Die approbierte Originalversion dieser Diplom-/ Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich. http://www.ub.tuwien.ac.at

> TU UB WIEN Universitätsbi

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.

http://www.ub.tuwien.ac.at/eng



# 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

$\mathbf{C}$	onter	nts	iii
A	ckno	wledgment	iv
A	bstra	$\mathbf{ct}$	$\mathbf{v}$
1	Intr	oduction	1
	1.1	Current at the Striking Point	1
	1.2	Simplifications of the Lightning Channel	2
	1.3	Current along the Channel based on Return Stroke Engineer-	
		ing Models	2
		1.3.1 Transmission Line (TL) Model	6
		1.3.2 Modified Transmission Line Model (MTLE) with Ex-	
		ponential Decay	$\overline{7}$
		1.3.3 Traveling Current Source (TCS) Model	$\overline{7}$
		1.3.4 Diendorfer-Uman (DU) Model	9
	1.4	Electric and Magnetic Field Calculation	9
	1.5	Lightning Strike to a Tall Object	14
<b>2</b>	$\mathbf{Ligl}$	ntning Strike to the Ground	26
	2.1	Basics	26
		2.1.1 Current and the Time-Vector	26
		2.1.2 Discretization of the Channel and the Channel-Vector	27
		2.1.3 Other Parameter	28
	2.2	Current Distribution and Field Calculation	29
		2.2.1 Current at Height $z'$	29
		2.2.2 Field Calculation	30
	2.3	Using the Program	32
		2.3.1 Defining the Input Parameters	32
		2.3.2 Calculating the Field for one Model	35
		2.3.3 Calculating the Field of all Models	35
		2.3.4 User Intervention $\ldots$	35
	2.4	Description of the Program Modules	40
		2.4.1 start	40

	$2.4.2$ func_par
	2.4.3 func_rsc
	2.4.4 field
	2.4.5 e_field
	2.4.6 i_field
	2.4.7 r_field
	2.4.8 d_field
	2.4.9 func_plot
	2.4.10 func_save
3 Lig	htning Strike to a Tall Object
3.1	Basics
	3.1.1 Current and the Time-Vector
	3.1.2 Discretization of the Channel
3.2	Current Distribution and Field Calculation
	3.2.1 Current Distribution
	3.2.2 Field Calculation
3.3	Using the Program
3.4	Defining the Parameters
3.5	Variables
3.6	Description of the Program Modules
	3.6.1 start
	3.6.2 def
	3.6.3 rsc
	3.6.4 heightvector
	3.6.5 distribution
	3.6.6 field
	$3.6.7$ plot_field $\ldots$
	3.6.8 save_field
4 Eva	luation of the Programs
4.1	Strike to the Ground
	4.1.1 Comparison with [Nucci et al., 1990]
	4.1.2 Comparison with [Diendorfer and Uman, 1990]
4.2	Strike to a Tall Object
	4.2.1 Current and Current Distribution
	4.2.2 Fields
5 Sur	nmary and Outlook
5.1	Summary
5.2	Outlook

$\mathbf{A}$	$\mathbf{List}$	ings		<b>67</b>
	A.1	Lightn	ing Strike to the Ground	67
		A.1.1	Startmodule	67
		A.1.2	Definition of Parameters	69
		A.1.3	Calculation of fields	74
		A.1.4	Plotting and saving of data	81
		A.1.5	Definitionfile	85
	A.2	Lightn	ing Strike to a Tall Object	86
		A.2.1	Startmodule	86
		A.2.2	Definition of Parameters	87
		A.2.3	Calculation of current distribution	92
		A.2.4	Calculation, plotting and saving of fields	94
		A.2.5	Definitionfile	97
Lis	st of	Tables	3	99
Lis	st of	Figure	2S	101
Bi	bliog	raphy		102

iii

# 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).

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

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

	$I_0$ / kA	$T_1 / \mu s$	$T_2 / \mu s$	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

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

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]:

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

 $u(t-\frac{z'}{v_f})$ : Heavy side 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 height nor an attenuation according to the propagation. So the current at any height z' at any time t is given by

$$I(z',t) = I(0,t - \frac{z'}{v_f})$$
(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  $\lambda$  which is normally assumed to be about 2000m.

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

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

8

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

$$I(z',t) = u(t - \frac{z'}{v_f}) \cdot I(0,t + \frac{z'}{c})$$
(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 coronacurrent 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  $\tau_B$  and  $\tau_C$  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.

