5.78898e-09   0.00000e+00
1.92578e+00   2.37829e-08   2.68456e-04   1.92551e+00   0.00000e+00   -6.42370e-09   -8.95472e-13   -6.42281e-09   0.00000e+00
2.08148e+00   3.47487e-08   3.41180e-04   2.08114e+00   0.00000e+00   -6.94309e-09   -1.13806e-12   -6.94195e-09   0.00000e+00
2.20864e+00   4.84220e-08   4.18972e-04   2.20822e+00   0.00000e+00   -7.36724e-09   -1.39754e-12   -7.36584e-09   0.00000e+00
2.31260e+00   6.49683e-08   5.00904e-04   2.31210e+00   0.00000e+00   -7.71401e-09   -1.67084e-12   -7.71234e-09   0.00000e+00
2.39789e+00   8.45231e-08   5.86225e-04   2.39730e+00   0.00000e+00   -7.99851e-09   -1.95544e-12   -7.99656e-09   0.00000e+00
2.46820e+00   1.07197e-07   6.74332e-04   2.46752e+00   0.00000e+00   -8.23303e-09   -2.24933e-12   -8.23078e-09   0.00000e+00
2.52646e+00   1.33083e-07   7.64740e-04   2.52569e+00   0.00000e+00   -8.42737e-09   -2.55090e-12   -8.42482e-09   0.00000e+00
2.57499e+00   1.62255e-07   8.57060e-04   2.57414e+00   0.00000e+00   -8.58926e-09   -2.85885e-12   -8.58641e-09   0.00000e+00
2.61564e+00   1.94777e-07   9.50976e-04   2.61469e+00   0.00000e+00   -8.72484e-09   -3.17212e-12   -8.72166e-09   0.00000e+00
2.64984e+00   2.30702e-07   1.04623e-03   2.64879e+00   0.00000e+00   -8.83891e-09   -3.48985e-12   -8.83542e-09   0.00000e+00
2.67874e+00   2.70074e-07   1.14261e-03   2.67760e+00   0.00000e+00   -8.93532e-09   -3.81135e-12   -8.93151e-09   0.00000e+00
2.70326e+00   3.12930e-07   1.23995e-03   2.70202e+00   0.00000e+00   -9.01711e-09   -4.13604e-12   -9.01298e-09   0.00000e+00
2.72413e+00   3.59303e-07   1.33810e-03   2.72280e+00   0.00000e+00   -9.08674e-09   -4.46343e-12   -9.08228e-09   0.00000e+00
2.74195e+00   4.09219e-07   1.43694e-03   2.74052e+00   0.00000e+00   -9.14618e-09   -4.79313e-12   -9.14139e-09   0.00000e+00
2.75720e+00   4.62702e-07   1.53637e-03   2.75566e+00   0.00000e+00   -9.19703e-09   -5.12479e-12   -9.19191e-09   0.00000e+00
2.77026e+00   5.19771e-07   1.63631e-03   2.76862e+00   0.00000e+00   -9.24061e-09   -5.45814e-12   -9.23515e-09   0.00000e+00
2.78146e+00   5.80442e-07   1.73667e-03   2.77973e+00   0.00000e+00   -9.27797e-09   -5.79293e-12   -9.27218e-09   0.00000e+00
2.79107e+00   6.44731e-07   1.83741e-03   2.78923e+00   0.00000e+00   -9.31003e-09   -6.12895e-12   -9.30390e-09   0.00000e+00
2.79931e+00   7.12650e-07   1.93846e-03   2.79737e+00   0.00000e+00   -9.33751e-09   -6.46603e-12   -9.33104e-09   0.00000e+00
2.80636e+00   7.84209e-07   2.03979e-03   2.80432e+00   0.00000e+00   -9.36104e-09   -6.80401e-12   -9.35423e-09   0.00000e+00
```

Figure 2.5: Screenshot: Header and first data lines for a one model data-file.



```
002.txt
TL Model: vf=1.300D+08 m/s
MTLE Model: vf=1.300D+08 m/s, lambda=2000 m
TCS Model: vf=1.300D+08 m/s
DU Model: vf=1.300D+08 m/s, Tb=0.0000001 s, Tc=0.0000001 s
fields at fieldpoint:  100.00 km
Name:            E_tl       E_mtle     E_tcs      E_du       B_tl       B_mtle     B_tcs      B_du
Length:      2501
Starttime:   3.335641e-04 s
Unit X-Axis:     1.200000e-08 s
1st x-Value:     3.335641e-04 s
Unit Y-Axis:     V/m   V/m   V/m   V/m   T     T     T     T
Data:
0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
8.21173e-02      8.21173e-02      3.52175e-01      8.21173e-02      -2.73914e-10     -2.73914e-10     -1.17473e-09     -2.73914e-10
3.02070e-01      3.01942e-01      1.16261e+00      5.24811e-01      -1.00760e-09     -1.00717e-09     -3.87805e-09     -1.75058e-09
6.02529e-01      6.02058e-01      2.00062e+00      9.82226e-01      -2.00982e-09     -2.00825e-09     -6.67334e-09     -3.27636e-09
9.26760e-01      9.25693e-01      2.62660e+00      1.32510e+00      -3.09134e-09     -3.08778e-09     -8.76142e-09     -4.42005e-09
1.23453e+00      1.23261e+00      3.01487e+00      1.54010e+00      -4.11794e-09     -4.11156e-09     -1.00565e-08     -5.13723e-09
1.50600e+00      1.50301e+00      3.22492e+00      1.67074e+00      -5.02349e-09     -5.01351e-09     -1.07572e-08     -5.57301e-09
1.73569e+00      1.73143e+00      3.32209e+00      1.75640e+00      -5.78965e-09     -5.77544e-09     -1.10813e-08     -5.85874e-09
1.92578e+00      1.92008e+00      3.35389e+00      1.82212e+00      -6.42370e-09     -6.40472e-09     -1.11874e-08     -6.07793e-09
2.08148e+00      2.07423e+00      3.35007e+00      1.88069e+00      -6.94309e-09     -6.91889e-09     -1.11746e-08     -6.27330e-09
2.20864e+00      2.19971e+00      3.32803e+00      1.93761e+00      -7.36724e-09     -7.33746e-09     -1.11011e-08     -6.46317e-09
2.31260e+00      2.30191e+00      3.29754e+00      1.99456e+00      -7.71401e-09     -7.67836e-09     -1.09994e-08     -6.65315e-09
2.39789e+00      2.38537e+00      3.26393e+00      2.05150e+00      -7.99851e-09     -7.95676e-09     -1.08873e-08     -6.84306e-09
2.46820e+00      2.45379e+00      3.23005e+00      2.10771e+00      -8.23303e-09     -8.18498e-09     -1.07743e-08     -7.03058e-09
2.52646e+00      2.51011e+00      3.19730e+00      2.16239e+00      -8.42737e-09     -8.37285e-09     -1.06651e-08     -7.21295e-09
2.57499e+00      2.55667e+00      3.16634e+00      2.21478e+00      -8.58926e-09     -8.52815e-09     -1.05618e-08     -7.38773e-09
2.61564e+00      2.59530e+00      3.13740e+00      2.26434e+00      -8.72484e-09     -8.65701e-09     -1.04652e-08     -7.55304e-09
2.64984e+00      2.62747e+00      3.11048e+00      2.31068e+00      -8.83891e-09     -8.76429e-09     -1.03755e-08     -7.70761e-09
2.67874e+00      2.65431e+00      3.08549e+00      2.35358e+00      -8.93532e-09     -8.85383e-09     -1.02921e-08     -7.85069e-09
2.70326e+00      2.67675e+00      3.06227e+00      2.39293e+00      -9.01711e-09     -8.92869e-09     -1.02146e-08     -7.98197e-09
2.72413e+00      2.69553e+00      3.04065e+00      2.42876e+00      -9.08674e-09     -8.99133e-09     -1.01425e-08     -8.10147e-09
2.74195e+00      2.71124e+00      3.02048e+00      2.46112e+00      -9.14618e-09     -9.04373e-09     -1.00752e-08     -8.20942e-09
2.75720e+00      2.72436e+00      3.00158e+00      2.49016e+00      -9.19703e-09     -9.08751e-09     -1.00122e-08     -8.30628e-09
2.77026e+00      2.73530e+00      2.98382e+00      2.51603e+00      -9.24061e-09     -9.12398e-09     -9.95296e-09     -8.39259e-09
```

Figure 2.6: Screenshot: Header and first data lines for an all model data-file.

Table 2.1: Strike to the ground: List of variables

Name	Description
tv	time-vector of current (vector 1:m)
ta	number of time-steps of current (scalar)
cv	channel-vector (vector 1:n)
ca	number of channel segments and time-steps of field (scalar)
tend	endtime of calculation in s (scalar)
Rp	distance of fieldpoint in m (scalar)
vf	upward-propagating return-front-wave speed in m/s (scalar)
rsmt	return stroke model type (1: TL, 2: MTLE, 3: TCS, 4: DU)
la	coefficient for MTLE model in m (scalar)
taub	breakdown coefficient for DU model in s (scalar)
taub	corona coefficient for DU model in s (scalar)
I0t	current at channel origin in A (vector 1:m)
I0tb	breakdown current for DU model in A (vector 1:m)
I0tc	corona current for DU model in A (vector 1:m)
E	electrical field in V/m (vector 1:n)
Ee	electrostatic field component (vector 1:n)
Ei	electric induction field component (vector 1:n)
Er	electric radiation field component (vector 1:n)
Ed	electric discontinuity field component (vector 1:n)
B	magnetic field in T (vector 1:n)
Bi	magnetic induction field component (vector 1:n)
Br	magnetical radiation field component (vector 1:n)
Bd	magnetic discontinuity field component (vector 1:n)
tf	time of arrival of field at field point in s (scalar)

2.4 Description of the Program Modules

2.4.1 start

`start.sce` is used to load and execute `start.sci`. `start.sci` contains the function to provide a menu-driven calculation and visualization of the field and to load the required sci-files. (Listing A.1.1)
call:

```
exec("start.sce",-1)
```

2.4.2 func_par

contains the functions to define the parameters and to calculate the necessary vectors and the current at the point of origin. (Listing A.1.2)

enter_user

defines the input-function for entering the parameters manually. (Figure 2.3)
call:

```
[tend,ta,vf,Rp,typ,I0,T1,T2,n,la,taub,tauc]=enter_user()
```

enter_file

defines the input-function for importing the parameters from the file def_ground.txt. (Listing of the plain text file A.1.5)
call:

```
[tend,ta,vf,Rp,typ,I0,T1,T2,n,la,taub,tauc] = enter_file()
```

calc_par

defines the function to calculate the time-vector, the channel-vector and the current at the point of origin.
call:

```
[tv,cv,I0t,I0tb,I0tc]=calc_par(tend,ta,vf,Rp,typ,I0,T1,T2,n
    )
```

requirements:
`'func_rsc.sci'`

2.4.3 func_rsc

contains the functions needed to calculate the current at the point of origin as a sum of currents. (Listing A.1.2)

heidler

defines the function to calculates the values according to a Heidler function.
call:

```
I0t=heidler(I0,T1,T2,n,tv)
```

dexp

defines the function to calculates the values according to a double exponential function.
call:

```
I0t=dexp(I0,T1,T2,tv)
```

2.4.4 field

is the module to calculate the total fields and its components. (Listing A.1.3)
call:

```
[E,Ee,Ei,Er,Ed,B,Bi,Br,Bd,tf]=field(I0t,I0tb,I0tc,tv,tend,
    cv,vf,rsmt,la,taub,tauc,Rp)
```

requirements:
`'e_field.sci', 'i_field.sci', 'r_field.sci', 'd_field.sci'`

2.4.5 e_field

defines the function to calculate the electrostatic component of the field. (Listing A.1.3)
call:

```
Ee=e_field(I0t,I0tb,I0tc,tv,cv,vf,rsmt,la,taub,tauc,Rp)
```

2.4.6 i_field

defines the function to calculate the induction components of the field. (Listing A.1.3)
call:

```
[Ei,Bi]=i_field(I0t,I0tb,I0tc,tvi,cv,vf,rsmt,la,taub,tauc,
    Rp)
```

2.4.7 r_field

defines the function to calculate the radiation components of the field. (Listing A.1.3)
call:

```
[Er,Br]=r_field(I0t,I0tb,I0tc,tvi,cv,vf,rsmt,la,taub,tauc,
    Rp)
```

2.4.8 d_field

defines the function to calculate the discontinuity components of the field for the DU model. (Listing A.1.3)
call:

```
[Ed,Bd]=d_field(I0t,tvi,cv,vf,Rp)
```

2.4.9 func_plot

contains the functions to plot the field. (Listing A.1.4)

plot_field

defines the function to plot the field with all components.
call:

```
[]=plot_field(E,Ee,Ei,Er,Ed,B,Bi,Br,Bd,tend,tv,tzoom,fn)
```

plot_all_field

defines the function to plot the fields of all models.
call:

```
[]=plot_all_field(E_tl,E_mtle,E_tcs,E_du,B_tl,B_mtle,B_tcs,
    B_du,tend,tv,tzoom,fn,la,taub,tauc)
```

2.4.10 func_save

contains the functions to save the parameters and the field values in a plain text file. (Listing A.1.4)

save_field

defines the function to save the field with all components.
call:

```
[]=save_field(E,Ee,Ei,Er,Ed,B,Bi,Br,Bd,tf,tv,vf,rsmt,la,
    taub,tauc,Rp)
```

save_all_field

defines the function to save the fields of all models.
call:

```
[]=save_all_field(E_tl,E_mtle,E_tcs,E_du,B_tl,B_mtle,B_tcs,
    B_du,tf,tv,vf,la,taub,tauc,Rp)
```

Chapter 3

Lightning Strike to a Tall Object

The program calculates the remote field of in case of a lightning strike to a tall object. The used model type is the transmission line. The user can define the parameter via keyboard or file. The results are shown as plots and are storable as plain text file.

3.1 Basics

3.1.1 Current and the Time-Vector

The time-vector is not affected by the presence of a tall structure and the short-circuit-current I_{SC} is built equally to the current $I(0,t)$ (see 2.1.1). [Baba and Rakov, 2005] expressed that the correlation between the short-circuit-current used for the strike to a tall object and the undisturbed current used for the strike to the ground is

$$I_{SC} = 2 \cdot I(0,t) \tag{3.1}$$

to effect the same way, assumed perfect reflection on the top of the tower.

3.1.2 Discretization of the Channel

When a lightning strikes a tall object there are two areas of different propagation speeds, the tall object with c_0 and the channel with v_f. So the discretization of the lightning channel and the tall object must be divided in this two parts.

Tall Object

The discretization of the tall structure is not as easy as it seems. The runtime method is too tricky as there are different result for upward and downward propagating currents. But with the assumption $H \ll R_p$, where H is the height of the tall object, the propagation direction of the reflected current wave and the runtime differences between different heights can be negligible. The solution of the simplified approach are equidistant segments with

$$\Delta z' = \Delta t \cdot c_0 \tag{3.2}$$

It is important to end the modeled tall structure with a whole segment. This results in a height H' which is i.g. different to the real height H. So the number of segments with the lowest height difference $\Delta H = \min|(H - H')|$ to the real height H is chosen. The difference to the real object is assumed to be without influence.

Lightning Channel

The channel discretisation is the same as in section 2.1.2, the only difference being that the starting height is not at ground level but at the adjusted height H' of the modeled tall object. The runtime of the field from to striking point to the point the field is calculated at changes to

$$t_B = \frac{\sqrt{R_p^2 + H^2}}{c} \tag{3.3}$$

and the runtime of the current to

$$\mathbf{t_C} = \frac{\mathbf{z}' - H}{c} \tag{3.4}$$

Plug in the runtime equation 2.4 and solve the quadratic equation to build the channel-vector.

$$\mathbf{z}'^2 \cdot \left[\left(\frac{c}{v_f}\right)^2 - 1\right] - \mathbf{z}' \cdot 2\frac{c}{v_f}(c\mathbf{t} + \sqrt{R_p^2 + H^2} + H \cdot \frac{c}{v_f}) + \left[(c\mathbf{t} + \sqrt{R_p^2 + H^2} + H \cdot \frac{c}{v_f})^2 - R_p^2\right] = 0 \tag{3.5}$$

3.2 Current Distribution and Field Calculation

3.2.1 Current Distribution

The current at a height is a sum of the injected current wave I_{SC} and the reflected respectively transmitted waves. Considering that only the TL model is calculated the current distribution is explicitly calculated. The equations 1.10 and 1.11 are given by [Baba and Rakov, 2005].

3.2.2 Field Calculation

The method of the field calculation is the same as discussed in section 2.2.2.

3.3 Using the Program

Due to the simplification that only the TL-model is used, there is no reason to implement a menu. So the program is block by block operation of the modules. Preliminaries like changing the working directory have to be done as discussed in 2.3.

3.4 Defining the Parameters

In addition to the parameters discussed in section 2.3.1 the tall object needs to be defined. The height of the object is the basic information. Also important are the current reflection coefficients. The number of reflections calculated is important for the calculation-time of the current distribution. 20 reflections should be sufficient.

3.5 Variables

At the end of the program all variables are still kept in the memory. So it may be useful to know them (Table 3.1).

3.6 Description of the Program Modules

3.6.1 start

starts the calculation and visualization of the field and loads the required sci-files. (Listing A.2.1)
load and call:

```
exec("start.sce",-1)
```

3.6.2 def

contains the functions for defining the parameters and to calculate the time-vector as well as the short-circuit-current. It also plots the current and its derivate. (Listing A.2.2)
call:

```
[tv,H,rhotop,rhobot,nstop,vf,Rp,I0] = def()
```

requirements:
'rsc.sci'

Table 3.1: Strike to a tall object: List of variables

Name	Description
tv	time-vector of current (vector 1:m)
zvt	height-vector of tall object (vector 1:k)
zvc	height-vector of channel (vector 1:l)
zv	merged height-vector (vector 1:n)
tend	endtime of calculation in s (scalar)
nstop	number of reflections calculated
Rp	distance to field-point in m (scalar)
vf	upward-propagating return-front-wave speed in m/s (scalar)
rhotop	reflection coefficient at the top (scalar)
rhobottom	reflection coefficient at the bottom (scalar)
I0	undisturbed current in A (vector 1:m)
Izt	current distribution in A (matrix m:n)
E	electrical field in V/m (vector 1:m)
Ee	electrostatic field component (vector 1:m)
Ei	electric induction field component (vector 1:m)
Er	electric radiation field component (vector 1:m)
B	magnetic field in T (vector 1:m)
Bi	magnetic induction field component (vector 1:m)
Br	magnetical radiation field component (vector 1:m)
tf	time of arrival of field at field point in s (scalar)

3.6.3 rsc

contains the functions needed for calculating the short-circuit-current as a sum of currents. (Listing A.2.2)

heidler

defines the function for calculating the values according to the so called Heidler-function.
call:

```
I0t=heidler(I0,T1,T2,n,tv)
```

dexp

defines the function for calculating the values according to a double exponential function.
call:

```
I0t=dexp(I0,T1,T2,tv)
```

3.6.4 heightvector

calculates the hight-vectors of the tall object and the channel. Further on it merge the two. (Listing A.2.2)
call:

```
[zvc,zvt] = heightvector(tv,Rp,H,vf)
```

3.6.5 distribution

calculates the current distribution in the tall object and the channel. (Listing A.2.3)
call:

```
[Izt,zv] = distribution(tv,zvt,zvc,I0,rhotop,rhobot,nstop,
    vf)
```

3.6.6 field

calculates the fields and its components. (Listing A.2.4)
call:

```
[E,Er,Ei,Ee,B,Br,Bi,tf] = field(Izt,zv,tv,Rp)
```

3.6.7 plot_field

plots the field with all components.
call:

```
plot_field(E,Ee,Ei,Er,B,Bi,Br,tv,tzoom,fn,vf,Rp,Hi)
```

3.6.8 save_field

saves the parameters and the field values as well as their components in a plain text file. (Listing A.2.4)
call:

```
save_field(E,Ee,Ei,Er,B,Bi,Br,tf,tv,H,rhotop,rhobot,vf,Rp)
```

Chapter 4

Evaluation of the Programs

To evaluate the properness of the programs a comparison with published results is the easiest way. The comparison shows that there is no difference between the calculated field by the Scilab programs and the published results using the same current parameters.

4.1 Strike to the Ground

4.1.1 Comparison with [Nucci et al., 1990]

[Nucci et al., 1990] used the TL, MTLE and TCS model to calculate their results. The undisturbed current is current N1 (see Table 1.1 and Figure 1.1). The speed of the upward propagation retrun-stroke-front is $v_f = 1.3 \cdot 10^8 \frac{\mathrm{m}}{\mathrm{s}}$. For the MTLE model is $\lambda = 2000$m and for the DU model is $\tau_B = \tau_C = 0.1\mu$s.

Since no numerical data values are available from [Nucci et al., 1990], the comparison is graphically. The Figures 4.1 to 4.6 show the results.

4.1.2 Comparison with [Diendorfer and Uman, 1990]

[Diendorfer and Uman, 1990] used two different undisturbed currents to calculate their results. The current used for the comparison is D1 (see Table 1.1 and Figure 1.2). The speed of the upward propagation retrun-stroke-front $v_f = 1.3 \cdot 10^8 \frac{\mathrm{m}}{\mathrm{s}}$. The breakdown discharge time constant is $\tau_{BD} = 0.6\mu$s and the corona discharge time constant is $\tau_C = 5\mu$s.

Since no numerical data values are available from [Diendorfer and Uman, 1990], the comparison is graphically. The Figures 4.7 to 4.10 show the results.

4.2 Strike to a Tall Object

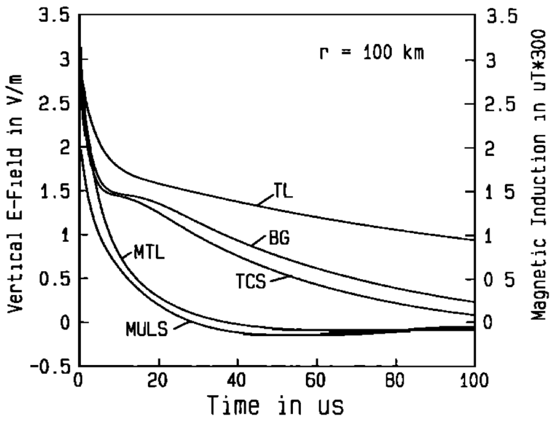
[Pavanello et al., 2007] is the paper to be compared. The speed of the upward propagation retrun-stroke-front $v_f = 1.5 \cdot 10^8 \frac{\mathrm{m}}{\mathrm{s}}$. Since no numerical data values are available from [Pavanello et al., 2007], the comparison is graphically.

4.2.1 Current and Current Distribution

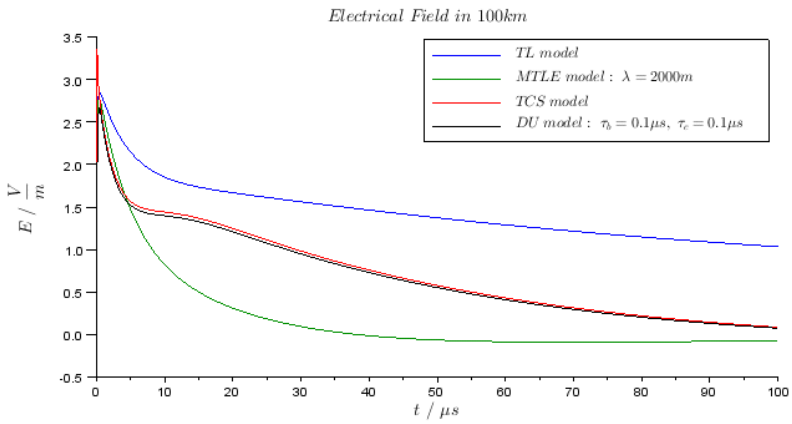
[Pavanello et al., 2007] gives the current at the bottom of the tall object as well as the current at the top. Also the current distribution at different times is given. The Figures 4.11 and 4.12 show the identicalness.

4.2.2 Fields

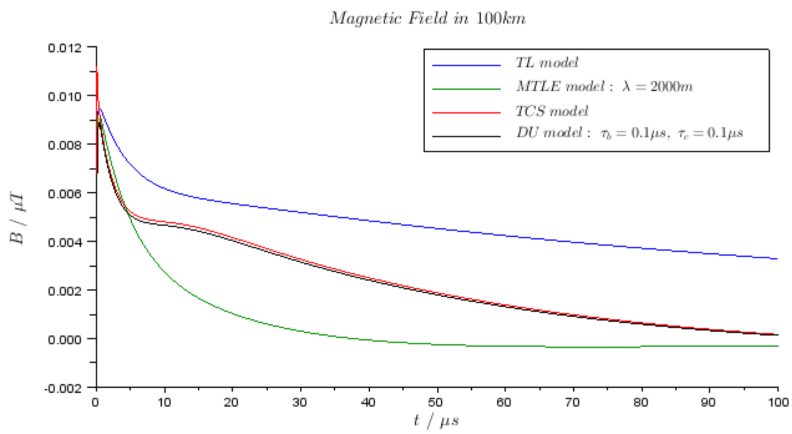
The computed fields of [Pavanello et al., 2007] are equal to the fields calculated with this program. The Figures 4.13 to 4.14 show the results.



(a) Published Fields

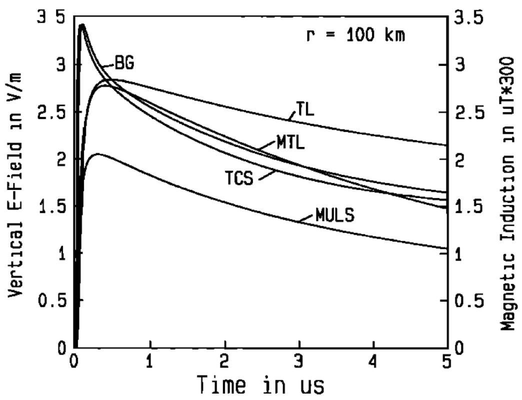


(b) Calculated Electrical Field

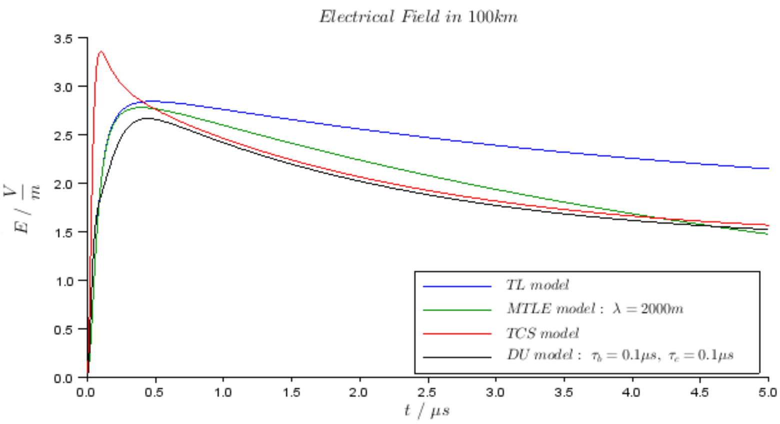


(c) Calculated Magnetic Filed

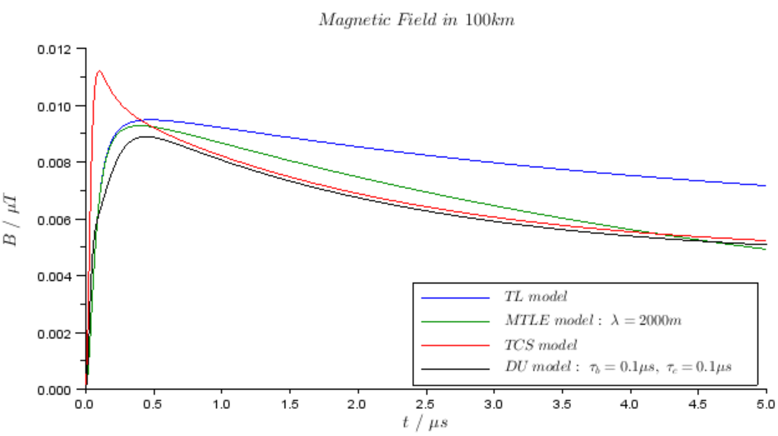
Figure 4.1: Comparison of the calculated fields with published results by [Nucci et al., 1990]: Distance 100km, Duration 100µs



(a) Published Fields

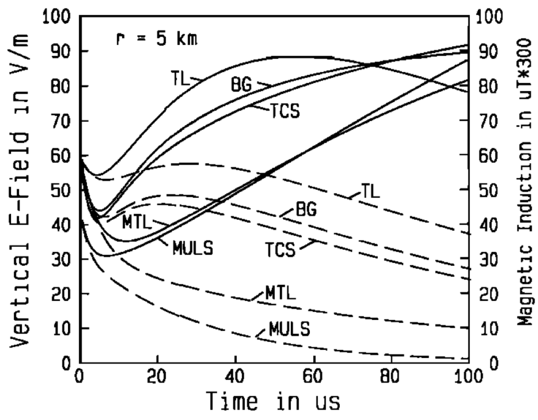


(b) Calculated Electrical Field

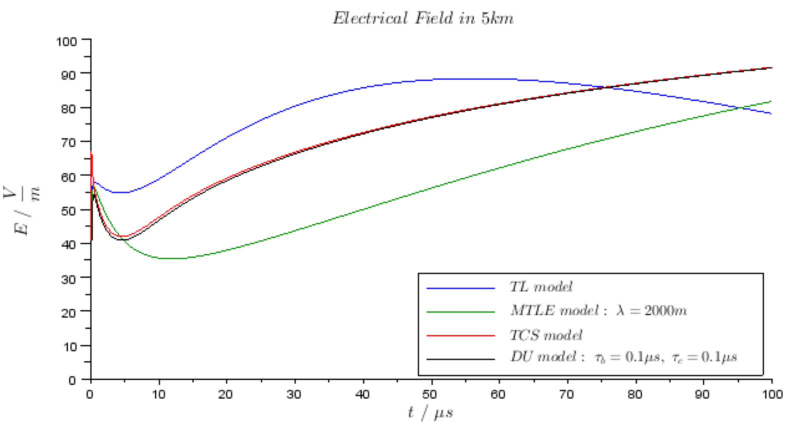


(c) Calculated Magnetic Filed

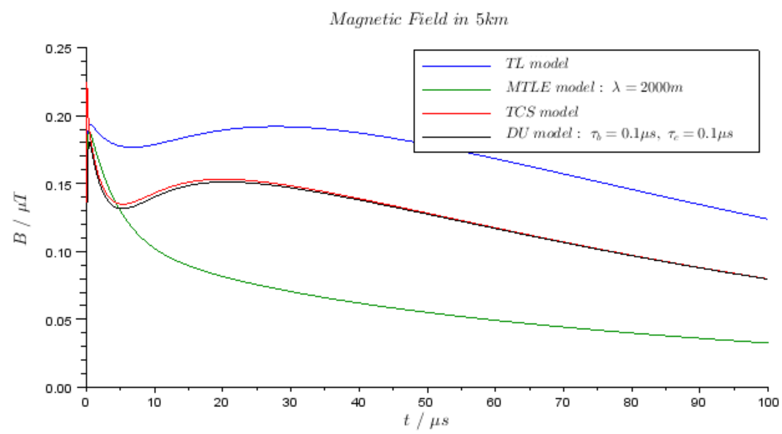
Figure 4.2: Comparison of the calculated fields with published results by [Nucci et al., 1990]: Distance 100km, Duration 5μs



(a) Published Fields

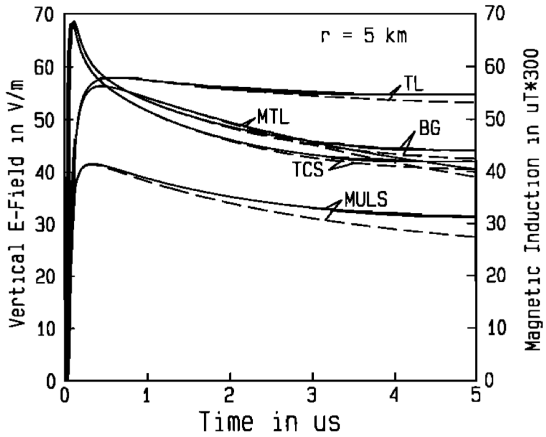


(b) Calculated Electrical Field

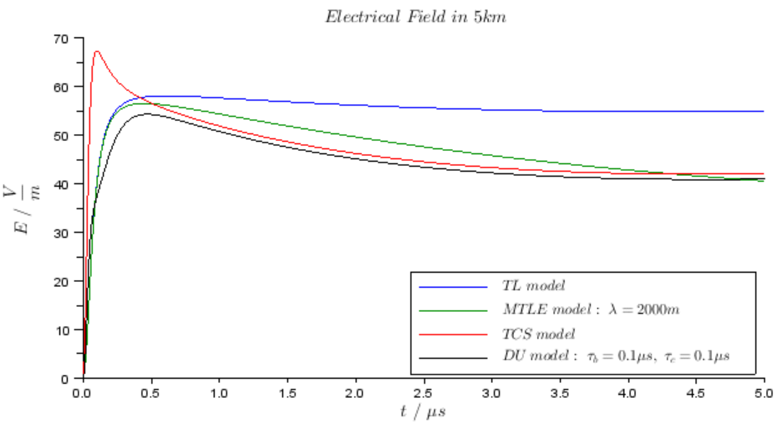


(c) Calculated Magnetic Filed

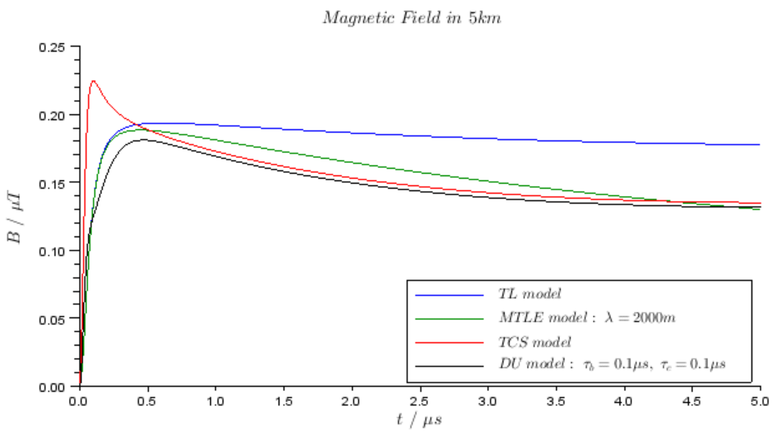
Figure 4.3: Comparison of the calculated fields with published results by [Nucci et al., 1990]: Distance 5km, Duration 100µs



(a) Published Fields

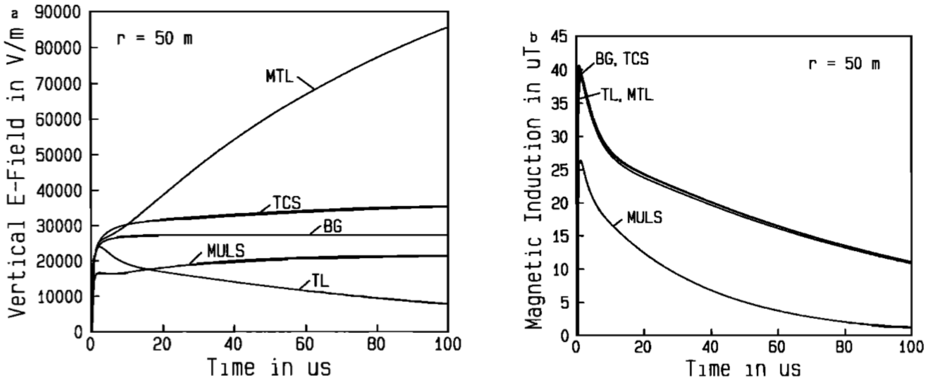


(b) Calculated Electrical Field

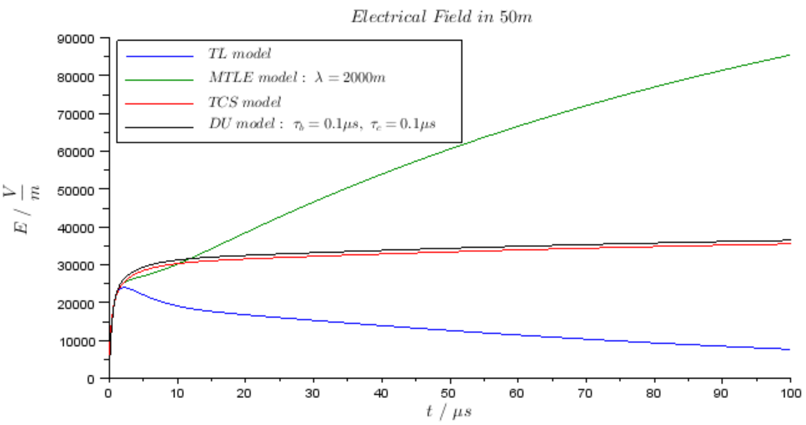


(c) Calculated Magnetic Filed

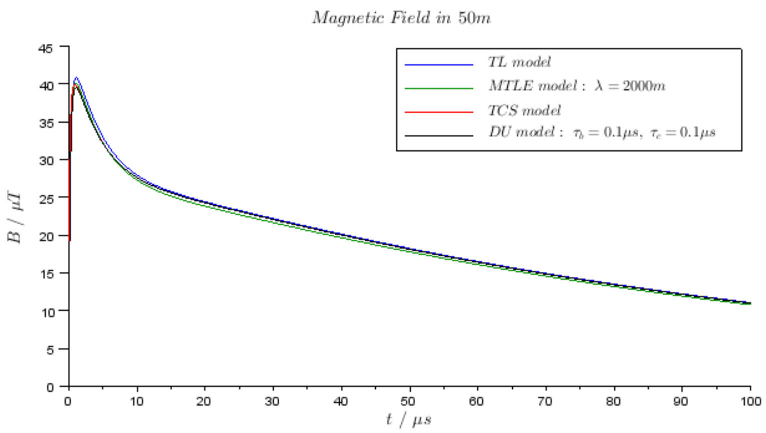
Figure 4.4: Comparison of the calculated fields with published results by [Nucci et al., 1990]: Distance 5km, Duration 5µs



(a) Published Fields

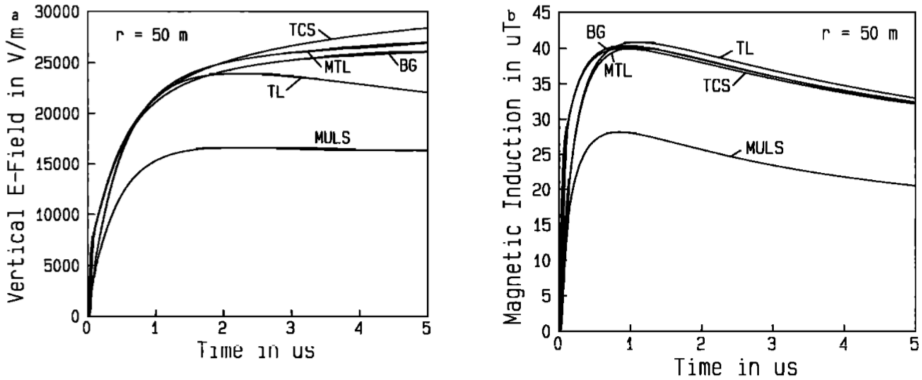


(b) Calculated Electrical Field

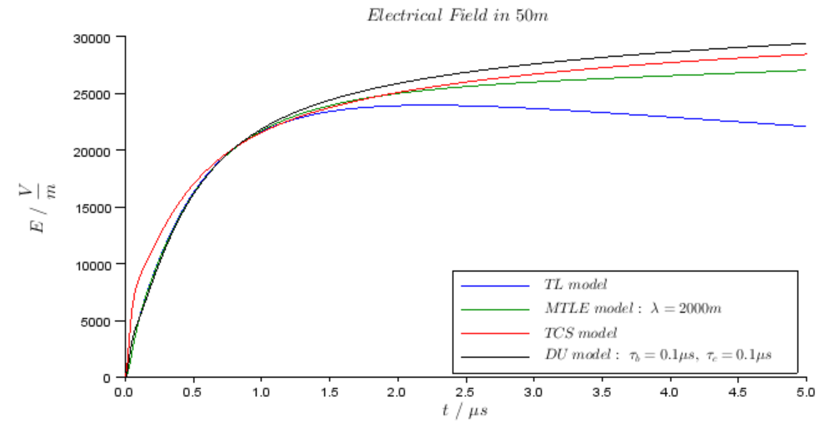


(c) Calculated Magnetic Filed

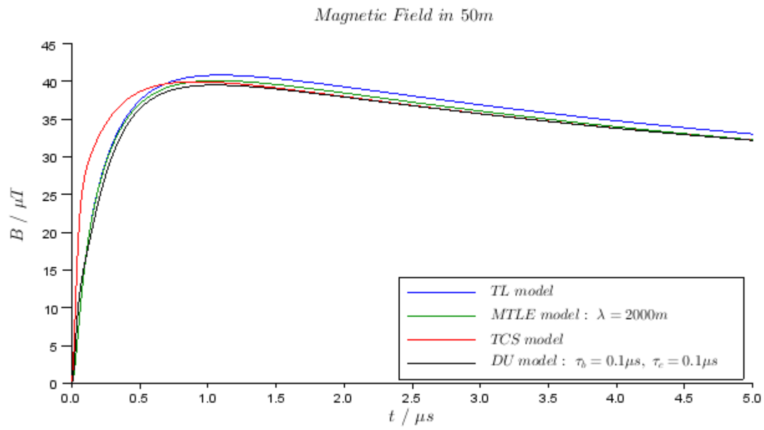
Figure 4.5: Comparison of the calculated fields with published results by [Nucci et al., 1990]: Distance 50m, Duration 100µs



(a) Published Fields

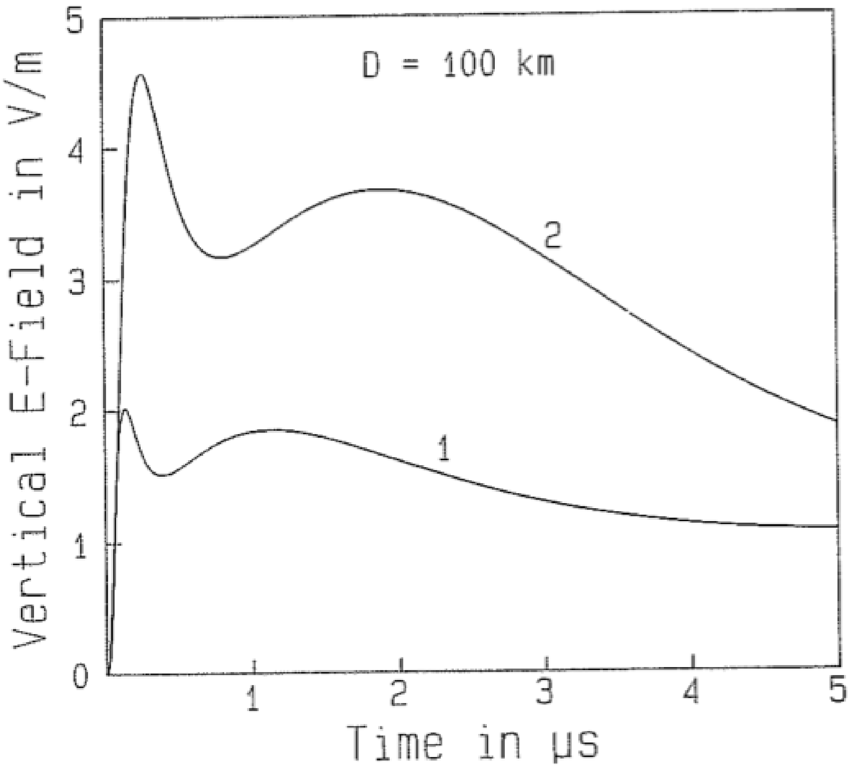