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



current in the channel at different times



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 = 0and 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]:



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].

$$E_{z}(r,t) = \frac{1}{2\pi\varepsilon_{0}} \int_{0}^{H(t)} \left[ \frac{2z'^{2} - r^{2}}{R^{5}(z')} \int_{\frac{z'}{v_{f}} + \frac{R(z')}{c}}^{t} I\left(z', \tau - \frac{R(z')}{c}\right) d\tau + \frac{2z'^{2} - r^{2}}{cR^{4}(z')} I\left(z', t - \frac{R(z')}{c}\right) - \frac{r^{2}}{c^{2}R^{3}(z')} \frac{\partial I\left(z', t - \frac{R(z')}{c}\right)}{\partial t} \right] dz' - \frac{1}{2\pi\varepsilon_{0}} \frac{r^{2}}{c^{2}R^{3}(H(t))} I\left(H(t), \frac{H(t)}{v_{f}}\right) \frac{dH(t)}{dt}$$
(1.8)

$$B_{\phi}(r,t) = \frac{\mu_0}{2\pi} \int_0^{H(t)} \left[ \frac{r}{R^3(z')} I\left(z', t - \frac{R(z')}{c}\right) + \frac{r}{cR^2(z')} \frac{\partial I\left(z', t - \frac{R(z')}{c}\right)}{\partial t} \right] dz' + \frac{\mu_0}{2\pi} \frac{r}{cR^2(H(t))} I\left(H(t), \frac{H(t)}{v_f}\right) \frac{dH(t)}{dt}$$
(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 proportial to  $Q = \int I(t) dt$  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  $(0 \le z' \le h)$  with a configuration shown in Figure 1.9 the current distribution is given by

$$I(z',t) = \frac{1-\varrho_{top}}{2} \sum_{n=0}^{\infty} \left[ \varrho_{bot}^n \, \varrho_{top}^n \, I_{SC} \left( h, t - \frac{h-z'}{c} - \frac{2nh}{c} \right) \right. \\ \left. + \, \varrho_{bot}^n \, \varrho_{top}^n \, I_{SC} \left( h, t - \frac{h+z'}{c} - \frac{2nh}{c} \right) \right] \text{ for } 0 < z \le h \quad (1.10)$$

In the channel  $(z' \ge h)$  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].

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

$$(1.11)$$

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).

$$I(z',t) = \left[i_0\left(h,t-\frac{z-h}{v}\right) - \varrho_{top} i_0\left(h,t-\frac{z-h}{c}\right) + (1-\varrho_{top})\left(1+\varrho_{bot}\right) \sum_{n=0}^{\infty} \left[\varrho_{bot}^{n+1} \varrho_{top}^n i_0\left(h,t-\frac{z+h}{c}-\frac{2nh}{c}\right)\right] \\ \cdot u\left(t-\frac{z-h}{v}\right) \text{ for } 0 < z \le h$$

$$(1.12)$$

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

Because the return stroke front propagates with  $v_f \approx \frac{1}{3} \dots \frac{2}{3} \cdot c$ , 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.



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.

















# 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.  $0.2\mu$ s (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.  $0.1\mu$ s (see Figure

1.1). The accuracy depends on the number of time-steps in that window. The best results were achieved with  $\geq 10$  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  $z'_n$ .

$$\mathbf{t}_{\mathbf{C}} + \mathbf{t}_{\mathbf{A}} - t_B = \mathbf{t} \tag{2.1}$$

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

$$\mathbf{t} = \begin{pmatrix} 0 & \Delta t & 2\Delta t & \cdots & n\Delta t \end{pmatrix} \tag{2.2}$$

and the height-vector of the channel



Figure 2.1: Method of discretization of the channel.

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

the runtime equation reveals

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

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

$$\mathbf{z}^{\prime 2} \cdot \left[ \left( \frac{c}{v_f} \right)^2 - 1 \right] - \mathbf{z}^{\prime} \cdot 2 \frac{c}{v_f} (R_p + c\mathbf{t}) + \left[ (R_p + c\mathbf{t})^2 - R_p^2 \right] = 0 \quad (2.5)$$

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 timesteps. 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  $\lambda$ .

```
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.

```
 \begin{array}{l} 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)) \quad Izt(1:(ca-j+1))] \end{array}
```
#### 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).

```
\begin{split} 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))] \end{split}
```

#### 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 (loopvariable i). Step by step the integration is done for every height (loopvariable 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<sup>2</sup>)./(c<sup>2</sup>\*R<sup>3</sup>\*(2\*%pi\*eps0)) bfac=(mu0\*Rp)./(2\*%pi\*c\*R<sup>2</sup>)

the value of the current at the discontinuity

```
\label{eq:listant} \begin{array}{l} tsstart = cv / vf + cv / c \\ Iht = interp \left( tsstart \ , tv \ , I0t \ , dI0t \ , "by_zero" \right) \end{array}
```

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 = 0s, the end-time is equal to the period of time the field is calculated over. The unit is  $\mu$ s.
- **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  $10^8$ m/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.

```
00
                                              Scilab Console
😰 🕒 👗 🗊 🚺 🗭 🗛 📇 🗧 🖉 🗶 🕖
Scilab Console
**********
* return-stroke model calculation *
**********
1 ... enter parameter via keyborad
2 ... read parameter from file
 3 ... calculate one model with all components
4 ... save field of one model with all components
 5 ... calculate all models
6 ... save field of all models
 8 ... interrupt
9 ... end
------
Your choice please: 1
timevector for field definition
   Endtime [us]: 50
amount of timesteps (best range: 1000 to 5000): 5000
channel definition
   upward-propagating return-front-wave speed in 10^8 m/s: 1.3
fieldpoint definition
   distance of fieldpoint in km: 100
current I(0,t) definition
   How many functions define the current? 2
   1 for Heidler, 2 for Double-Exponetial, 3 Nucci90, 4 DU90 other to break: 3
modelparameter definition
input parameter if needed
exponential decay coefficient in m for MTLE: 2000
   discharge time constant for breakdown current in us for DU: 0.1 discharge time constant for corona current in us for DU: 0.1 \,
timevector calculation
     --> length of timestep in us:
0.01000
channelvector calculation
     --> amount of channel pieces:
             5000.
     --> calculated height of channel in m:
           6410.98
figure of current
Draw figures of current? 1 for yes, other for no:
```

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

**model-parameter definition:** The model specific parameters are the exponential decay  $\lambda$  in m for the MTLE model (section 1.3.2) and the time constants for the breakdown respectively the corona current in  $\mu$ s 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.