(b) Calculated Electrical Field

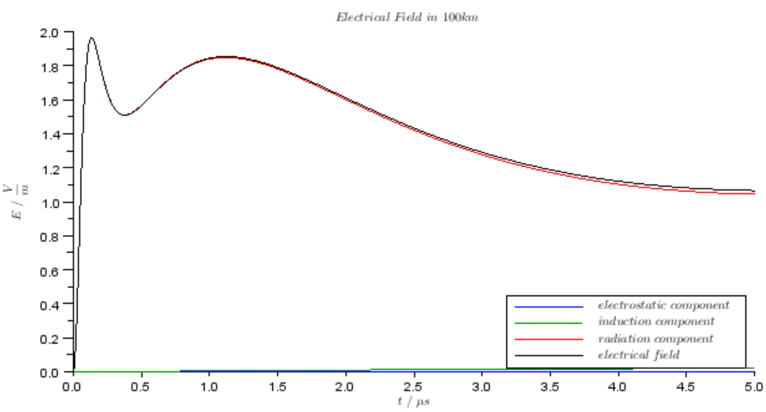


(c) Calculated Magnetic Filed

Figure 4.6: Comparison of the calculated fields with published results by [Nucci et al., 1990]: Distance 50m, Duration 5µs

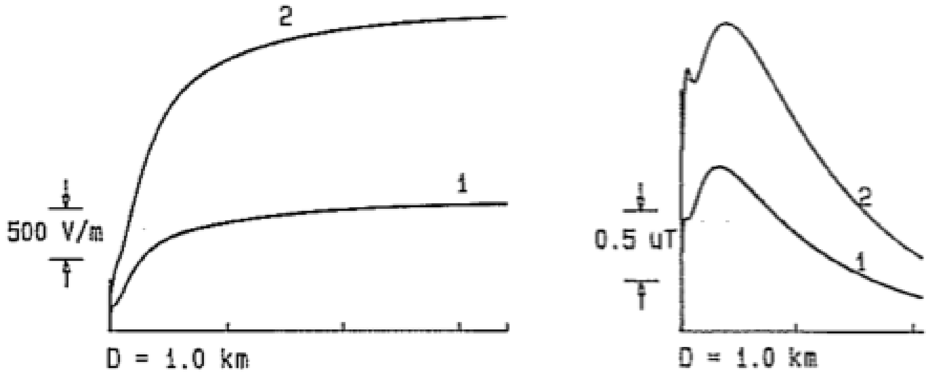


(a) Published Fields

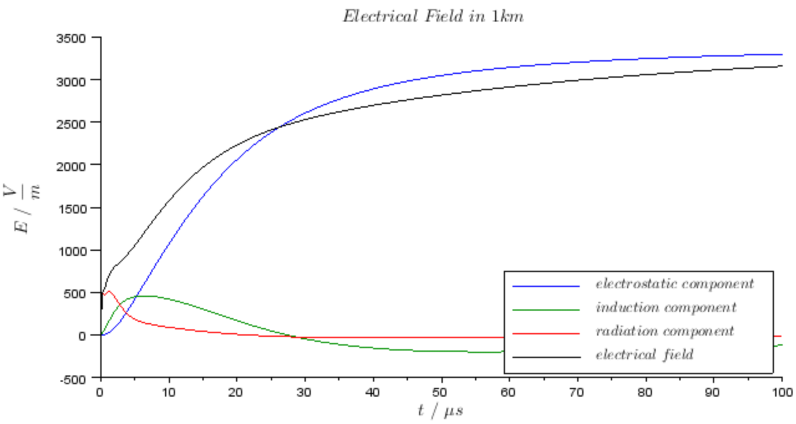


(b) Calculated Electrical Field

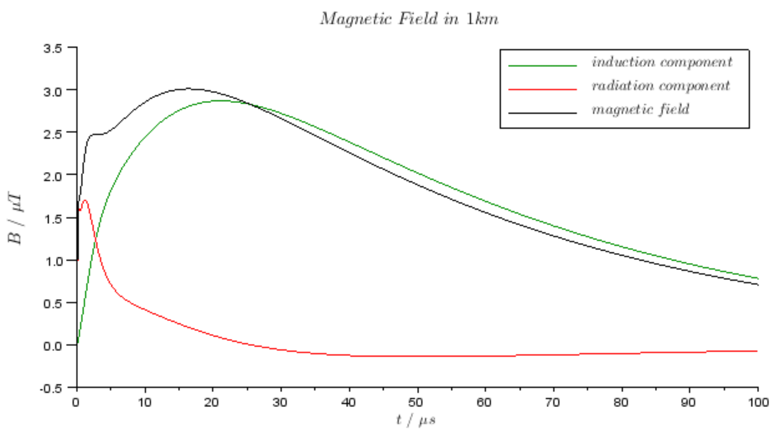
Figure 4.7: Comparison of the calculated fields with published results by [Diendorfer and Uman, 1990]: Distance 100km, Duration 5µs



(a) Published Fields

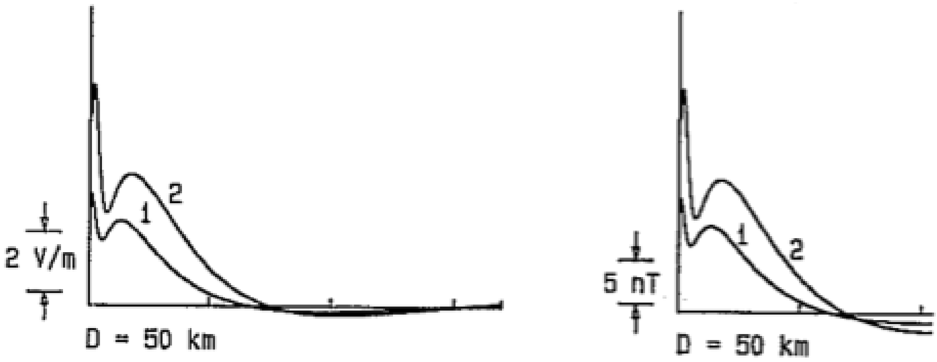


(b) Calculated Electrical Field

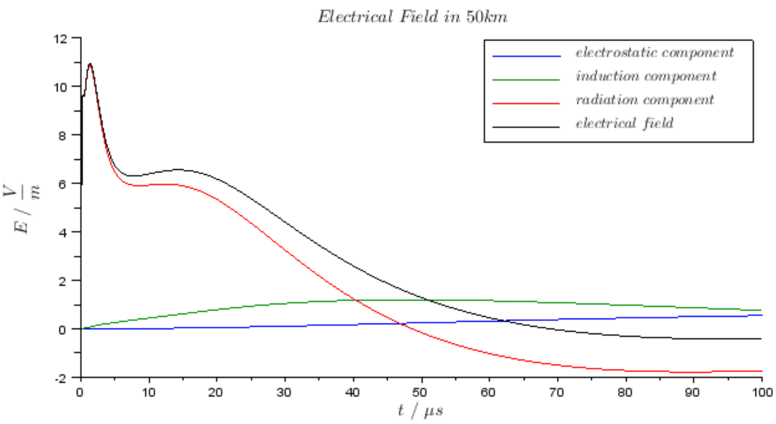


(c) Calculated Magnetic Filed

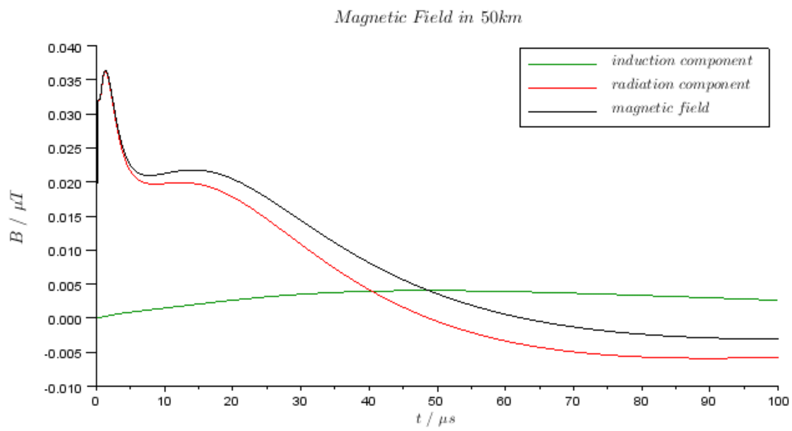
Figure 4.8: Comparison of the calculated fields with published results by [Diendorfer and Uman, 1990]: Distance 1km, Duration 100µs



(a) Published Fields

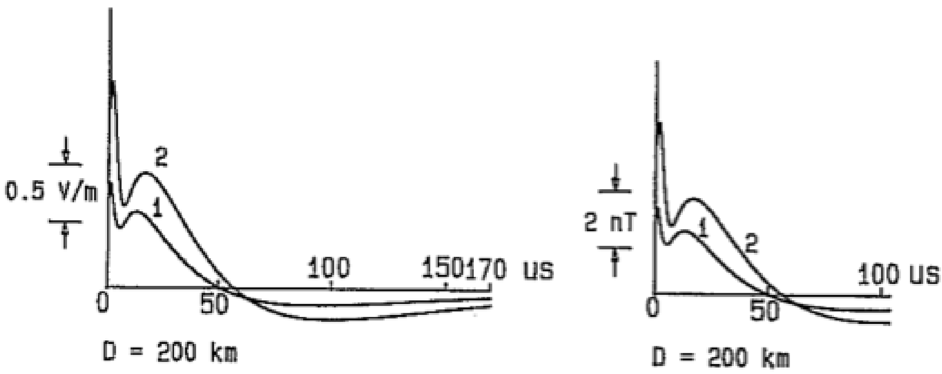


(b) Calculated Electrical Field

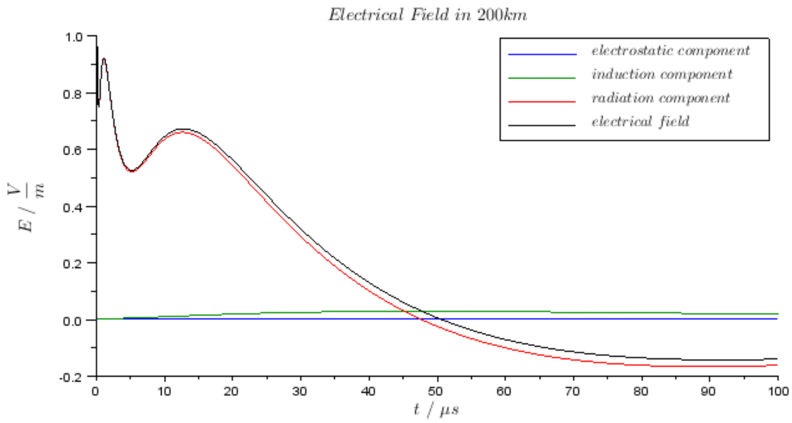


(c) Calculated Magnetic Filed

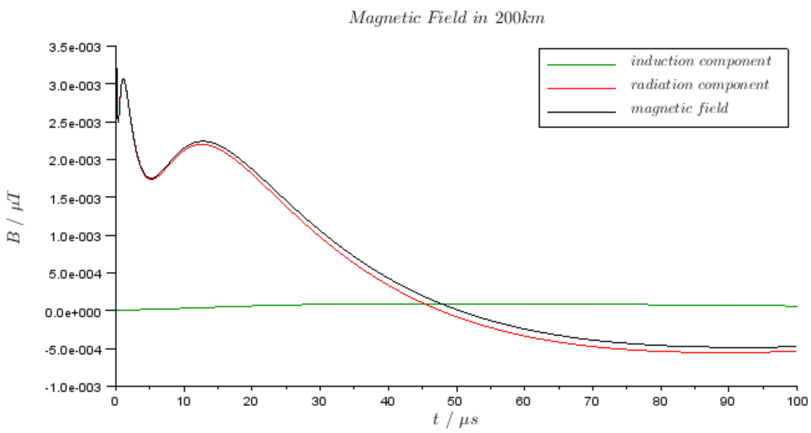
Figure 4.9: Comparison of the calculated fields with published results by [Diendorfer and Uman, 1990]: Distance 50km, Duration 100µs



(a) Published Fields

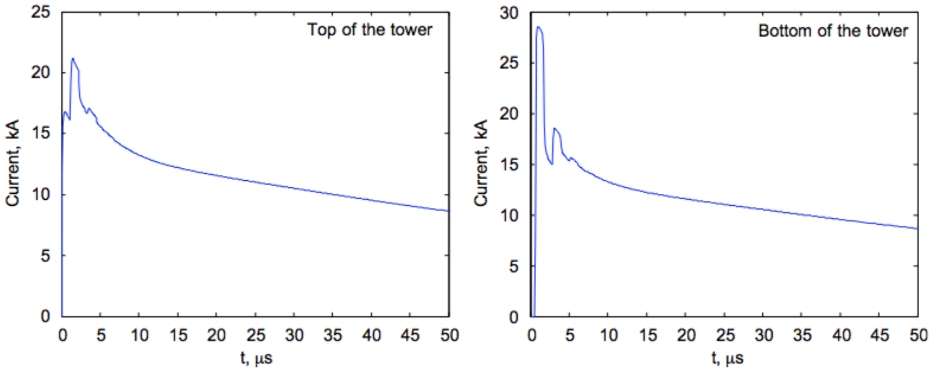


(b) Calculated Electrical Field

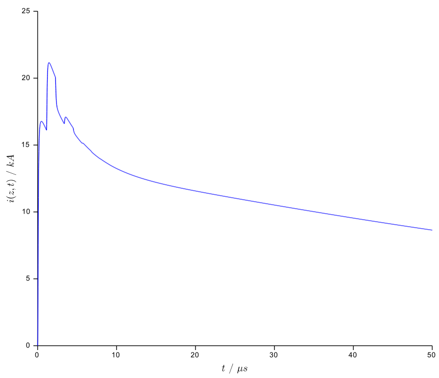


(c) Calculated Magnetic Filed

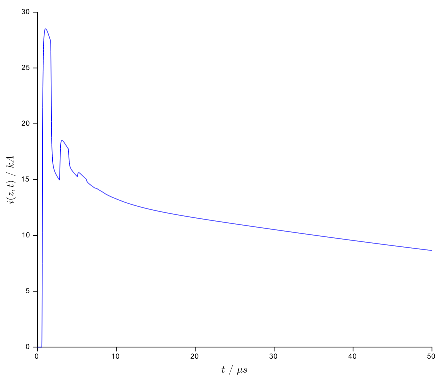
Figure 4.10: Comparison of the calculated fields with published results by [Diendorfer and Uman, 1990]: Distance 200km, Duration 100µs



(a) Published Current at the Top and the Bottom

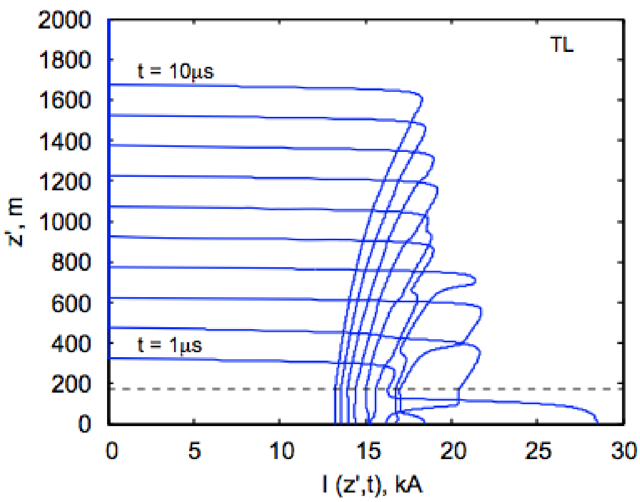


(b) Calculated Current at the Top

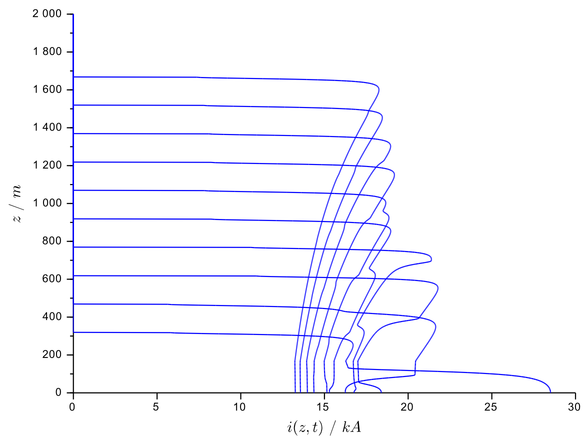


(c) Calculated Current at the Bottom

Figure 4.11: Comparison of the calculated current at the top and the bottom of the tall object with published results by [Pavanello et al., 2007]

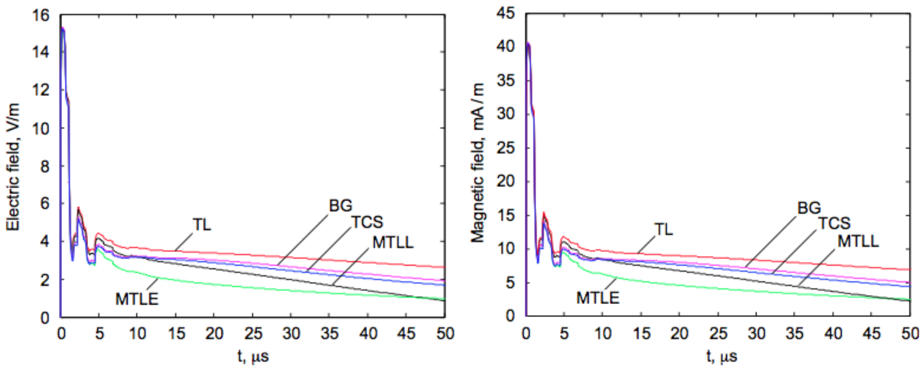


(a) Published Current Distribution

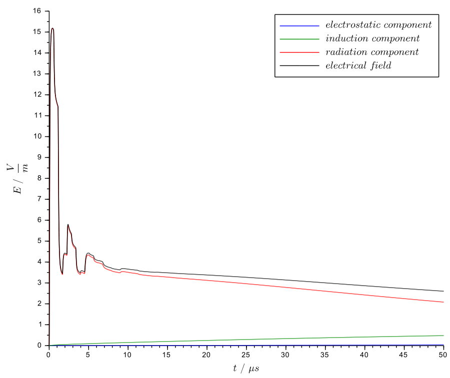


(b) Calculated Current Distribution

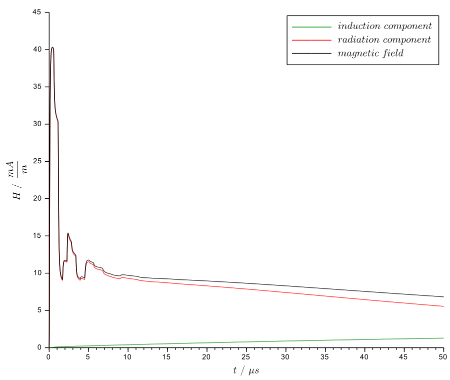
Figure 4.12: Comparison of the calculated Current Distribution with Published Results by [Pavanello et al., 2007]



(a) Published Fields

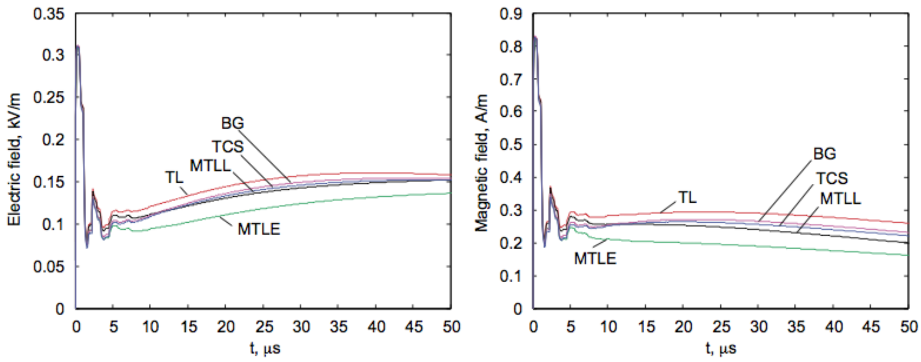


(b) Calculated Electrical Field

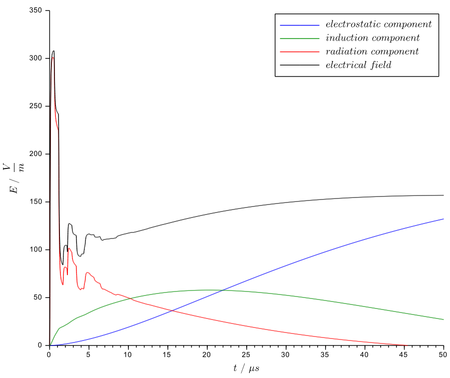


(c) Calculated Magnetically Field

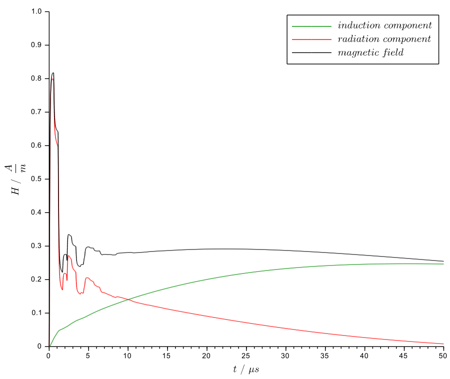
Figure 4.13: Comparison of the Calculated Fields in 100km Distance with Published Results by [Pavanello et al., 2007]



(a) Published Fields



(b) Calculated Electrical Field



(c) Calculated Magnetically Field

Figure 4.14: Comparison of the Calculated Fields in 5km Distance with Published Results by [Pavanello et al., 2007]

Chapter 5

Summary and Outlook

5.1 Summary

The benefit of open source software under a GPL or similar license in academic research is massive. It is free to use, and this applies even to copying and modifying. But there is a drawback. The first concern of communities to develop software packages like Scilab is to find an opinion to cost intensive packages like Matlab. For a long term use there may be a problem. First of all open source packages disappear often as fast as they appear. Secondly the support often depends on the good will of the developer or the community. And last the further development over a long time further away also the affinity to the substituted software package. The first two problems are not directed at Scilab, but the third is noticably. But altogether Scilab is a good alternative to Matlab and its works properly and solidly.

5.2 Outlook

The program uses numerical computation of the lightning electromagnetic fields without consideration the equations of the current, the calculation is based on. This results in the possibility to use measured currents for computing fields. First trials have been successful. So it would be interesting to compare a considerable number of measured fields with the computed ones.

Appendix A

Listings

A.1 Lightning Strike to the Ground

A.1.1 Startmodule

ground/start.sce