```
0 0
                                             Scilab Console
 🕜 🔚 : 👗 🗊 i 📄 : 🗛 📇 : 🚍 : 🕮 : 🍩 🔞
Scilab Console
    .. read parameter from file
3 ... calculate one model with all components
4 ... save field of one model with all components
5 ... calculate all models
6 ... save field of all models
8 ... interrupt
9 ... end
_____
Your choice please: 3
model definition
   model type: 1 for TL, 2 for MTLE, 3 for TCS, 4 for DU, other to break: 1
figure of field
  Draw figure of field? 1 for yes, other for no: 2
   ... calculating radiation component
  ... calculating induction component
   ... calculating electrostatic component
*****
 return-stroke model calculation *
1 ... enter parameter via keyborad
2 ... read parameter from file
3 ... calculate one model with all components
4 ... save field of one model with all components
5 ... calculate all models
6 ... save field of all models
8 ... interrupt
9 ... end
 -------
Your choice please: 4
save field
 save field and components to file? 1 for yes, other for no: 1
  filename: 001.txt
... writing data to file: 001.txt
                                                                                                      .
```

Figure 2.4: Screenshot: Calculation of one Model.

TL Model: <u>Vf</u> =1.3000+08 m/s field and components at <u>fieldpg</u> Name: E							
Length: 2501 Starttime: 3.33564le-04 s Unit X-Axis: 1.200000e-08 Unit Y-Axis: V/m V/m	Lint: 100.00 km Ei Er Ed B K.s V/m V/m T	Bi Br Bd T T T					
Data: 0.00000e+00 0.00000e+00 8.21173e-02 2.77927e-11 3.02070e-01 2.11750e-10 6.03202-01 8.24152e-10	0.00000e+00 1.54511e-06 8.68185e-06 2.53645e-06	0.00000e+00 8.21158e-02 3.02061e-01 6.02504e-01	0.00000e+00 0.00000e+00 0.00000e+00	0.00000+00 -2.73914e-10 -1.00760e-09 -7.000876-09	0.00000e+00 -5.15392e-15 -2.89596e-14 -8.46071e-14	0.00000+00 -2.73909e-10 -1.00757e-09 -7 00074-09	0.00000e+00 0.00000e+00 0.00000e+00
0.022299-01 0.4105E-10 9.26760e-01 2.24162e-09 1.23435e400 4.87504e-09 1.50600e+00 9.11840e-09 1.73569e+00 1.53218e-08	<pre>c</pre>	1.2349e-01 9.26706e-01 1.23443e+00 1.50586e+00 1.73549e+00	0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00	-2.003626-09 -3.091346-09 -4.117946-09 -5.023496-09 -5.789656-09	-0.400/le-14 -1.78250e-13 -3.10096e-13 -4.76801e-13 -6.73570e-13	-2.003/46-09 -3.091166-09 -4.117636-09 -5.023016-09 -5.788986-09	0.000000+00 0.000000+00 0.000000+00 0.000000+00 0.000000+00
1.92578e+00 2.37829e-08 2.08148e+00 3.47487-08 3.47487e-08 3.47487e-08 4.84220e-08 2.31260e+00 6.49683e-08 2.39789e+00 8.45231e-08 2.39789e+00 8.45231e-08	2.68456e-04 3.41180e-04 4.18972e-04 5.00904e-04 5.86225e-04	1.92551e+00 2.08114e+00 2.20822e+00 2.31210e+00 2.39730e+00	0.00000+00 0.00000+00 0.00000+00 0.00000+00 0.00000+00	-6.42370e-09 -6.94309e-09 -7.36724e-09 -7.71401e-09 -7.99851e-09	-8.95472e-13 -1.13806e-12 -1.39754e-12 -1.67084e-12 -1.95544e-12	-6.42281e-09 -6.94195e-09 -7.36584e-09 -7.71234e-09 -7.99656e-09	0.00000+00 0.00000+00 0.00000+00 0.00000+00 0.00000+00
2.52646e400 1.07197-07 2.52646e400 1.33083e-07 2.57499e400 1.62555e-07 2.61564e400 1.94777e-07 2.64984e400 2.30702-07 2.64984e400 2.70074e-07 2.70326e400 3.12930e-07	6.74332e-04 7.64740e-04 8.57060e-04 9.50976e-04 1.04623e-03 1.14261a-03 1.23956-03	2.467526+00 2.52569e400 2.57414e+00 2.61469e+00 2.64879e+00 2.67760e+00 2.67760e+00 2.70202e+00	0.000000++00 0.000000++00 0.000000++00 0.000000++00 0.000000+90 0.000000+90 0.000000+90	-8.23303-09 -8.42737e-09 -8.72484e-09 -8.83891e-09 -8.93532e-09 -9.01711e-09	-2.14934e-12 -2.550906-12 -3.152126-12 -3.172126-12 -3.489856-12 -3.811356-12 -4.136046-12	-8.23078e-09 -8.42482e-09 -8.72166e-09 -8.83547e-09 -8.83547e-09 -8.93151e-09 -9.01298e-09	0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00
2.72413e+00 3.53932e-07 7.7193e+00 4.09219e-07 2.771956+00 4.62702e-07 2.77026e+00 5.19771e-07 2.78146e+00 5.89442-07 2.78146e+00 5.89442-07	1.33810e-03 1.43694e-03 1.53637e-03 1.53631e-03 1.73651e-03	2.72280e+00 2.74052e+00 2.75566+00 2.7556e+00 2.76972e+00 2.77973e+00	0.00000+00 0.00000+00 0.00000+00 0.00000+00 0.00000+00 0.00000+00	-9.08674e-09 -9.14618e-09 -9.19703e-09 -9.27797e-09 -9.27797e-09	-4.46343e-12 -4.79313e-12 -5.12479e-12 -5.479e-12 -5.79293e-12 -5.79293e-12	-9.08228e-09 -9.14139e-09 -9.19191e-09 -9.23518e-09 -9.27218e-09	0.00000+00 0.000000+00 0.000000+00 0.00000+00 0.00000+00 0.00000+00
2.79316+00 0.447316-07 2.799316+00 7.126506-07 2.806366+00 7.842096-07	1.03/41e-03 1.93846e-03 2.03979e-03	2.79737e+00 2.80432e+00 2.80432e+00	0.00000e+00 0.00000e+00 0.00000e+00	-9.33751e-09 -9.36104e-09 -9.36104e-09	-0.120930-12 -6.466030-12 -6.804010-12	-9.33104e-09 -9.33104e-09 -9.35423e-09	0.00000e+00 0.00000e+00 0.00000e+00

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

TL Model: Xr=1.3000-08 m/s         Mill Model: Xr=1.3000-08 m/s         Mill Model: Xr=1.3000-08 m/s           TCS Model: Xr=1.3000-08 m/s         TC=0.000001 s         TC=0.000001 s         TC=0.000001 s           TCS Model: Xr=1.3000-08 m/s         TC=0.000001 s         TC=0.000001 s         TC=0.000001 s           TCS Model: Xr=1.3000-08 m/s         TC=0.000001 s         TC=0.000001 s         TC=0.000001 s           TCS Model: Xr=1.3000-08 m/s         TC=0.000001 s         TC=0.000001 s         TC=0.000001 s           Mume:         TEIL         T00.00 m         T         T         T           Mome:         T.100.00 m         V/m V/m         V/m         V/m         V/m         V/m           Mome:         T.100.00 m         T         T         T         T         T         T           Lengther         2301         E.11         E.100.00 m         0.00000000000000000000000000000000000	
Wirt Kic Model: M-1, 3000-08 m/s, lambda-2000 m           TCS Model: M-1, 3000-08 m/s, Tc-0.000001 s, Tc-0.000000+00         B. du           TCS Model: M-1, 100.00 km         E. du         B. dl         B. dl           Fields at fieldpoint: 100.00 km         E. du         B. dl         B. dl           Stortike: 3.335641e-04 s         E. du         B. dl         B. dl           Stortike: 3.335641e-04 s         E. du         B. dl         B. dl           Stortike: 3.335641e-04 s         E. du         B. dl         B. dl           Stortike: 3.335641e-04 s         E. dl         B. dl         B. dl           Stortike: 3.335641e-04 s         E. dl         B. dl         B. dl         E. dl           Stortike: 3.335641e-04 s         E. 31735e-01         E. 273914e-10         E. 11477e-09         E. 773600000+00         E. 0000000+00         E. 0000000+00         E. 77347e-09         E. 77347e-09         E. 77346e-09         E. 71347e-09	
TCS Model: YF41.300-08 /m Sr. The 0000001 s, Tc=0.0000001 s       TCS Model: YF41.300-08 /m Sr. The 0000001 s, Tc=0.0000001 s         Field of fieldpoint: 100.00 /m       E.M. E.M.L. E.M.L.E.M.L. E.M.L. E.M.L. E.M.L. E.M.L. E.M.L. E.M.L. E.M.L. E.M.L. E.M	
DU Model: Xrfa1.3000-08 m/s, Tr=0.0000001 s, Tc=0.0000001 s, Tc=0.000000 km       B_utl.       B_mtl.e	
инецая ст. Д.	
Nome:         EdL         BtLi         EdL         BtLi         BtLi <td></td>	
Sturttime:       3.335641e-04 s         Sturttime:       1.200000e-08 s         Unt X -Axis:       1.200000e-08 s         Unt Y -Axis:       1.20000e-08 s         Unt Y -Axis:       1.335641e-04 s         0.00000e+00       0.00000e+00       0.00000e+00         0.00000e+01       0.00000e+00       0.00000e+00         0.15561e+01       2.73914e-10       -1.1747a-09         3.02570e-01       3.01457e-01       8.21175e-01       -2.73914e-10         3.02570e-01       3.01457e-01       3.22492e+00       -3.66734e-09       -3.1754e-09         3.02550e-01       1.23550e+00       -3.09146e-09       -4.11754e-09       -1.10752e-08         3.02550e-01       1.23550e+00       1.75560e+00       -5.09124e-09       -1.10752e-08         3.02550e-01       1.23550e+00       1.75560e+09       -5.10752e-08       -5.11754e-05         1.25560e+00       1.73550e+09       -7.5754e-09       -1.10161e-08       -6.6737e-09         1.25560e+00       1.75560e+09       -5.10772e+08       -7.5754e-09       -1.101674e-	