```
// v7.1

// startmodule for calculation of fields for a return stroke

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

  funcprot(0)

  stacksize('max')

  exec ("start.sci",-1)

  start
```

ground/start.sci

```
// v7.1

// function to start return stroke model calculation

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname start

// input: none

// output: none

function start()

  exec ("func_par.sci",-1)
  exec ("field.sci",-1)
  exec ("func_plot.sci",-1)
  exec ("func_save.sci",-1)

  par=0
  fie=0
  menu=0
  while menu<>9
    write(%io(2),"   ")
    write(%io(2),"   ")
```

```
write(%io(2),"    ")
write(%io(2),"    ")
write(%io(2),"    ")
write(%io(2),"************************************")
write(%io(2),"*                                  *")
write(%io(2),"*  Lightning Strike to the Ground  *")
write(%io(2),"*                                  *")
write(%io(2),"************************************")
write(%io(2),"    ")
write(%io(2)," 1 ... enter parameter via keyborad")
write(%io(2)," 2 ... read parameter from file")
write(%io(2),"    ")
write(%io(2)," 3 ... calculate one model with all components")
write(%io(2)," 4 ... save field of one model with all components")
write(%io(2),"    ")
write(%io(2)," 5 ... calculate all models")
write(%io(2)," 6 ... save field of all models")
write(%io(2),"    ")
write(%io(2)," 8 ... interrupt")
write(%io(2)," 9 ... end")
write(%io(2),"    ")
write(%io(2),"------------------------------")
write(%io(2),"    ")
menu=input("Your choice please: ")
write(%io(2),"    ")
select menu,
  case 1 then
    [tend,ta,vf,Rp,typ,I0,T1,T2,n,la,taub,tauc]=enter_user()
    [tv,cv,I0t,I0tb,I0tc]=calc_par(tend,ta,vf,Rp,typ,I0,T1,T2,n)
    par=1
    write(%io(2),"    ")
  case 2 then
    [tend,ta,vf,Rp,typ,I0,T1,T2,n,la,taub,tauc]=enter_file()
    [tv,cv,I0t,I0tb,I0tc]=calc_par(tend,ta,vf,Rp,typ,I0,T1,T2,n)
    par=1
    write(%io(2),"    ")
  case 3 then
    if par==1 then
      write(%io(2),"model definition")
      write(%io(2),"-----------------")
      write(%io(2),"    ")
      rsmt=input("    model type: 1 for TL, 2 for MTLE, 3 for TCS, 4 for DU,
         other to break: ")
      write(%io(2),"  ")
      write(%io(2),"figure of field")
      write(%io(2),"---------------")
      write(%io(2),"  ")
      fig=input("   Draw figure of field? 1 for yes, other for no: ")
      write(%io(2),"  ")
      if fig==1
        tzoom=input("    zoom at tmax in us (0 for none): ")
        write(%io(2),"  ")
        fn=input("    start figure number: ")
      end
      [E,Ee,Ei,Er,Ed,B,Bi,Br,Bd,tf]=field(I0t,I0tb,I0tc,tv,tend,cv,vf,rsmt,la
         ,taub,tauc,Rp)
      if fig==1, plot_field(rsmt,E,Ee,Ei,Er,Ed,B,Bi,Br,Bd,tend,tv,tzoom,fn),
         end
      fie=1
    else
      write(%io(2),"input parameter first!")
    end
    write(%io(2),"    ")
  case 4 then
    if fie==1 then
      save_field(E,Ee,Ei,Er,Ed,B,Bi,Br,Bd,tf,tv,vf,rsmt,la,taub,tauc,Rp)
    else
      write(%io(2),"calculate one field first")
      write(%io(2),"    ")
    end
  case 5 then
    if par==1 then
      write(%io(2),"figure of field")
      write(%io(2),"---------------")
      write(%io(2),"    ")
      fig1=input("    Draw figure of fields? 1 for yes, other for no: ")
      fig2=0
      write(%io(2),"  ")
      if fig1==1
        tzoom=input("    zoom at tmax in us (0 for none): ")
        write(%io(2),"  ")
        fn=input("    start figure number: ")
        write(%io(2),"  ")
```

```
            fig2=input("   Draw also single fields with components: 1 for yes,
                other for no: ")
            write(%io(2),"  ")
          end
          write(%io(2),"TL model:")
          write(%io(2),"---------")
          [E_tl,Ee,Ei,Er,Ed,B_tl,Bi,Br,Bd,tf]=field(I0t,I0tb,I0tc,tv,tend,cv,vf
              ,1,la,taub,tauc,Rp)
          if fig2==1, plot_field(1,E_tl,Ee,Ei,Er,Ed,B_tl,Bi,Br,Bd,tend,tv,tzoom,
              fn+2), end
          write(%io(2),"MTLE model:")
          write(%io(2),"-----------")
          [E_mtle,Ee,Ei,Er,Ed,B_mtle,Bi,Br,Bd,tf]=field(I0t,I0tb,I0tc,tv,tend,cv,
              vf,2,la,taub,tauc,Rp)
          if fig2==1, plot_field(2,E_mtle,Ee,Ei,Er,Ed,B_mtle,Bi,Br,Bd,tend,tv,
              tzoom,fn+6), end
          write(%io(2),"TCS model:")
          write(%io(2),"----------")
          [E_tcs,Ee,Ei,Er,Ed,B_tcs,Bi,Br,Bd,tf]=field(I0t,I0tb,I0tc,tv,tend,cv,vf
              ,3,la,taub,tauc,Rp)
          if fig2==1, plot_field(3,E_tcs,Ee,Ei,Er,Ed,B_tcs,Bi,Br,Bd,tend,tv,tzoom
              ,fn+10), end
          write(%io(2),"DU model:")
          write(%io(2),"---------")
          [E_du,Ee,Ei,Er,Ed,B_du,Bi,Br,Bd,tf]=field(I0t,I0tb,I0tc,tv,tend,cv,vf
              ,4,la,taub,tauc,Rp)
          if fig2==1, plot_field(4,E_du,Ee,Ei,Er,Ed,B_du,Bi,Br,Bd,tend,tv,tzoom,
              fn+14), end
          if fig1==1, plot_all_field(E_tl,E_mtle,E_tcs,E_du,B_tl,B_mtle,B_tcs,
              B_du,tend,tv,tzoom,fn,la,taub,tauc), end
          fie=2
        else
          write(%io(2),"input parameter first!")
        end
        write(%io(2),"   ")
      case 6 then
        if fie==2 then
          save_all_field(E_tl,E_mtle,E_tcs,E_du,B_tl,B_mtle,B_tcs,B_du,tf,tv,vf,
              la,taub,tauc,Rp)
        else
          write(%io(2),"calculate all fields first")
          write(%io(2),"   ")
        end
      case 8 then
        write(%io(2),"   ")
        write(%io(2),"   ---> type resume to continue <---")
        pause
      case 9 then
        write(%io(2),"   ")
        write(%io(2),"   !!! Have a good one !!!")
      else
        write(%io(2),"   ")
        write(%io(2),"   No menu entry !")
    end
  end
endfunction
```

A.1.2 Definition of Parameters

ground/func_par.sci

```
// v7.1

// functions for parameter input and calculation (enter_user, enter_file,
    calc_par)

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics


// functionname enter_user

// function to define parameter through keyboard

// input: none

// output: tend ... endtime of calculation in s
//         ta  ... number of timesteps
//         vf  ... front-wave speed in m/s
//         Rp  ... distance of fieldpoint in m
```

```
//          typ ... type of part i current (de: double-exponential, h. heidler)
//          I0, T1, T2, n ... parameter for part i of current according to typ(i)
//          la ... coefficient for MTLE model in m
//          taub ... breakdown coefficient for DU model in s
//          tauc ... corona coefficient for DU model in s

function [tend,ta,vf,Rp,typ,I0,T1,T2,n,la,taub,tauc]=enter_user()

  n=0

  write(%io(2),"   ")
  write(%io(2),"timevector for field definition")
  write(%io(2),"-------------------------------")
  write(%io(2),"   ")
  tend=input("   Endtime [us]: ")
  tend=tend*10^(-6)
  ta=input("   number of timesteps (best range: 1000 to 5000): ")
  write(%io(2),"   ")

  write(%io(2),"channel definition")
  write(%io(2),"------------------")
  write(%io(2),"   ")
  vf=input("   upward-propagating return-front-wave speed in 10^8 m/s: ")
  vf=vf*10^8
  write(%io(2),"   ")

  write(%io(2),"fieldpoint definition")
  write(%io(2),"---------------------")
  write(%io(2),"   ")
  Rp=input("   distance of fieldpoint in km: ")
  Rp=Rp*10^3
  write(%io(2),"   ")

  write(%io(2),"current I(0,t) definition")
  write(%io(2),"-------------------------")
  write(%io(2),"   ")
  a=input("   How many functions define the current? ")
  write(%io(2),"  ")
  for i=1:a
    b=input("   1 for Heidler, 2 for Double-Exponetial, 3 Nucci90, 4 DU90 other
         to break: ")
    write(%io(2),"   ")

    if b>2 then, break, end

    if b==1
      typ(i)="h"
      write(%io(2),"   Heidler:")
      write(%io(2),"   ")
      I0(i)=input("     I0 [kA]: ")
      I0(i)=I0(i)*10^3
      T1(i)=input("     current rise time constant T1 [us]: ")
      T1(i)=T1(i)*10^(-6)
      T2(i)=input("     current decay time constant T2 [us]: ")
      T2(i)=T2(i)*10^(-6)
      n(i)=input("     n []: ")
      write(%io(2),"   ")
    end

    if b==2
      typ(i)="de"
      write(%io(2),"   double exponential:")
      write(%io(2),"   ")
      I0(i)=input("     I0 [kA]: ")
      I0(i)=I0(i)*10^3
      T1(i)=input("     current decay time constant T1 [us]: ")
      T1(i)=T1(i)*10^(-6)
      T2(i)=input("     current rise time constant T2 [us]: ")
      T2(i)=T2(i)*10^(-6)
      write(%io(2),"   ")
    end
  end

  if b==3 // define current used by Nucci et al. (1990)
    typ(1)="h"
    I0(1)=-9.9*10^3
    T1(1)=0.072*10^(-6)
    T2(1)=5*10^(-6)
    n(1)=2
    typ(2)="de"
    I0(2)=-7.5*10^3
    T1(2)=100*10^(-6)
    T2(2)=6*10^(-6)
  end
```

```

  if b==4 // define current used by Diendorfer et al. (1990)
    typ(1)="h"
    I0(1)=-13*10^3
    T1(1)=0.73*10^(-6)
    T2(1)=3*10^(-6)
    n(1)=2
    typ(2)="h"
    I0(2)=-7*10^3
    T1(2)=5*10^(-6)
    T2(2)=50*10^(-6)
    n(2)=2
  end

  write(%io(2),"modelparameter definition")
  write(%io(2),"-------------------------")
  write(%io(2),"   ")
  write(%io(2),"input parameter if needed")
  la=input("   exponential decay coefficient in m for MTLE: ")
  write(%io(2),"   ")
  taub=input("   discharge time constant for breakdown current in us for DU: ")
  taub=taub*10^(-6)
  tauc=input("   discharge time constant for corona current in us for DU: ")
  tauc=tauc*10d^(-6)
  write(%io(2),"   ")

endfunction

// functionname enter_file

// function to define parameter through file

// input: none

// output: tend ... endtime of calculation in s
//         ta ... number of timesteps
//         vf ... front-wave speed in m/s
//         Rp ... distance of fieldpoint in m
//         typ ... type of part i current (de: double-exponential, h: heidler)
//         I0, T1, T2, n ... parameter for part i of current according to typ(i)
//         la ... coefficient for MTLE model in m
//         taub ... breakdown coefficient for DU model in s
//         tauc ... corona coefficient for DU model in s


function [tend,ta,vf,Rp,typ,I0,T1,T2,n,la,taub,tauc] = enter_file()

  write(%io(2),"   ")
  write(%io(2),"read from file")
  write(%io(2),"--------------")
  write(%io(2),"   ")
  rd=input("   read definition from file def_ground.txt? 1 for yes, other for no:
     ")
  write(%io(2),"   ")

  if rd==1 then
    errcatch(999,"continue")
    fid=mopen("def_ground.txt","r")
    if fid==-1 then
      write(%io(2),"   error: cannot open file def.txt!")
    else
      header=1
      while header==1
        val=mfscanf(1,fid,"%s")
        if val=="start_of_def" then, header=0, end
      end
      data=1
      while data==1
        val=mfscanf(1,fid,"%s")
        select val
          case "tend", tend=mfscanf(1,fid,"%f")*10^(-6),
          case "ta", ta=mfscanf(1,fid,"%f"),
          case "vf", vf=mfscanf(1,fid,"%f")*10^8,
          case "Rp", Rp=mfscanf(1,fid,"%f")*10^3,
          case "currents" then
            next=1
            i=1
            while next==1
              [dummy,typ(i),I0(i),T1(i),T2(i),n(i)]=mfscanf(1,fid,"%s %f %f %f %f
                 %f")
              if typ(i)=="x" then
                next=0
                typ=typ(1:(i-1))
                I0=I0(1:(i-1))
```

```
                T1=T1(1:(i-1))
                T2=T2(1:(i-1))
                n=n(1:(i-1))
              else
                I0(i)=I0(i)*10^3
                T1(i)=T1(i)*10^-6
                T2(i)=T2(i)*10^-6
                i=i+1
              end
            end
          case "la", la=mfscanf(1,fid,"%f"),
          case "taub", taub=mfscanf(1,fid,"%f")*10^(-6),
          case "tauc", tauc=mfscanf(1,fid,"%f")*10^(-6),
          case "end_of_def", data=0,
        end
      end
    end
    mclose(fid)
    errcatch(-1)
  end

endfunction



// functionname calc_par

// function for to calculate vectors and input-current

// input: tend ... endtime of calculation in s
//        ta ... number of timesteps
//        vf ... front-wave speed in m/s
//        Rp ... distance of fieldpoint in m
//        typ ... type of part i current (de: double-exponential, h: heidler)
//        I0, T1, T2, n ... parameter for part i of current according to typ(i)

// output: tv ... timevector
//         cv ... channelvector
//         I0t=I(0,t) ... current at channel origin
//         I0tb=Ib(0,t) ... breakdown current for DU model
//         I0tc=Ic(0,t) ... corona current for DU model


function [tv,cv,I0t,I0tb,I0tc]=calc_par(tend,ta,vf,Rp,typ,I0,T1,T2,n)

  exec ("func_rsc.sci",-1)

  c=299792458 // speed of light

  write(%io(2),"timevector calculation")
  write(%io(2),"-------------------------")
  write(%io(2),"   ")
  ts=tend/ta // timestep
  tv=0:ts:tend // build timevector
  write(%io(2),"     --> length of timestep in us:")
  write(%io(2),(ts/10^(-6)),"(7X,F10.5)")
  write(%io(2),"   ")

  write(%io(2),"channelvector calculation")
  write(%io(2),"-------------------------")
  write(%io(2),"   ")
  A=(c/vf).^2-1 // coefficient for quadratic equation
  B=-2*c/vf*(Rp+tv*c) // coefficient for quadratic equation
  C=2*Rp*tv*c+(tv*c).^2 // coefficient for quadratic equation
  cv=(-B-sqrt(B.^2-4*A*C))/(2*A) // calculate channelvector that current at cv
      effects on fieldpoint at time tv
  ca=length(cv)
  write(%io(2),"     --> number of channel segments:")
  write(%io(2),(ca-1),"(7X,F10.0)")
  write(%io(2),"     --> calculated height of channel in m:")
  write(%io(2),cv(ca),"(7X,F10.2)")
  write(%io(2),"   ")

  tvend=tend+max(cv)/vf+max(cv)/c+2*tv(2) // extended cause of DU and TCS model
  tv=0:tv(2):tvend // new extended timevector for current

  a=length(I0)
  I0t=zeros(tv)

  for i=1:a
    if typ(i)=="h"

      I(i,:)=heidler(I0(i),T1(i),T2(i),n(i),tv) // part i of current
```

```
      txt1(i)="$Heidler: \ I_{0}="+string(I0(i)*10^(-3))+"kA,\ T_{1}="+string(T1(i)*10^6)+"\mu s, \ T_{2}="+string(T2(i)*10^6)+"\mu s, \ n="+string(n(i))+"$" // for figure
    end
    if typ(i)=="de"
      I(i,:)=dexp(I0(i),T1(i),T2(i),tv) // part i of current
      txt1(i)="$double \ exp: \ I_{0}="+string(I0(i)*10^(-3))+"kA,\ T_{1}="+string(T1(i)*10^6)+"\mu s, \ T_{2}="+string(T2(i)*10^6)+"\mu s$" // for figure
    end
    I0t=I0t+I(i,:) // calculate current at origin
  end

  if a==2
    I0tb=I(1,:) // breakdown current for DU model
    I0tc=I(2,:) // corona current for DU model
  else
    I0tb=0
    I0tc=0
  end

  write(%io(2),"figure of current")
  write(%io(2),"-----------------")
  write(%io(2),"   ")
  fig=input("   Draw figures of current? 1 for yes, other for no: ")
  write(%io(2),"   ")

  if fig==1
    tzoom=input("   zoom at tmax in us (0 for none): ")
    write(%io(2),"   ")
    fn=input("      start figure number: ")
    write(%io(2),"   ")

    f1=scf(fn) // seclect figure
    clf(f1) // clear if exist
    tmax=find(tv>=tend,1)
    if a>1
      for i=1:a
        plot2d((tv(1:tmax)*10^6),(I(i,1:tmax)*10^(-3)),style=(i+1)) // plot part i of current
      end
    end
    plot2d((tv(1:tmax)*10^6),(I0t(1:tmax)*10^(-3)),style=1) // plot complete current
    f1.figure_size=[1500 1000]
    f1.children.grid=[1 1]
    select a
      case 1, h1=legend(txt1(1),4),
      case 2, h1=legend(txt1(1),txt1(2),"$current \ at \ origin$",4),
      case 3, h1=legend(txt1(1),txt1(2),txt1(3),"$current \ at \ origin$",4),
      case 4, h1=legend(txt1(1),txt1(2),txt1(3),txt1(4),"$current \ at \ origin$",4),
    end
    xtitle("","$t \; / \; \mu s$","$I \; / \; kA$")

    if tzoom>0
      f2=scf((f1.figure_id+1)) // select figure
      clf(f2) // clear if exist
      tzoom=tzoom*10^(-6)
      tzoom=find(tv>=tzoom,1) // find next index according to zoom given
      if a>1
        for i=1:a
          plot2d((tv(1:tzoom)*10^6),(I(i,1:tzoom)*10^(-3)),style=(i+1)) // plot part i of zoomed current
        end
      end
      plot2d((tv(1:tzoom)*10^6),(I0t(1:tzoom)*10^(-3)),style=1) // plot zoomed current
      f2.figure_size=[1500 1000]
      f2.children.grid=[1 1]
      select a
        case 1, h1=legend(txt1(1),4),
        case 2, h1=legend(txt1(1),txt1(2),"$current \ at \ origin$",4),
        case 3, h1=legend(txt1(1),txt1(2),txt1(3),"$current \ at \ origin$",4),
        case 4, h1=legend(txt1(1),txt1(2),txt1(3),txt1(4),"$current \ at \ origin$",4),
      end
      xtitle("","$t \; / \; \mu s$","$I \; / \; kA$")
    end

    fig=input("   Draw figures of current derivative? 1 for yes, other for no: ")
    write(%io(2),"   ")

    if fig==1
```

```
      tzoom=input("    zoom at tmax in us (0 for none): ")
      write(%io(2),"    ")
      dI0t=splin(tv,I0t)

      f3=scf((f1.figure_id+2)) // select figure
      clf(f3) // clear if exist
      plot2d((tv(1:tmax)*10^6),(dI0t(1:tmax)*10^(-9)),style=1) // plot complete
          current derivative
      f3.figure_size=[1500 1000]
      f3.children.grid=[1 1]
      xtitle("","$t \ \; / \; \mu s$","$\frac{\mathrm{d}I}{\mathrm{d}t} \; / \; \
          frac{kA}{\mu s}$")

      if tzoom>0
        f4=scf((f1.figure_id+3)) // select figure
        clf(f4) // clear if exist
        tzoom=tzoom*10^(-6)
        tzoom=find(tv>=tzoom,1) // find next index according to zoom given
        plot2d((tv(1:tzoom)*10^6),(dI0t(1:tzoom)*10^(-9)),style=1) // plot zoomed
            current derivative
        f4.figure_size=[1500 1000]
        f4.children.grid=[1 1]
        xtitle("","$t \ \; / \; \mu s$","$\frac{\mathrm{d}I}{\mathrm{d}t} \; / \;
            \frac{kA}{\mu s}$")
      end
    end
  end

endfunction
```

ground/func_rsc.sci

```
// v7.1

// functions to calculate return-stroke-currents at channel origin (heidler, dexp
   )

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics



// functionname heidler

// function to calculate current at channel origin
//           as a heidler function

// input: I0, T1, T2, n (scalar)
//         tv ... timevector (vector 1:m)

// output: I0t=I(0,t) ... current at channel origin (vector 1:m)

function I0t=heidler(I0,T1,T2,n,tv)

   eta=exp(-T1/T2*(n*T2/T1)^(1/n))
   frac1=(tv/T1).^n // to reduce processing time
   I0t=I0/eta*(frac1.*(ones(frac1)+frac1).^(-1)).*exp(-tv/T2)

endfunction



// functionname dexp

// function to calculate current at channel origin
//           as a double-exponential function

// input: I0, T1, T2 (scalar)
//         tv ... timevector (vector 1:m)

// output: I0t=I(0,t) ... current at channel origin (vector 1:m)

function I0t=dexp(I0,T1,T2,tv)

   I0t=I0*(exp(-tv/T1)-exp(-tv/T2))

endfunction
```

A.1.3 Calculation of fields

ground/field.sci

```
// v7.1

// module to calculate the field and its components

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname field

// input: I0t=I(0,t) ... current at channel origin
//        I0tb=Ib(0,t) ... breakdown current for DU model
//        I0tc=Ic(0,t) ... corona current for DU model
//        tv ... timevector
//        tend ... endtime for calculation
//        cv ... channelvector
//        vf ... upward-propagating return-front-wave speed in m/s
//        rsmt ... return-stroke-model type (1: TL, 2: MTLE, 3: TCS, 4: DU)
//        la ... coefficient for MTLE model in m
//        taub ... breakdown coefficient for DU model in s
//        taub ... corona coefficient for DU model in s
//        Rp ... distance of fieldpoint in m

// output: E=E(t) ... electrical field in V/m
//         Ee, Ei, Er, Ed ... componenets of electrical field in V/m
//         B=B(t) ... magnetic field in T
//         Bi, Br, Bd ... componenets of magnetic field in T
//         tf ... time of arrival of field at field point in s


function [E,Ee,Ei,Er,Ed,B,Bi,Br,Bd,tf]=field(I0t,I0tb,I0tc,tv,tend,cv,vf,rsmt,la,
    taub,tauc,Rp)

   exec ("e_field.sci",-1) // load field functions
   exec ("i_field.sci",-1) // load field functions
   exec ("r_field.sci",-1) // load field functions
   exec ("d_field.sci",-1) // load field functions

   c=299792458 // speed of light

   tf=Rp/c // starttime of field at fieldpoint

   write(%io(2)," ")

//   timer()

   write(%io(2),"   ... calculating radiation component")
   write(%io(2)," ")
   [Er,Br]=r_field(I0t,I0tb,I0tc,tv,cv,vf,rsmt,la,taub,tauc,Rp) // calculate
       radiation component

//   write(%io(2),timer())
//   write(%io(2)," ")
//   timer()

   write(%io(2),"   ... calculating induction component")
   write(%io(2)," ")
   [Ei,Bi]=i_field(I0t,I0tb,I0tc,tv,cv,vf,rsmt,la,taub,tauc,Rp) // calculate
       induction component

//   write(%io(2),timer())
//   write(%io(2)," ")
//   timer()

   write(%io(2),"   ... calculating electrostatic component")
   write(%io(2)," ")
   Ee=e_field(I0t,I0tb,I0tc,tv,cv,vf,rsmt,la,taub,tauc,Rp) // calculate
       electrostatic component)

//   write(%io(2),timer())
//   write(%io(2)," ")
//   timer()

   if rsmt==3 then // for TCS model
      write(%io(2),"   ... calculating component of discontinuity")
      write(%io(2)," ")
      [Ed,Bd]=d_field(I0t,tv,cv,vf,Rp) // calculate discontinuity component
   else
      Ed=0
      Bd=0
   end
```

```
//    write(%io(2),timer())
    write(%io(2)," ")

    E=Ee+Ei+Er+Ed
    B=Bi+Br+Bd

endfunction
```

ground/e_field.sci

```
// v7.1

// function to calculate the electrostatic component of the field
//          with perfectly conducting ground in height z=0

// by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname e_field

// input: I0t=I(0,t) ... current at channel origin (vector 1:k)
//        I0tb=Ib(0,t) ... breakdown current at channel origin for DU model (
     vector 1:k)
//        I0tc=Ic(0,t) ... corona current at channel origin for DU model (vector
     1:k)
//        tv ... timevector for current (vector 1:k)
//        cv ... channelvector (vector 1:n)
//        vf ... upward-propagating return-front-wave speed (scalar)
//        rsmt ... return-stroke-model type (scalar)
//        la ... coefficient for MTLE model (scalar)
//        taub ... breakdown coefficient for DU model (scalar)
//        taub ... corona coefficient for DU model (scalar)
//        Rp ... distance of fieldpoint (scalar)

// output: Ee(t) ... electric induction component

function Ee=e_field(I0t,I0tb,I0tc,tv,cv,vf,rsmt,la,taub,tauc,Rp)

    c=299792458 // speed of light
    eps0=8.8542*10^(-12) // permittivity
    mu0=4*%pi*10^(-7) // permeability
    ca=length(cv) // number of channelsteps (and timesteps of calculation)
    R=sqrt(Rp^2+cv.^2) // distance from dipole to point P
    ta=length(tv) // number of timesteps of current

    efac=(2*cv.^2-Rp^2)./(R.^5*(2*%pi*eps0)) // factor electrostatic component

    if (rsmt<>4)|(I0tb==zeros(I0tb)) // not Du model or 1 current for DU model
       dI0te=(I0t(ta)-I0t(ta-1))/tv(2) // derivative of current I0t an the end
       dI0t=splin(tv,I0t,"clamped",[0 dI0te]) // derivatives of current I0t with 0
            at starttime
    else // 2 currents form DU model
       dI0te=(I0t(ta)-I0t(ta-1))/tv(2) // derivative of current I0t an the end
       dI0t=splin(tv,I0t,"clamped",[0 dI0te]) // derivatives of current I0t with 0
            at starttime
       dI0tbe=(I0tb(ta)-I0tb(ta-1))/tv(2) // derivative of current I0tb an the end
       dI0tb=splin(tv,I0tb,"clamped",[0 dI0tbe]) // derivatives of breakdown
            current I0tb with 0 at starttime
       dI0tce=(I0tc(ta)-I0tc(ta-1))/tv(2) // derivative of current I0tc an the end
       dI0tc=splin(tv,I0tc,"clamped",[0 dI0tce]) // derivatives of corona current
            I0tc with 0 at starttime
    end

    dFe=zeros(ca,ca) // build inital field matrix
    Ee=zeros(1,ca) // build inital vector of Ee electrostatic component of E

    for j=1:ca

    select rsmt

       case 1 then // TL model
          Izt=[zeros(1,(j-1)) I0t(1:(ca-j+1))]
          index=find(Izt<>0,1)
          if index==[], break, end
          for i=index:ca
             dFe(j,i)=dFe(j,(i-1))+tv(2)*(Izt(i-1)+Izt(i))/2 //calculate Q
          end

       case 2 then // MTLE model
          Izt=[zeros(1,(j-1)) I0t(1:(ca-j+1))*exp(-cv(j)/la)]
```

```
        index=find(Izt<>0,1)
        if index==[], break, end
        for i=index:ca
          dFe(j,i)=dFe(j,(i-1))+tv(2)*(Izt(i-1)+Izt(i))/2 //calculate Q
        end

      case 3 then // TCS model
        tsstart=cv(j)/vf+cv(j)/c // starttime of timestep-vector
        tsend=tsstart+(ca-j)*tv(2) // endtime of timestep-vector
        ts=tsstart:tv(2):tsend// timestep-vector for interpolation
        Izt=interp(ts,tv,I0t,dI0t,"by_zero") // calculate current as spline
            interpolation
        Izt=[zeros(1,(j-1)) Izt(1:(ca-j+1))]
        index=find(Izt<>0,1)
        if index==[], break, end
        for i=index:ca
          dFe(j,i)=dFe(j,(i-1))+tv(2)*(Izt(i-1)+Izt(i))/2 //calculate Q
        end

      case 4 then // DU model
        tsstart=cv(j)/vf+cv(j)/c // starttime of timestep-vector
        tsend=tsstart+(ca-j)*tv(2) // endtime of timestep-vector
        ts=tsstart:tv(2):tsend // timestep-vector for interpolation
        if I0tb==zeros(I0tb) then // one current
          Izt=interp(ts,tv,I0t,dI0t,"by_zero") // calculate current as spline
              interpolation
          dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
          Izt=Izt-Izt(1).*dfb
        else // breakdown and corona current
          Iztb=interp(ts,tv,I0tb,dI0tb,"by_zero") // calculate current as
              spline interpolation
          dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
          Iztc=interp(ts,tv,I0tc,dI0tc,"by_zero") // calculate current as
              spline interpolation
          dfc=exp(-tv(1:length(ts))/tauc) // decrease factor corona
          Izt=Iztb-Iztb(1).*dfb+Iztc-Iztc(1).*dfc
        end
        Izt=[zeros(1,(j-1)) Izt(1:(ca-j+1))]
        index=find(Izt<>0,1)
        if index==[], break, end
        for i=index:ca
          dFe(j,i)=dFe(j,(i-1))+tv(2)*(Izt(i-1)+Izt(i))/2 //calculate Q
        end
      end
    end

    for i=1:ca
      Ee(i)=inttrap(cv(1:i),(dFe(1:i,i)'.*efac(1:i))) // calculate field
          integrate over the channel
    end

endfunction
```

ground/i_field.sci

```
// v7.1

// function to calculate the induction components of the field
//             with perfectly conducting ground in height z=0

// by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname i_field

// input: I0t=I(0,t) ... current at channel origin (vector 1:k)
//        I0tb=Ib(0,t) ... breakdown current at channel origin for DU model (
    vector 1:k)
//        I0tc=Ic(0,t) ... corona current at channel origin for DU model (vector
    1:k)
//        tv ... timevector for current (vector 1:k)
//        cv ... channelvector (vector 1:n)
//        vf ... upward-propagating return-front-wave speed (scalar)
//        rsmt ... return-stroke-model type (scalar)
//        la ... coefficient for MTLE model (scalar)
//        taub ... breakdown coefficient for DU model (scalar)
//        taub ... corona coefficient for DU model (scalar)
//        Rp ... distance of fieldpoint (scalar)

// output: Ei(t) ... electric induction component (vector 1:n)
//          Bi(t) ... magnetic induction component (vector 1:n)
```

```

function [Ei,Bi]=i_field(I0t,I0tb,I0tc,tv,cv,vf,rsmt,la,taub,tauc,Rp)

   c=299792458 // speed of light
   eps0=8.8542*10^(-12) // permittivity
   mu0=4*%pi*10^(-7) // permeability
   ca=length(cv) // number of time- and channelsteps
   R=sqrt(Rp^2+cv.^2) // distance from dipole to point P
   ta=length(tv) // number of timesteps of current

   efac=(2*cv.^2-Rp^2)./(c*R.^4*(2*%pi*eps0)) // factor electric induction component
   bfac=(mu0*Rp)./(2*%pi*R.^3) // factor magnetic induction component

   if (rsmt<>4)|(I0tb==zeros(I0tb)) // not Du model or 1 current for DU model
      dI0te=(I0t(ta)-I0t(ta-1))/tv(2) // derivative of current I0t an the end
      dI0t=splin(tv,I0t,"clamped",[0 dI0te]) // derivatives of current I0t with 0 at starttime
   else // 2 currents form DU model
      dI0te=(I0t(ta)-I0t(ta-1))/tv(2) // derivative of current I0t an the end
      dI0t=splin(tv,I0t,"clamped",[0 dI0te]) // derivatives of current I0t with 0 at starttime
      dI0tbe=(I0tb(ta)-I0tb(ta-1))/tv(2) // derivative of current I0tb an the end
      dI0tb=splin(tv,I0tb,"clamped",[0 dI0tbe]) // derivatives of breakdown current I0tb with 0 at starttime
      dI0tce=(I0tc(ta)-I0tc(ta-1))/tv(2) // derivative of current I0tc an the end
      dI0tc=splin(tv,I0tc,"clamped",[0 dI0tce]) // derivatives of corona current I0tc with 0 at starttime
   end

   dFi=zeros(ca,ca) // build inital field matrix
   Ei=zeros(1,ca) // build inital line vector of Ei electric induction component of E
   Bi=zeros(1,ca) // build inital line vector of Bi magnetic induction component of E

   for j=1:ca

      select rsmt

      case 1 then // TL model
         Izt=[zeros(1,(j-1)) I0t(1:(ca-j+1))]
         dFi(j,:)=Izt

      case 2 then // MTLE model
         Izt=I0t*exp(-cv(j)/la)
         Izt=[zeros(1,(j-1)) I0t(1:(ca-j+1))*exp(-cv(j)/la)]
         dFi(j,:)=Izt

      case 3 then // TCS model
         tsstart=cv(j)/vf+cv(j)/c // starttime of timestep-vector
         tsend=tsstart+(ca-j+1)*tv(2) // endtime of timestep-vector
         ts=tsstart:tv(2):tsend// timestep-vector for interpolation
         Izt=interp(ts,tv,I0t,dI0t,"by_zero")   // calculate current as spline interpolation by_zero: if out-of-bound finish extrapolation with 0
         Izt=[zeros(1,j-1) Izt(1:(ca-j+1))]
         dFi(j,:)=Izt

      case 4 then // DU model
         tsstart=cv(j)/vf+cv(j)/c // starttime of timestep-vector
         tsend=tsstart+(ca-j+1)*tv(2) // endtime of timestep-vector
         ts=tsstart:tv(2):max(tv) // timestep-vector for interpolation
         if I0tb==zeros(I0tb) then
            Izt=interp(ts,tv,I0t,dI0t,"by_zero") // calculate current as spline interpolation by_zero: if out-of-bound finish extrapolation with 0
            dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
            Izt=Izt-Izt(1).*dfb
         else
            Iztb=interp(ts,tv,I0tb,dI0tb,"by_zero") // calculate current as spline interpolation by_zero: if out-of-bound finish extrapolation with 0
            dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
            Iztc=interp(ts,tv,I0tc,dI0tc,"by_zero") // calculate current as spline interpolation by_zero: if out-of-bound finish extrapolation with 0
            dfc=exp(-tv(1:length(ts))/tauc) // decrease factor corona
            Izt=Iztb-Iztb(1).*dfb+Iztc-Iztc(1).*dfc
         end
         Izt=[zeros(1,j-1) Izt(1:(ca-j+1))]
         dFi(j,:)=Izt
      end
   end

```

```
   for i=2:ca
      Ei(i)=inttrap(cv(1:i),(dFi(1:i,i)'.*efac(1:i))) // calculate field integrate over the channel
      Bi(i)=inttrap(cv(1:i),(dFi(1:i,i)'.*bfac(1:i)))
   end

endfunction
```

ground/r_field.sci

```
// v7.1

// function to calculate the radiation components of the field
//           with perfectly conducting ground in height z=0

// by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname r_field

// input: I0t=I(0,t) ... current at channel origin (vector 1:k)
//        I0tb=Ib(0,t) ... breakdown current at channel origin for DU model (vector 1:k)
//        I0tc=Ic(0,t) ... corona current at channel origin for DU model (vector 1:k)
//        tv ... timevector for current (vector 1:k)
//        cv ... channelvector (vector 1:n)
//        vf ... upward-propagating return-front-wave speed (scalar)
//        rsmt ... return-stroke-model type (scalar)
//        la ... coefficient for MTLE model (scalar)
//        taub ... breakdown coefficient for DU model (scalar)
//        taub ... corona coefficient for DU model (scalar)
//        Rp ... distance of fieldpoint (scalar)

// output: Er(t) ... electric radiation component (vector 1:n)
//         Br(t) ... magnetical radiation component (vector 1:n)

function [Er,Br]=r_field(I0t,I0tb,I0tc,tv,cv,vf,rsmt,la,taub,tauc,Rp)

   c=299792458 // speed of light
   eps0=8.8542*10^(-12) // permittivity
   mu0=4*%pi*10^(-7) // permeability
   ca=length(cv) // amount of time- and channelsteps
   R=sqrt(Rp^2+cv.^2) // distance from dipole to point P
   ta=length(tv) // amount of timesteps of current

   efac=(-1)*(Rp^2)./(c^2*R.^3*(2*%pi*eps0)) // factor electric radiation component
   bfac=(mu0*Rp)./(2*%pi*c*R.^2) // factor magnetic radiaion component

   if (rsmt<>4)|(I0tb==zeros(I0tb)) // not Du model or 1 current for DU model
      dI0te=(I0t(ta)-I0t(ta-1))/tv(2) // derivative of current I0t an the end
      dI0t=splin(tv,I0t,"clamped",[0 dI0te]) // derivatives of current I0t with 0 at starttime
   else // 2 currents form DU model
      dI0te=(I0t(ta)-I0t(ta-1))/tv(2) // derivative of current I0t an the end
      dI0t=splin(tv,I0t,"clamped",[0 dI0te]) // derivatives of current I0t with 0 at starttime
      dI0tbe=(I0tb(ta)-I0tb(ta-1))/tv(2) // derivative of current I0tb an the end
      dI0tb=splin(tv,I0tb,"clamped",[0 dI0tbe]) // derivatives of breakdown current I0tb with 0 at starttime
      dI0tce=(I0tc(ta)-I0tc(ta-1))/tv(2) // derivative of current I0tc an the end
      dI0tc=splin(tv,I0tc,"clamped",[0 dI0tce]) // derivatives of corona current I0tc with 0 at starttime
   end

   dFr=zeros(ca,ca) // build inital field matrix
   Er=zeros(1,ca) // build inital line vector of Er radiation component of E
   Br=zeros(1,ca) // build inital line vector of Br radiation component of B

   for j=1:ca

   select rsmt

      case 1 then // TL model
         dFr(j,:)=[zeros(1,j-1) dI0t(1:(ca-j+1))]

      case 2 then // MTLE model
         Izt=I0t*exp(-cv(j)/la)
         dIzte=(Izt(ta)-Izt(ta-1))/tv(2)
         dIzt=splin(tv,Izt,"clamped",[0 dIzte])
```

```
         dFr(j,:)=[zeros(1,j-1) dIzt(1:(ca-j+1))]

      case 3 then // TCS model
         tsstart=cv(j)/vf+cv(j)/c // starttime of timestep-vector
         tsend=tsstart+(ca-j+1)*tv(2) // endtime of timestep-vector
         ts=tsstart:tv(2):tsend // timestep-vector for interpolation
         Izt=interp(ts,tv,I0t,dI0t,"by_zero") // calculate current as spline
              interpolation by_zero: if out-of-bound finish extrapolation with 0
         dIzt=splin(ts,Izt,"monotone") // derivative of current in channel
         dFr(j,:)=[zeros(1,j-1) dIzt(1:(ca-j+1))]

      case 4 then // DU model
         tsstart=cv(j)/vf+cv(j)/c // starttime of timestep-vector
         tsend=tsstart+(ca-j+1)*tv(2) // endtime of timestep-vector
         ts=tsstart:tv(2):tsend //max(tv)// timestep-vector for interpolation
         if I0tb==zeros(I0tb) then
            Izt=interp(ts,tv,I0t,dI0t,"by_zero") // calculate current as spline
                 interpolation by_zero: if out-of-bound finish extrapolation with
                 0
            dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
            Izt=Izt-Izt(1).*dfb
         else
            Iztb=interp(ts,tv,I0tb,dI0tb,"by_zero") // calculate current as
                 spline interpolation by_zero: if out-of-bound finish
                 extrapolation with 0
            dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
            Iztc=interp(ts,tv,I0tc,dI0tc,"by_zero") // calculate current as
                 spline interpolation by_zero: if out-of-bound finish
                 extrapolation with 0
            dfc=exp(-tv(1:length(ts))/tauc) // decrease factor corona
            Izt=Iztb-Iztb(1).*dfb+Iztc-Iztc(1).*dfc
         end
         tsl=length(ts)
         dIzte=(Izt(tsl)-Izt(tsl-1))/tv(2)
         dIzt=splin(ts,Izt,"clamped",[0 dIzte])
         dFr(j,:)=[zeros(1,j-1) dIzt(1:(ca-j+1))]
      end
   end


   for i=2:ca
      Er(i)=inttrap(cv(1:i),(dFr(1:i,i)'.*efac(1:i))) // calculate field
           integrate over the channel
      Br(i)=inttrap(cv(1:i),(dFr(1:i,i)'.*bfac(1:i)))
   end;

endfunction
```

ground/d_field.sci

```
// v7.1

// function to calculate the discontinuity components of the field
//          with perfectly conducting ground in height z=0

// by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname d_field

// input: I0t=I(0,t) ... current at channel origin (vector 1:k)
//        tv ... timevector for current (vector 1:k)
//        cv ... channelvector (vector 1:n)
//        vf ... upward-propagating return-front-wave speed (scalar)
//        Rp ... distance of fieldpoint (scalar)

// output: Ed(t) ... electric discontinuity component (vector 1:n)
//         Bd(t) ... magnetic discontinuity component (vector 1:n)

function [Ed,Bd]=d_field(I0t,tv,cv,vf,Rp)

   c=299792458 // speed of light
   eps0=8.8542*10^(-12) // permittivity
   mu0=4*%pi*10^(-7) // permeability
   ca=length(cv) // number of channel- and timesteps
   R=sqrt(Rp^2+cv.^2) // distance from dipole to point P

   dI0te=(I0t(ta)-I0t(ta-1))/tv(2) // derivative of current I0t an the end
   dI0t=splin(tv,I0t,"clamped",[0 dI0te]) // derivatives of current I0t with 0 at
        starttime
```

```
  dH=splin(tv(1:ca),cv) // derivative of high of discontinuity with respect to time

  efac=-(Rp^2)./(c^2*R.^3*(2*%pi*eps0)) // factor electric discontinuity component
  bfac=(mu0*Rp)./(2*%pi*c*R.^2) // factor magnetic discontinuity component

  Ed=zeros(1,ca) // build inital line vector of Ed discontinuity component of E
  Bd=zeros(1,ca) // build inital line vector of Bd discontinuity component of B

  tsstart=cv/vf+cv/c // starttime of current
  Iht=interp(tsstart,tv,I0t,dI0t,"by_zero") // calculate current as spline interpolation by_zero: if out-of-bound finish extrapolation with 0
  Ed=efac.*Iht.*dH
  Bd=bfac.*Iht.*dH

endfunction
```

A.1.4 Plotting and saving of data

ground/func_save.sci

```
// v7.1

// functions to save the field (save_field, save_all_field)

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics



// functionname save_field

// functions to save field and its components

// input: E=E(t) ... electrical field in V/m
//        Ee, Ei, Er, Ed ... components of electrical field in V/m
//        B=B(t) ... magnetic field in T
//        Bi, Br, Bd ... components of magnetic field in T
//        tf ... time of arrival of field at field point in s
//        tv ... timevector
//        vf ... upward-propagating return-front-wave speed in m/s
//        rsmt ... return-stroke-model type (1: TL, 2: MTLE, 3: TCS, 4: DU)
//        la ... coefficient for MTLE model in m
//        taub ... breakdown coefficient for DU model in s
//        taub ... corona coefficient for DU model in s
//        Rp ... distance of fieldpoint in m

// output: none

function []=save_field(E,Ee,Ei,Er,Ed,B,Bi,Br,Bd,tf,tv,vf,rsmt,la,taub,tauc,Rp)

  select rsmt
    case 1, txt="TL Model: vf="+string(vf)+"m/s"
    case 2, txt="MTLE Model: vf="+string(vf)+"m/s, lambda="+string(la)+"m"
    case 3, txt="TCS Model: vf="+string(vf)+"m/s"
    case 4, txt="DU Model: vf="+string(vf)+"m/s, Tb="+string(taub)+"s, Tc="+string(tauc)+"s"
    else txt=""
  end

  if length(Ed)<length(E) then
    Ed=zeros(E)
    Bd=zeros(E)
  end

  write(%io(2),"save field")
  write(%io(2),"----------")
  write(%io(2),"  ")
  sav=input("  save field and components to file? 1 for yes, other for no: ")
  write(%io(2),"  ")
  if sav==1 then
    errcatch(999,"continue")
    fnw=input("    filename: ","s")
    fid=mopen(fnw,"w")
    if fid==-1 then
      write(%io(2),"    Error: Cannot open file for writing!")
    else
      write(%io(2),"    ... writing data to file: "+fnw)
      write(%io(2),"    ")
```

```
      mfprintf(fid,"%s \n",txt)
      mfprintf(fid,"%s %3.2f %s \n","field and components at fieldpoint: ",(Rp*10^(-3)),"km")
      mfprintf(fid,"%s \t \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \n","Name:","E","Ee","Ei","Er","Ed","B","Bi","Br","Bd")
      mfprintf(fid,"%s \t %i \n","Length:",length(E))
      mfprintf(fid,"%s \t %e %s \n","Starttime:",tf,"s")
      mfprintf(fid,"%s \t %e s \n","Unit X-Axis:",tv(2))
      mfprintf(fid,"%s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \n","Unit Y-Axis:","V/m","V/m","V/m","V/m","V/m","T","T","T","T")
      mfprintf(fid,"%s \n","Data:")
      for i=1:length(E)
        mfprintf(fid,"%1.5e \t %1.5e \t %1.5e \t %1.5e \t %1.5e \t %1.5e \t %1.5e \t %1.5e \t %1.5e \t \n",E(i),Ee(i),Ei(i),Er(i),Ed(i),B(i),Bi(i),Br(i),Bd(i))
      end
      mclose(fid)
    end
    errcatch(-1)
  end

endfunction



// functionname save_all_field

// function to save fields of all models

// input: E_tl, E_mtle, E_tcs, E_du ... electrical field of all models in V/m
//        B_tl, B_mtle, B_tcs, B_du ... magnetic field of all models in T
//        tf ... time of arrival of field at field point in s
//        tv ... timevector
//        vf ... upward-propagating return-front-wave speed in m/s
//        la ... coefficient for MTLE model in m
//        taub ... breakdown coefficient for DU model in s
//        taub ... corona coefficient for DU model in s
//        Rp ... distance of fieldpoint in m

// output: none

function []=save_all_field(E_tl,E_mtle,E_tcs,E_du,B_tl,B_mtle,B_tcs,B_du,tf,tv,vf,la,taub,tauc,Rp)

  txt1="TL Model: vf="+string(vf)+"m/s"
  txt2="MTLE Model: vf="+string(vf)+"m/s, lambda="+string(la)+"m"
  txt3="TCS Model: vf="+string(vf)+"m/s"
  txt4="DU Model: vf="+string(vf)+"m/s, Tb="+string(taub)+"s, Tc="+string(tauc)+"s"

   write(%io(2),"save fields")
   write(%io(2),"-----------")
   write(%io(2),"  ")
   sav=input("  save fields to file? 1 for yes, other for no: ")
   write(%io(2),"  ")
   if sav==1 then
      fnw=input("   filename: ","s")
      fid=mopen(fnw,"w")
      if fid==-1 then
         write(%io(2),"   Error: Cannot open file for writing!")
      else
         write(%io(2),"   ... writing data to file: "+fnw)
         write(%io(2),"   ")
         mfprintf(fid,"%s \n",txt1)
         mfprintf(fid,"%s \n",txt2)
         mfprintf(fid,"%s \n",txt3)
         mfprintf(fid,"%s \n",txt4)
         mfprintf(fid,"%s %3.2f %s \n","fields at fieldpoint: ",(Rp*10^(-3)),"km")
         mfprintf(fid,"%s \t \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \n","Name:","E_tl","E_mtle","E_tcs","E_du","B_tl","B_mtle","B_tcs","B_du")
         mfprintf(fid,"%s \t %i \n","Length:",length(E_tl))
         mfprintf(fid,"%s \t %e %s \n","Starttime:",tf,"s")
         mfprintf(fid,"%s \t %e s \n","Unit X-Axis:",tv(2))
         mfprintf(fid,"%s \t %e s \n","1st x-Value:",tf)
         mfprintf(fid,"%s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \n","Unit Y-Axis:","V/m","V/m","V/m","V/m","T","T","T","T")
         mfprintf(fid,"%s \n","Data:")
         for i=1:length(E_tl)
            mfprintf(fid,"%1.5e \t %1.5e \t %1.5e \t %1.5e %1.5e \t %1.5e \t %1.5e \t %1.5e \n",E_tl(i),E_mtle(i),E_tcs(i),E_du(i),B_tl(i),B_mtle(i),B_tcs(i),B_du(i))
         end
```

```
        mclose(fid)
      end
    end

endfunction
```

ground/func_plot.sci

```
// v7.1

// functions to plot the field (plot_field, plot_all_field)

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics



// functionname plot_field

// function to plot field and its components

// input: E=E(t) ... electrical field in V/m
//        Ee, Ei, Er, Ed ... components of electrical field in V/m
//        B=B(t) ... magnetic field in T
//        Bi, Br, Bd ... components of magnetic field in T
//        tend ... endtime of calculation in s
//        tv ... time-vector
//        tzoom ... zoomtime in us
//        fn ... startnumber for figure

// output: none

function []=plot_field(rsmt,E,Ee,Ei,Er,Ed,B,Bi,Br,Bd,tend,tv,tzoom,fn)

      tzoom=tzoom*10^(-6)

      wide=1000
      high=500
      tit=""

      select rsmt
      case 1 then
          tit="Transmission Line Model"
      case 2 then
          tit="Modified Transmission Line Model"
      case 3 then
          tit="Travelling Current Source Model"
      case 4 then
          tit="Diendorfer Uman Model"
      end

      f1=scf(fn) // select figure
      clf(f1) // clear if exist
      tmax=find(tv>=tend,1) // find next index according to endtime given
      if Ed<>0
         plot2d((tv(1:tmax)*10^6)',[Ee(1:tmax)' Ei(1:tmax)' Er(1:tmax)' Ed(1:tmax)' E(1:tmax)'],style=[2 13 5 3 1])
      else
         plot2d((tv(1:tmax)*10^6)',[Ee(1:tmax)' Ei(1:tmax)' Er(1:tmax)' E(1:tmax)'],style=[2 13 5 1])
      end
      f1.figure_size=[wide high]
      f1.children.grid=[1 1]
      f1.children.margins(2)=0.3
      if Ed<>0
        h1=legend("$electrostatic \ component$","$induction \ component$","$radiation \ component$","$discontinuity \ component$","$electrical \ field$",-1)
      else
        h1=legend("$electrostatic \ component$","$induction \ component$","$radiation \ component$","$electrical \ field$",-1)
      end
      xtitle(tit,"$t \ / \ \mu s$","$E \ / \ \frac{V}{m}$")

      if tzoom>0
         f2=scf(f1.figure_id+1) // select figure
         clf(f2) // clear if exist
         tzoom=find(tv>=tzoom,1) // find next index according to zoom given
         if Ed<>0
            plot2d((tv(1:tzoom)*10^6)',[Ee(1:tzoom)' Ei(1:tzoom)' Er(1:tzoom)' Ed(1:tzoom)' E(1:tzoom)'],style=[2 13 5 3 1])
```

```
        else
           plot2d((tv(1:tzoom)*10^6)',[Ee(1:tzoom)' Ei(1:tzoom)' Er(1:tzoom)' E(1:tzoom)'],style=[2 13 5 1])
        end
        f2.figure_size=[wide high]
        f2.children.grid=[1 1]
        f2.children.margins(2)=0.3
        if Ed<>0
           h1=legend("$electrostatic \ component$","$induction \ component$","$radiation \ component$","$discontinuity \ component$","$electrical \ field$",-1)
        else
           h1=legend("$electrostatic \ component$","$induction \ component$","$radiation \ component$","$electrical \ field$",-1)
        end
        xtitle(tit,"$t \ / \ \mu s$","$E \ / \ \frac{V}{m}$")
     end

     f3=scf(f1.figure_id+2) // select figure
     clf(f3) // clear if exist
     if Bd<>0
        plot2d((tv(1:tmax)*10^6)',[(Bi(1:tmax)*10^6)' (Br(1:tmax)*10^6)' (Bd(1:tmax)*10^6)' (B(1:tmax)*10^6)'],style=[2 13 5 1])
     else
        plot2d((tv(1:tmax)*10^6)',[(Bi(1:tmax)*10^6)' (Br(1:tmax)*10^6)' (B(1:tmax)*10^6)'],style=[2 13 1])
     end
     f3.figure_size=[wide high]
     f3.children.grid=[1 1]
     f3.children.margins(2)=0.3
     if Bd<>0
       h1=legend("$induction \ component$","$radiation \ component$","$discontinuity \ component$","$magnetic \ field$",-1)
     else
       h1=legend("$induction \ component$","$radiation \ component$","$magnetic \ field$",-1)
     end
     xtitle(tit,"$t \ / \ \mu s$","$B \ / \ \mu T$")

     if tzoom>0
        f4=scf(f1.figure_id+3) // select figure
        clf(f4) // clear if exist
        if Bd<>0
          plot2d((tv(1:tzoom)*10^6)',[(Bi(1:tzoom)*10^6)' (Br(1:tzoom)*10^6)' (Bd(1:tzoom)*10^6)' (B(1:tzoom)*10^6)'],style=[2 13 5 1])
        else
           plot2d((tv(1:tzoom)*10^6)',[(Bi(1:tzoom)*10^6)' (Br(1:tzoom)*10^6)' (B(1:tzoom)*10^6)'],style=[2 13 1])
        end
        f4.figure_size=[wide high]
        f4.children.grid=[1 1]
        f4.children.margins(2)=0.3
        if Bd<>0
           h1=legend("$induction \ component$","$radiation \ component$","$discontinuity \ component$","$magnetic \ field$",-1)
        else
           h1=legend("$induction \ component$","$radiation \ component$","$magnetic \ field$",-1)
        end
        xtitle(tit,"$t \ / \ \mu s$","$B \ / \ \mu T$")
     end

endfunction


// functionname plot_all_field

// function to plot fields of all models

// input: E_tl, E_mtle, E_tcs, E_du ... electrical field of all models in V/m
//        B_tl, B_mtle, B_tcs, B_du ... magnetic field of all models in T
//        tend ... endtime of calculation in s
//        tv ... time-vector
//        tzoom ... zoomtime in us
//        fn ... startnumber for figure
//        la ... coefficient for MTLE model in m
//        taub ... breakdown coefficient for DU model in s
//        tauc ... corona coefficient for DU model in s

// output: none

function []=plot_all_field(E_tl,E_mtle,E_tcs,E_du,B_tl,B_mtle,B_tcs,B_du,tend,tv,tzoom,fn,la,taub,tauc)

```

```
  tzoom=tzoom*10^(-6)

  wide=1000
  high=500

  txt1="$TL \ model$"
  txt2="$MTLE \ model: \ \lambda="+string(la)+"m $"
  txt3="$TCS \ model$"
  if tauc==0 then
    txt4="$DU \ model: \ \tau ="+string((taub*10^6))+"\mu s$"
  else
    txt4="$DU \ model: \ \tau_{b} ="+string((taub*10^6))+"\mu s, \ \tau_{c} ="+string((tauc*10^6))+"\mu s$"
  end

  f1=scf(fn) // select figure
  clf(f1) // clear if exist
  tmax=find(tv>=tend,1)
  plot2d((tv(1:tmax)*10^6)',[E_tl(1:tmax)' E_mtle(1:tmax)' E_tcs(1:tmax)' E_du(1:tmax)'],style=[2 13 5 1])
  h1=legend(txt1,txt2,txt3,txt4,-1)
  f1.figure_size=[wide high]
  f1.children.grid=[1 1]
  f1.children.margins(2)=0.3
  xtitle("All Models","$t \ / \ \mu s$","$E \ / \ \frac{V}{m}$")

  if tzoom>0
    f2=scf(f1.figure_id+1) // select figure
    clf(f2) // clear if exist
    tzoom=find(tv>=tzoom,1) // find next index according to zoom given
    plot2d((tv(1:tzoom)*10^6)',[E_tl(1:tzoom)' E_mtle(1:tzoom)' E_tcs(1:tzoom)' E_du(1:tzoom)'],style=[2 13 5 1])
    h1=legend(txt1,txt2,txt3,txt4,-1)
    f2.figure_size=[wide high]
    f2.children.grid=[1 1]
    f2.children.margins(2)=0.3
    xtitle("All Models","$t \ / \ \mu s$","$E \ / \ \frac{V}{m}$")
  end

  f3=scf(f1.figure_id+2) // select figure
  clf(f3) // clear if exist
  plot2d((tv(1:tmax)*10^6)',[(B_tl(1:tmax)*10^6)' (B_mtle(1:tmax)*10^6)' (B_tcs(1:tmax)*10^6)' (B_du(1:tmax)*10^6)'],style=[2 13 5 1])
  f3.figure_size=[wide high]
  f3.children.grid=[1 1]
  f3.children.margins(2)=0.3
  h1=legend(txt1,txt2,txt3,txt4,-1)
  xtitle("All Models","$t \ / \ \mu s$","$B \ / \ \mu T$")

  if tzoom>0
    f4=scf(f1.figure_id+3) // select figure
    clf(f4) // clear if exist
    plot2d((tv(1:tzoom)*10^6)',[(B_tl(1:tzoom)*10^6)' (B_mtle(1:tzoom)*10^6)' (B_tcs(1:tzoom)*10^6)' (B_du(1:tzoom)*10^6)'],style=[2 13 5 1])
    f4.figure_size=[wide high]
    f4.children.grid=[1 1]
    f4.children.margins(2)=0.3
    h1=legend(txt1,txt2,txt3,txt4,-1)
    xtitle("All Models","$t \ / \ \mu s$","$B \ / \ \mu T$")
  end

endfunction
```

A.1.5 Definitionfile

ground/def_ground.txt

```
definitionfile for ground

remarks:

********** ********** **********
start_of_def

tend        50
ta          10000
vf          1.3
Rp          100

currents
h    13    0.73    3    2
h    7     5       50   2
```

```
x     0     0     0     0
x     0     0     0     0
x     0     0     0     0

modelparameter
la        2000
taub      0.1
tauc      0.1

end_of_def
********** ********** **********

format of data:

tend      endtime of field calculation in us
ta        number of timesteps
vf        return stroke front wave speed in the channel in 10^8 m/s
Rp        distance of field point in km

currents
a     b     c     d     e     f

a ... function of current: de for double exponentiell, h for heidler-function, x
     for no more currents defined
b ... I0 in [kA]
c ... T1 in [us]
d ... T2 in [us]
e ... n (need only for heidler-function)

modelparameter
la        exponential decay coefficient for MTLE model in m
taub      breakdown discharge time constant for DU model in us
tauc      corona discharge time constant for DU model in us

for copy/paste:

Nucci90:

h     -9.9     0.072     5     2
de    -7.5     100       6     0

Diendorfer90:

h     -13     0.73     3     2
h     -7      5        50    2
```

A.2 Lightning Strike to a Tall Object

A.2.1 Startmodule

tall/start.sce

```
// v3.1

// startmodule for calculation of fields for a strike on a tall grounded object

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

  funcprot(0)

  stacksize('max')

  exec ("def.sci",-1)
  exec ("heightvector.sci",-1)
  exec ("distribution.sci",-1)
  exec ("field.sci",-1)
  exec ("plot_field.sci",-1)
  exec ("save_field.sci",-1)

  write(%io(2),"    ")
  write(%io(2),"    ")
  write(%io(2),"Lightning Strike to a Tall Object")
  write(%io(2),"---------------------------------")
  write(%io(2),"    ")

  d=0 // discontinuity field
```

```

  [tv,H,rhotop,rhobot,nstop,vf,Rp,I0] = def(); // input parameter

  [zvc,zvt] = heightvector(tv,Rp,H,vf); // calculate heightvectors

  [Izt,zv] = distribution(tv,zvt,zvc,I0,rhotop,rhobot,nstop,vf); // calculate current distribution

  [E,Er,Ei,Ee,B,Br,Bi,tf] = field(Izt,zv,tv,Rp); // calculate fields

  if d==1 then
      exec ("d_field.sci",-1)
      [E,Ed,B,Bd] = d_field(Izt,zv,zvc,tv,Rp,E,B) // calculate discontinuity field
  end

  write(%io(2),"figures of fields")
  write(%io(2),"-----------------")
  write(%io(2),"    ")
  fig=input("    Draw figures of fields? 1 for yes, other for no: ")
  write(%io(2),"    ")
  if fig==1
    tzoom=input("    zoom at tmax in us (0 for none): ")
    write(%io(2),"    ")
    fn=input("        start figure number: ")
    write(%io(2),"    ")
    Hi=max(zvt)

    if d==1 then
        exec ("plot_d_field.sci",-1)
        plot_d_field(E,Ee,Ei,Er,Ed,B,Bi,Br,Bd,tv,tzoom,fn,vf,Rp,Hi);
    else
        plot_field(E,Ee,Ei,Er,B,Bi,Br,tv,tzoom,fn,vf,Rp,Hi);
    end
  end

  save_field(E,Ee,Ei,Er,B,Bi,Br,tf,tv,H,rhotop,rhobot,vf,Rp);
```

A.2.2 Definition of Parameters

tall/def.sci

```
// v3.1

// function to define parameter and plot I0

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname def

// input: none

// output: tv ... timevector in s
//         H ... height of tall object in m
//         rhotop ... current reflection coefficient at the top
//         rhobot ...current reflection coefficient at the bottom
//         nstop ... max index of successive multiple reflections
//         vf ... retrun-stroke front wave speed in m/s
//         Rp ... distance of field point in m
//         I0 ... undisturbed current in A

function [tv,H,rhotop,rhobot,nstop,vf,Rp,I0] = def()

  exec ("rsc.sci",-1) // load return stroke current functions

  c=299792458 // speed of light

  write(%io(2),"input parameter")
  write(%io(2),"---------------")
  write(%io(2),"    ")
  rd=input("    read definition from file def_tall.txt? 1 for yes, other for no: ")
  write(%io(2),"    ")

  if rd==1 then
    errcatch(999,"continue")
    fid=mopen("def_tall.txt","r")
    if fid==-1 then
      write(%io(2),"    error: cannot open file def.txt!")
```

```
    else
      header=1
      while header==1
        val=mfscanf(1,fid,"%s")
        if val=="start_of_def" then, header=0, end
      end
      data=1
      while data==1
        val=mfscanf(1,fid,"%s")
        select val
          case "tend", tend=mfscanf(1,fid,"%f")*10^(-6),
          case "ta" then
            ta=mfscanf(1,fid,"%f")
            ts=tend/ta // timestep
            tv=0:ts:tend // build timevector
          case "H", H=mfscanf(1,fid,"%f"),
          case "rhotop", rhotop=mfscanf(1,fid,"%f"),
          case "rhobot", rhobot=mfscanf(1,fid,"%f"),
          case "n", nstop=mfscanf(1,fid,"%f"),
          case "vf", vf=mfscanf(1,fid,"%f")*10^8,
          case "Rp", Rp=mfscanf(1,fid,"%f")*10^3,
          case "currents" then
            I=zeros(tv)
            i=1
            next=1
            while next==1
              [dummy,typ,I0,T1,T2,n]=mfscanf(1,fid,"%s %f %f %f %f %f")
              if typ=="x" then
                next=0
              else
                I0=I0*10^3
                T1=T1*10^-6
                T2=T2*10^-6
                if typ=="h" then
                  I(i,:)=heidler(I0,T1,T2,n,tv) // build currentvector
                  txt1(i)="$Heidler: \ I0[kA]="+string(I0*10^(-3))+",\ T1[\mu s]="+string(T1*10^6)+", \ T2[\mu s]="+string(T2*10^6)+", \ n="+string(n)+"$" // for figure
                end
                if typ=="de" then
                  I(i,:)=dexp(I0,T1,T2,tv) // build currentvector
                  txt1(i)="$double \ exp: \ I0[kA]="+string(I0*10^(-3))+",\ T1[\mu s]="+string(T1*10^6)+", \ T2[\mu s]="+string(T2*10^6)+"$" // for figure
                end
                i=i+1
              end
            end
          case "end_of_def", data=0,
        end
      end
    end
    mclose(fid)
    errcatch(-1)

  else
    write(%io(2),"   ")
    write(%io(2),"   ")
    write(%io(2),"timevector")
    write(%io(2),"----------")
    write(%io(2),"   ")
    tend=input("   Endtime [us]: ")
    write(%io(2),"   ")
    tend=tend*10^(-6)
    ta=input("   number of timesteps []: ")
    write(%io(2),"   ")
    ts=tend/ta // timestep
    tv=0:ts:tend // build timevector
    write(%io(2),"      --> length of timestep [us]:")
    write(%io(2),(ts/10^(-6)),"(7X,F10.5)")
    write(%io(2),"   ")

    write(%io(2),"   ")
    write(%io(2),"   ")
    write(%io(2),"tall objekt")
    write(%io(2),"-----------")
    write(%io(2),"   ")
    H=input("   Height [m]: ")
    write(%io(2),"   ")
    rhotop=input("   current reflection coefficient at the top []: ")
    write(%io(2),"   ")
    rhobot=input("   current reflection coefficient at the bottom []: ")
    write(%io(2),"   ")
    nstop=input("   max. number of reflections to calculate []: ")
```

```
    write(%io(2),"   ")

    write(%io(2),"   ")
    write(%io(2),"   ")
    write(%io(2),"channel")
    write(%io(2),"-------")
    write(%io(2),"   ")
    vf=input("   return stroke front speed [10^8 m/s]: ")
    vf=vf*10^(8)
    write(%io(2),"   ")

    write(%io(2),"   ")
    write(%io(2),"   ")
    write(%io(2),"field-point")
    write(%io(2),"-----------")
    write(%io(2),"   ")
    Rp=input("   distance to field-point [km]: ")
    Rp=Rp*10^(3)
    write(%io(2),"   ")

    write(%io(2),"   ")
    write(%io(2),"   ")
    write(%io(2),"short-circuit current")
    write(%io(2),"---------------------")
    write(%io(2),"   ")
    a=input("   How many functions define the current? ")
    write(%io(2),"  ")

    for i=1:a
      b=input("   1 for Heidler, 2 for Double-Exponetial, 3 Nucci90, 4 DU90 other to break: ")
      write(%io(2),"   ")

      if b>2 then, break, end

      if b==1
        write(%io(2),"   Heidler:")
        I0=input("     I0 [kA]: ")
        I0=I0*10^3
        T1=input("     T1 [us]: ")
        T1=T1*10^(-6)
        T2=input("     T2 [us]: ")
        T2=T2*10^(-6)
        n=input("     n []: ")
        write(%io(2),"   ")
        I(i,:)=heidler(I0,T1,T2,n,tv) // build currentvector
        txt1(i)="$Heidler: \ I_{0}="+string(I0*10^(-3))+"kA,\ T_{1}="+string(T1*10^6)+"\mu s, \ T_{2}="+string(T2*10^6)+"\mu s, \ n="+string(n)+"$" // for figure
      end

      if b==2
        write(%io(2),"   double exponential:")
        I0=input("     I0 [kA]: ")
        I0=I0*10^3
        T1=input("     T1 [us]: ")
        T1=T1*10^(-6)
        T2=input("     T2 [us]: ")
        T2=T2*10^(-6)
        write(%io(2),"   ")
        I(i,:)=dexp(I0,T1,T2,tv) // build currentvector
        txt1(i)="$double \ exp: \ I_{0}="+string(I0*10^(-3))+"kA,\ T_{1}="+string(T1*10^6)+"\mu s, \ T_{2}="+string(T2*10^6)+"\mu s $" // for figure
      end
    end

    if b==3 // define current used by Nucci et al. (1990)
      I(1,:)=heidler(-9900,0.072*10^(-6),5*10^(-6),2,tv)
      txt1(1)="$Heidler: \ I_{0}=-9.9kA, \ T_{1}=0.072\mu s,\ T_{2}=5\mu s,\ n=2$"
      I(2,:)=dexp(-7500,100*10^(-6),6*10^(-6),tv)
      txt1(2)="$double \ exp: \ I_{0}=-7.5kA, \ T_{1}=100\mu s, \ T_{2}=6\mu s$"
    end

    if b==4 // define current used by Diendorfer et al. (1990)
      I(1,:)=heidler(-13000,0.73*10^(-6),3*10^(-6),2,tv)
      txt1(1)="$Heidler: \ I_{0}=-13kA, \ T_{1}=0.73\mu s,\ T_{2}=3\mu s,\ n=2$"
      I(2,:)=heidler(-7000,5*10^(-6),50*10^(-6),2,tv)
      txt1(2)="$Heidler: \ I_{0}=-7kA, \ T_{1}=5\mu s,\ T_{2}=50\mu s,\ n=2$"
    end
  end

  a=length(I)/length(tv) // real number of functions given

```

```
ta=length(tv) // number of timesteps of current
I0=zeros(1,ta) // build inital matix of current

for i=1:a
  I0=I0+I(i,:) // calculate current at origin
end

if H==0
  rhotop=-1 // set rhottop that I0 equals I(0,t)
end

write(%io(2),"figure of current")
write(%io(2),"-----------------")
write(%io(2),"  ")
fig=input("  Draw figures of current? 1 for yes, other for no: ")
write(%io(2),"  ")

if fig==1
  tmax2=input("  zoom at tmax in us (0 for none): ")
  write(%io(2),"  ")
  fn=input("     start figure number: ")
  write(%io(2),"  ")

  f1=scf(fn) // seclect figure
  clf(f1) // clear if exist
  tmax1=length(tv)
  if a>1
    for i=1:a
      plot2d((tv(1:tmax1)*10^6),(I(i,1:tmax1)*10^(-3)),style=(i+1)) // plot part i of current
    end
  end
  plot2d((tv(1:tmax1)*10^6),(I0(1:tmax1)*10^(-3)),style=1) // plot complete current
  f1.figure_size=[750 500]
  select a
    case 1, h1=legend(txt1(1),4),
    case 2, h1=legend(txt1(1),txt1(2),"$short-circuit \ current$",4),
    case 3, h1=legend(txt1(1),txt1(2),txt1(3),"$short-circuit \ current$",4),
    case 4, h1=legend(txt1(1),txt1(2),txt1(3),txt1(4),"$short-circuit \ current$",4),
  end
  xtitle("","$t \ / \ \mu s$","$I \ / \ kA$")

  if tmax2>0
    f2=scf((f1.figure_id+1)) // select figure
    clf(f2) // clear if exist



    tmax2=tmax2*10^(-6)

    tmax2=find(tv>=tmax2,1) // find next index according to zoom given
    if a>1
      for i=1:a
        plot2d((tv(1:tmax2)*10^6),(I(i,1:tmax2)*10^(-3)),style=(i+1)) // plot part i of zoomed current
      end
    end
    plot2d((tv(1:tmax2)*10^6),(I0(1:tmax2)*10^(-3)),style=1) // plot zoomed current
    f2.figure_size=[750 500]
    select a
      case 1, h1=legend(txt1(1),4),
      case 2, h1=legend(txt1(1),txt1(2),"$short-circuit \ current$",4),
      case 3, h1=legend(txt1(1),txt1(2),txt1(3),"$short-circuit \ current$",4),
      case 4, h1=legend(txt1(1),txt1(2),txt1(3),txt1(4),"$short-circuit \ current$",4),
    end
    xtitle("","$t \ / \ \mu s$","$I \ / \ kA$")
  end

  fig=input("  Draw figures of current derivative? 1 for yes, other for no: ")
  write(%io(2),"  ")

  if fig==1
    tmax2=input("  zoom at tmax in us (0 for none): ")
    write(%io(2),"  ")
    dI0=splin(tv,I0)

    f3=scf((f1.figure_id+2)) // select figure
    clf(f3) // clear if exist
    plot2d((tv(1:tmax1)*10^6),(dI0(1:tmax1)*10^(-9)),style=1) // plot complete current derivative
```

```
      f3.figure_size=[750 500]
      xtitle("","$t \ / \ \mu s$","$\frac{\mathrm{d}I}{\mathrm{d}t}  \ / \ \frac{
          kA}{\mu s}$")

      if tmax2>0
        f4=scf((f1.figure_id+3)) // select figure
        clf(f4) // clear if exist
        tmax2=tmax2*10^(-6)
        tmax2=find(tv>=tmax2,1) // find next index according to zoom given
        plot2d((tv(1:tmax2)*10^6),(dI0(1:tmax2)*10^(-9)),style=1) // plot zoomed
            current derivative
        f4.figure_size=[750 500]
        xtitle("","$t \ / \ \mu s$","$\frac{\mathrm{d}I}{\mathrm{d}t}  \ / \ \
            frac{kA}{\mu s}$")
      end
    end
  end
endfunction
```

tall/rsc.sci

```
// v3.1

// functions to calculate return stroke currents at channel origin

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics



// functionname heidler

// function to calculate current at channel origin
//          as a heidler function

// input: I0, T1, T2, n (scalar)
//        tvi ... timevector (vector 1:m)

// output: I0t=I(0,t) ... current at channel origin (vector 1:m)

function I0t=heidler(I0,T1,T2,n,tvi)

   eta=exp(-T1/T2*(n*T2/T1)^(1/n))
   frac1=(tvi/T1).^n // to reduce prozessing time
   I0t=I0/eta*(frac1.*(ones(frac1)+frac1).^(-1)).*exp(-tvi/T2)

endfunction



// functionname dexp

// function to calculate current at channel origin
//          as a double-exponential function

// input: I0, T1, T2 (scalar)
//        tvi ... timevector (vector 1:m)

// output: I0t=I(0,t) ... current at channel origin (vector 1:m)

function I0t=dexp(I0,T1,T2,tvi)

   I0t=I0*(exp(-tvi/T1)-exp(-tvi/T2))

endfunction
```

tall/heightvector.sci

```
// v3.1

// function to calculate the heigthvectors

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname heightvector

```

```
// input: tv ... timevector in s
//        Rp ... distance of field-point in km
//        H ... height of tall object in km
//        vf ... return stroke front wave speed in m/s

// output: zvc ... heightvector of channel
//         zvt ... heightvector of tall object

function [zvc,zvt] = heightvector(tv,Rp,H,vf)

  c=299792458 // speed of light
  zvt=0


  write(%io(2),"   ")
  write(%io(2),"   ")
  write(%io(2),"calculation of heightvector")
  write(%io(2),"---------------------------")
  write(%io(2),"   ")

// ---> tall object begin <---
  if H<>0 then
//    zvt=(2*Rp*tv*c+(tv*c)^2)./(2*(Rp+tv*c)) // calculate heightvector of tall object
    delz=c*tv(2)
    zvt=0:delz:(H+delz)
    zvt1=zvt(find(zvt>H,1))
    zvt2=zvt(find(zvt>H,1)-1)
    if (zvt1-max(zvt))>(max(zvt)-zvt2) then
      zvt=zvt(1:(find(zvt>H,1)-1))
    else
      zvt=zvt(1:(find(zvt>H,1)))
    end
  end
  Hi=max(zvt)

// ---> tall object end <---

// ---> channel begin <---
  D=c*(tv+Hi/vf)+sqrt(Rp^2+Hi^2) // auxiliary variable
  A=(c/vf)^2-1 // coefficient for quadratic equation for channel
  B=-2*c/vf*D // coefficient for quadratic equation for channel
  C=D.^2-Rp^2 // coefficient for quadratic equation for channel

  zvc=(-B-sqrt(B.^2-4*A*C))/(2*A) // calculate heightvector of channel
// ---> channel end <---

  write(%io(2),"   tall object:")
  write(%io(2),"   ")
  write(%io(2),"     --> number of segments []:")
  write(%io(2),(length(zvt)-1),"(7X,F10.0)")
  write(%io(2),"     --> calculated height [m]:")
  write(%io(2),max(zvt),"(7X,F10.2)")
  write(%io(2),"   ")

  write(%io(2),"   channel:")
  write(%io(2),"   ")
  write(%io(2),"     --> number of segments []:")
  write(%io(2),(length(zvc)-1),"(7X,F10.0)")
  write(%io(2),"     --> calculated height [m]:")
  write(%io(2),max(zvc),"(7X,F10.2)")
  write(%io(2),"   ")

endfunction
```

A.2.3 Calculation of current distribution

tall/distribution.sci

```
// v3.1

// function to calculate the current distribution in the tall object and the channel

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname distribution

// input: tv ... timevector in s
```

```
//          zvt ... heightvctor of tall object in m
//          zvc ... heightvctor of channel in m
//          I0 ... undisturbed current in A
//          rhotop ... current reflection coefficient at the top
//          rhobot ...current reflection coefficient at the bottom
//          nstop ... max index of successive multiple reflections
//          vf ... return stroke front wave speed in m/s

// output: Izt ... current distribution in the channel in A
//          zv ... heightvector in m (tall object and channel)

function [Izt,zv] = distribution(tv,zvt,zvc,I0,rhotop,rhobot,nstop,vf)

  c=299792458 // speed of light
  zvta=length(zvt) // ammount of segments (+1) for tall object
  zvca=length(zvc) // ammount of segments (+1) for channel
  tva=length(tv) // number of timesteps (+1)
  dI0=splin(tv,I0) // derivates for interpolation

  Iztt=zeros(zvta,tva) // build inital matix of current distribution in tall object
  Iztc=zeros(zvca,tva) // build inital matix of current distribution in channel

  write(%io(2),"   ")
  write(%io(2),"   ")
  write(%io(2),"calculation of current distribution")
  write(%io(2),"-----------------------------------")
  write(%io(2),"   ")
  write(%io(2),"   ... calculating current distribution")
  write(%io(2)," ")

  // ---> start: current distribution in tall object <---

  if zvta>1 then

      for j=1:zvta
        Iztt((zvta-j+1),j:tva)=(1-rhotop)*I0(1:(tva-j+1)) // calculate transmitted wave
      end

      stop=0
      m=0

      //// ---> start: reflected waves <---

      while stop==0
        m=m+1

        for j=1:zvta
          stu=(zvta+2*(m-1)*(zvta-1)+j-1) // index of starttime of the upward running wave on bottom
          std=(zvta+(2*m-1)*(zvta-1)+j-1) // index of starttime of the downward running wave on top
          if (stu>=tva) then
            stop=1
          else // upward runnning
            I0i=[zeros(1,stu-1) I0(1:(tva-stu+1))]
            Iztt(j,:)=Iztt(j,:)+(1-rhotop)*rhobot^m*rhotop^(m-1)*I0i // add n-th reflected wave on bottom
          end
          if (std>tva) then
            stop=1
          else // downward runnig
            I0i=[zeros(1,std-1) I0(1:(tva-std+1))]
            Iztt((zvta-j+1),:)=Iztt((zvta-j+1),:)+(1-rhotop)*(rhobot*rhotop)^m*I0i // add n-th reflected wave on top
          end
        end

        if m==nstop, stop=1, end

      end

      //// ---> end: reflected waves <---

  end

  // ---> end: current distribution in tall object <---

  // ---> start: current distribution in channel <---

  if length(zvt)>1 then

    for j=1:zvca
```

```
        Iztc(j,:)=[zeros(1,(j-1)) I0(1:tva-j+1)] // built injected wave
        ti=(zvc(j)-zvc(1))/c // starttime of reflected wave at heigth j
        I0i=(-1)*(rhotop)*interp((tv-ti),tv,I0,dI0,"by_zero") // built reflected
            wave
        Iztc(j,:)=Iztc(j,:)+I0i(1:tva) // add reflected wave
//          Iztc(j,:)=[zeros(1,(j-1)) (1-rhotop)*I0(1:tva-j+1)] // stop at return-
     front
      end

      for m=1:nstop
        for j=1:zvca
          ti=(zvc(j)+zvc(1)*(2*m-1))/c // starttime of n-th transmitted wave at
              heigth j
//          if ti>(zvc(j)-zvc(1)/vf) then ti=(zvc(j)-zvc(1)/vf); end  stop at
     return-front
          I0i=(1-rhotop)*(1+rhotop)*rhobot^(m)*rhotop^(m-1)*interp((tv-ti),tv,I0,
              dI0,"by_zero") // built n-th transmitted wave
          Iztc(j,:)=Iztc(j,:)+I0i(1:tva) // add n-th transmitted wave
        end
      end

      Iztc=triu(Iztc) // Heavyside

  end

  // ---> end: current distribution in channel <---

  if  length(zvt)>1 then
    zv=[zvt zvc(2:length(zvc))] // merge tall object and channel
    Izt=[Iztt;Iztc(2:length(zvc),:)] // merge tall object and channel
  else
    zv=zvc
    Izt=Iztc
  end

endfunction
```

A.2.4 Calculation, plotting and saving of fields

tall/field.sci

```
// v3.1

// function to calculate the fields

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname field

// input: Izt ... current distribution in the channel in A
//        zv  ... heightvector in m (tall object and channel)
//        tv  ... timevector in s
//        Rp  ... distance of field-point in km

// output:E=E(t) ... vertical electrical field in V/m
//        Ee, Ei, Er ... componenets of vertical electrical field in V/m
//        B=B(t) ... horizontal magnetic field in T
//        Bi, Br ... componenets of horizontal magnetic field in T
//        tf ... time of arrival of field at field point in s

function [E,Er,Ei,Ee,B,Br,Bi,tf] = field(Izt,zv,tv,Rp)

  Er=0
  Br=0
  Ei=0
  Bi=0
  Ee=0

  c=299792458 // speed of light
  eps0=8.8542*10^(-12) // permittivity
  mu0=4*%pi*10^(-7) // permeability

  tva=length(tv) // number of timesteps
  zva=length(zv) // number of channelsteps

  tf=sqrt(H^2+Rp^2)/c // starttime of field
  R=sqrt(Rp^2+zv.^2) // distance dipole - field point

  write(%io(2),"    ")
```

```
  write(%io(2),"   ")
  write(%io(2),"calculation of fields")
  write(%io(2),"---------------------")
  write(%io(2),"   ")

// ---> start: field calculation <---

//// ---> start: radiation component calculation <---
  write(%io(2),"   ... calculating radiation field")
  write(%io(2)," ")
  efac=(-1)*(Rp^2)./(c^2*R.^3*(2*%pi*eps0)) // factor electric radiation component
  bfac=(mu0*Rp)./(2*%pi*c*R.^2) // factor magnetic radiaion component
  dF=zeros(zva,tva)

  for j=1:zva
    dF(j,:)=splin(tv,Izt(j,:))
  end

  for i=1:tva
    Er(i)=inttrap(zv,(dF(:,i)'.*efac)) // intergrate over the channel
    Br(i)=inttrap(zv,(dF(:,i)'.*bfac))
  end
//// ---> end: radiation component calculation <---

//// ---> start: induction component calculation <---
  write(%io(2),"   ... calculating induction field")
  write(%io(2)," ")
  efac=(2*zv.^2-Rp^2)./(c*R.^4*(2*%pi*eps0)) // factor electric induction component
  bfac=(mu0*Rp)./(2*%pi*R.^3) // factor magnetic induction component
  dF=Izt

  for i=1:tva
    Ei(i)=inttrap(zv,(dF(:,i)'.*efac)) // intergrate over the channel
    Bi(i)=inttrap(zv,(dF(:,i)'.*bfac))
  end
//// ---> end: induction component calculation <---

//// ---> start: electrostatic component calculation <---
  write(%io(2),"   ... calculating electrostatic field")
  write(%io(2)," ")
  efac=(2*zv.^2-Rp^2)./(R.^5*(2*%pi*eps0)) // factor electrostatic component
  dF=zeros(zva,tva)

  for j=2:zva
    index=find(Izt(j,:)<>0,1)
    if index==[], break, end
    for i=index:tva
       dF(j,i)=dF(j,(i-1))+tv(2)*(Izt(j,(i-1))+Izt(j,i))/2
    end
  end

  for i=1:tva
    Ee(i)=inttrap(zv,(dF(:,i)'.*efac)) // intergrate over the channel
  end
//// ---> end: electrostatic component calculation <---

  E=Er+Ei+Ee
  B=Br+Bi

// ---> end: field calculation <---

endfunction
```

tall/plot_field.sci

```
// v3.1

// function to plot the fields and its components

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics

// functionname plot_field

// input: E=E(t) ... vertical electrical field in V/m
//        Ee, Ei, Er ... componenets of vertical electrical field in V/m
//        B=B(t) ... horizontal magnetic field in T
//        Bi, Br ... componenets of horizontal magnetic field in T
//        tv ... timevector
```

```
//          tzoom ... zoomtime in us
//          fn ... startnumber for figure
//          vf ... return stroke front wave speed in m/s
//          Rp ... distance of field-point in km
//          Hi ... height of tall object in km

// output: none

function []=plot_field(E,Ee,Ei,Er,B,Bi,Br,tv,tzoom,fn,vf,Rp,Hi)

      tzoom=tzoom*10^(-6)
      txt1="TL \ model: \ v_{f}="+string(vf*10^(-8))+"\cdot 10^8 \frac{m}{s}, \
           starttime="+string(tf*10^6)+"\mu s, \ object-heigth="+string(Hi)+"m"

      if abs(max(E))<abs(min(E)) then // allways plot positiv
          pe=-1
      else
          pe=1
      end

      if abs(max(B))<abs(min(B)) then // allways plot positiv
          pb=-1
      else
          pb=1
      end

      f1=scf(fn) // select figure
      clf(f1) // clear if exist
      tmax=length(tv)
      plot2d((tv(1:tmax)*10^6'),[Ee(1:tmax) Ei(1:tmax) Er(1:tmax) E(1:tmax)]*pe,
           style=[2 13 5 1])
      f1.figure_size=[750 750]
      h1=legend("$electrostatic \ component$","$induction \ component$","
           $radiation \ component$","$electrical \ field$",1)
      xtitle("$electrical \ field \ in \ "+string(Rp*10^(-3))+"km: \ "+txt1+"$","
           $t \ / \ \mu s$","$E \ / \ \frac{V}{m}$")

      if tzoom>0
         f2=scf(f1.figure_id+1) // select figure
         clf(f2) // clear if exist
         tzoom=find(tv>=tzoom,1) // find next index according to zoom given
         plot2d((tv(1:tzoom)*10^6)',[Ee(1:tzoom) Ei(1:tzoom) Er(1:tzoom) E(1:
              tzoom)]*pe,style=[2 13 5 1])
         f2.figure_size=[750 750]
         h1=legend("$electrostatic \ component$","$induction \ component$","
              $radiation \ component$","$electrical \ field$",1)
         xtitle("$electrical \ field \ in \ "+string(Rp*10^(-3))+"km: \ "+txt1+"$
              ","$t \ / \ \mu s$","$E \ / \ \frac{V}{m}$")
      end

      f3=scf(f1.figure_id+2) // select figure
      clf(f3) // clear if exist
      plot2d((tv(1:tmax)*10^6)',[(Bi(1:tmax)*10^6) (Br(1:tmax)*10^6) (B(1:tmax)
           *10^6)]*pb,style=[13 5 1])
      f3.figure_size=[750 750]
      h1=legend("$induction \ component$","$radiation \ component$","$magnetic \
           field$",1)
      xtitle("$magnetic \ field \ in \ "+string(Rp*10^(-3))+"km: \ "+txt1+"$","$t
            \ / \ \mu s$","$B \ / \ \mu T$")

      if tzoom>0
         f4=scf(f1.figure_id+3) // select figure
         clf(f4) // clear if exist
         plot2d((tv(1:tzoom)*10^6)',[(Bi(1:tzoom)*10^6) (Br(1:tzoom)*10^6) (B(1:
              tzoom)*10^6)]*pb,style=[13 5 1])
         f4.figure_size=[750 750]
         h1=legend("$induction \ component$","$radiation \ component$","$magnetic
               \ field$",1)
         xtitle("$magnetic \ field \ in \ "+string(Rp*10^(-3))+"km: \ "+txt1+"$",
              "$t \ / \ \mu s$","$B \ / \ \mu T$")
      end

endfunction
```

tall/save_field.sci

```
// v3.1

// module to save the field and its components

// written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
```

```
// Institute of Power Systems and Energy Economics

// functionname save_field

// input: E=E(t) ... vertical electrical field in V/m
//        Ee, Ei, Er ... componenets of electrical field in V/m
//        B=B(t) ... horizontal magnetic field in T
//        Bi, Br ... componenets of magnetic field in T
//        tf ... time of arrival of field at field point in s
//        tv ... timevector
//        H ... height of tall object in m
//        rhotop ... current reflection coefficient at the top
//        rhobot ...current reflection coefficient at the bottom
//        vf ... upward-propagating return-front-wave speed in m/s
//        Rp ... distance of fieldpoint in m

// output: none

function []=save_field(E,Ee,Ei,Er,B,Bi,Br,tf,tv,H,rhotop,rhobot,vf,Rp)

  write(%io(2),"save field")
  write(%io(2),"----------")
  write(%io(2),"  ")
  sav=input("  save field and components to file? 1 for yes, other for no: ")
  write(%io(2),"  ")
  if sav==1 then
    errcatch(999,"continue")
    fnw=input("   filename: ","s")
    fid=mopen(fnw,"w")
    if fid==-1 then
      write(%io(2),"   Error: Cannot open file for writing!")
    else
      write(%io(2),"   ... writing data to file: "+fnw)
      write(%io(2),"   ")
      mfprintf(fid,"%s \t %3.2f %s \n","Height:",H,"m")
      mfprintf(fid,"%s \t %1.5f \n","rhotop:",rhotop)
      mfprintf(fid,"%s \t %1.5f \n","rhobot:",rhobot)
      mfprintf(fid,"%s %3.2f %s \n","field and components at fieldpoint: ",(Rp*10^(-3)),"km")
      mfprintf(fid,"%s \t \t %s \t %s \t %s \t %s \t %s \t %s \t %s \n","Name:","E","Ee","Ei","Er","B","Bi","Br")
      mfprintf(fid,"%s \t %i \n","Length:",length(E))
      mfprintf(fid,"%s \t %e %s \n","Starttime:",tf,"s")
      mfprintf(fid,"%s \t %e s \n","Unit X-Axis:",tv(2))
      mfprintf(fid,"%s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \n","Unit Y-Axis:","V/m","V/m","V/m","V/m","T","T","T")
      mfprintf(fid,"%s \n","Data:")
      for i=1:length(E)
        mfprintf(fid,"%1.5e \t %1.5e \t %1.5e \t %1.5e \t %1.5e \t %1.5e \t %1.5e \n",E(i),Ee(i),Ei(i),Er(i),B(i),Bi(i),Br(i))
      end
      mclose(fid)
    end
    errcatch(-1)
  end

endfunction
```

A.2.5 Definitionfile

tall/def_tall.txt

```
definitionfile for tall object

remarks:

********** ********** **********
start_of_def

tend        51
ta          5000
H           168
rhotop      -0.53
rhobot      0.7
n           20
vf          1.5
Rp          5

currents
h     9.9    0.072    5    2
de    7.5    100      6    0
x     0      0        0    0
```

```
x    0    0    0    0
x    0    0    0    0

end_of_def
********** ********** **********

format of data:

tend         endtime of field calculation in [us]
ta           number of timesteps for field calculation
H            height of tall object in [m]
rhotop       current reflection coefficient at the top (tall oblect - channel)
rhobot       current reflection coefficient at the bottom (tall object - ground)
n            index of successive multiple reflections
vf           return stroke front wave speed in the channel in [10^8 m/s]
Rp           distance of field point [km]

currents
a    b    c    d    e    f

a ... function of current: de for double exponential, h for heidler-function, x
     for no more currents defined
b ... I0 in [kA]
c ... T1 in [us]
d ... T2 in [us]
e ... n (need only for heidler-function)

for copy/paste:

Nucci90:

h    -9.9    0.072    5    2
de   -7.5    100      6    0

Diendorfer90:

h    -13    0.73    3    2
h    -7     5       50   2
```

List of Tables

List of Figures

Bibliography

[Baba and Rakov, 2005] Baba, Y. and Rakov, V. A. (2005). On the use of lumped sources in lightning return stroke models. *J. Geophys. Res.*, 110(D3).

[CEA-CNRS-INRIA, 2013] CEA-CNRS-INRIA (2013). http://www.cecill.info.

[Diendorfer and Uman, 1990] Diendorfer, G. and Uman, M. A. (1990). An improved return stroke model with specified channel-base current. *J. Geophys. Res.*, 95(D9):13621–13644.

[Heidler, 1985] Heidler, F. (1985). travelling current source model for lemp calculation. In *Proc. 6th Int. Symp. on Electromagnetic Compatibility, Zürich, Switzerland*, pages 157–62.

[Nucci et al., 1990] Nucci, C. A., Diendorfer, G., Uman, M. A., Rachidi, F., Ianoz, M., and Mazzetti, C. (1990). Lightning return stroke current models with specified channel-base current: A review and comparison. *J. Geophys. Res.*, 95(D12):20395–20408.

[Pavanello et al., 2007] Pavanello, D., Rachidi, F., Rakov, V. A., Nucci, C. A., and Bermudez, J. L. (2007). Return stroke current profiles and electromagnetic fields associated with lightning strikes to tall towers: Comparison of engineering models. *Journal of Electrostatics*, 65(5–6):316–321.

[Rakov, 1997] Rakov, V. A. (1997). Lightning electromagnetic fields: Modeling and measurements. *Proceedings of the 12th International Zurich Symposium on Electromagnetic Compatibility*, pages 59–64.

[Rakov, 2007] Rakov, V. A. (2007). Lightning return stroke speed. *Journal of Lightning Research*, 1:80–89.

[Rakov and Dulzon, 1991] Rakov, V. A. and Dulzon, A. A. (1991). A modified transmission line model for lightning return stroke calculation. In *Proc. 9th Int. Symp. on Electromagnetic Compatibility, Zürich, Switzerland*, pages 229–35.

[Rakov and Uman, 2007] Rakov, V. A. and Uman, M. A. (2007). *Lightning: Physics and Effects*. Cambridge University Press, New York.

[Scilab-Enterprises, 2014] Scilab-Enterprises (2014). http://www.scilab.org.

[Thottappillil et al., 1997] Thottappillil, R., Rakov, V. A., and Uman, M. A. (1997). Distribution of charge along the lightning channel: Relation to remote electric and magnetic fields and to return-stroke models. *J. Geophys. Res.*, 102(D6):6987–7006.

[Uman and McLain, 1969] Uman, M. A. and McLain, D. K. (1969). Magnetic field of lightning return stroke. *J. Geophys. Res.*, 74(28):6899–6910.