Unit X.Axis:       1:200000e-08 s         Unit Y.Axis:       1:20000e-08 s         Unit Y.Axis:       1:335641e-04 s         Unit Y.Axis:       1:335641e-04 s         Unit Y.Axis:       1:335641e-04 s         Unit Y.Axis:       1:335641e-04 s         Unit Y.Axis:       0:00000e+00       0.00000e+00       0.00000e+00         0:00000e+00       0.00000e+00       0.00000e+00       0.00000e+00         0:00000e+01       0.00000e+00       0.00000e+00       0.00000e+00         0:00000e+02       0.00000e+00       0.00000e+00       0.00000e+00         0:00000e+01       1.0551175e-01       1.11734e-03       -1.11745e-09       -1.11745e-09         0:00000e+00       1.25503e+00       1.25108+00       1.25108+00       -1.00775e-09       -3.87055e-09       -5.7142e-09         0:00000e+00       1.25503e+00       1.25704e+00       -3.03146e-09       -1.11734e-09       -3.07142e-09       -1.07555e-09       -1.07555e-09       -1.07555e-09       -5.75142e-09       -5.75142e-09       -5.75142e-09       -5.75142e-09       -5.75142e-09       -5.75142e-09       -5.75142e-09       -5.75146e-08       -5.75146e-08       -5.75146e-08       -5.75146e-08       -5.75146e-08       -5.75146e-08       -5.75144e-09       -5.75144e-09       -5.75146e-	
15t x-Value:       3.335641e-04 s         Unit Y-Axis:       V/m	
Unit Y-Axis:         V/m         V/m         V/m         T         T         T           Data:         0.000000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.00000+00         0.0000+00         0.00000+00	
Data:         Data:         Commone-point         0.00000e+00         0.0000e+00         0.0000e+00         0.0000e+00         0.0000e+00         0.0000e+00 <th0.000< th="">         0.0000e+00         <th0.000< th=""></th0.000<></th0.000<>	
0.000000+00         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.000000+400         0.00000+4400         0.00000+4400         0.00000+400         0.00000+400         0.00000+4400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400         0.00000+400 <th0.1017620+90< th="">         0.00000+400         0.000</th0.1017620+90<>	
8.21173e-02       3.22175e-01       8.21173e-02       -7.73914e-10       -1.17473e-09       -2.7         3.02077e-01       3.01942e-01       1.16261e400       5.24811e-01       -1.0076e-09       -1.0077e-09       -3.78735e-09       -2.7         3.02077e-01       3.01942e-01       1.16261e400       5.22876e-01       -1.0076e-09       -1.0077e-09       -3.78734e-09       -3.78734e-09       -5.7334e-09       -5.73334e-09       -1.10714e-08       -5.73344e-08       -5.73344e-08       -5.73344e-08       -5.73344e-08       -5.733346e-09       -1.107146e-08	0e+00 0.00000e+00
3.020706-01         3.01944e-01         1.162616+00         5.24811e-01         -1.00756e-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -3.87856-09         -5.67344-09         -3.87856-09         -5.67344-09         -3.87856-09         -5.67344-09         -5.1311874-09         -5.1311874-09         -5.1311874-09         -5.1311874-09         -5.1311874-09         -5.1311874-09         -5.1311874-09         -5.1311874-09         -5.1311874-09         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -5.1311874-08         -6.6         -5.1312666+08         -6.6         -5.131266+09         -1.1075774-08         -1.1075774-08         -5.131266+08         -6.6         -5.131266+08         -6.6         -5.131266+08         -6.6         -5.131266+08         -6.6         -5.131266+08         -6.100         -5.110774-08         -6.100<	73e-09 -2.73914e-10
6.02529e-01         6.02058e-01         2.00062e+00         9.82226e-01         -2.00982e-09         -5.03334e-09         -5.03334e-09         -5.03334e-09         -3.2           9.25750e+01         9.235593e+01         2.06660e+00         1.24010e+00         -3.03134e-09         -5.03875e-09         -6.67334e-09         -4.43           9.25559e+00         1.73143e+00         3.22209e+00         1.75410e+00         -5.03349e-09         -1.10556e-09         -1.07572e-08         -5.1           1.73559e+00         1.73143e+00         3.22209e+00         1.7544e-09         -1.07572e-09         -5.1           1.73559e+00         1.73143e+00         3.32209e+00         1.75144e-09         -5.1         -6.42370e         -5.1           1.73559e+00         1.73143e+00         3.32209e+00         1.75142e-09         -1.10813e-08         -5.1           2.08148e+00         1.93761e+00         -5.73376e-09         -5.11074e-09         -1.10746e-09         -5.13126e-09         -1.10746e-09         -5.11164e-08         -6.64           2.031250e+00         3.32803e+00         1.93761e+00         -7.33746e-09         -1.10746e-09         -6.4         -6.4           2.23759e+00         3.32803e+00         1.93761e+00         -7.33746e-09         -1.10746e-08         -6.4         -6.4	05e-09 -1.75058e-09
9.26760e-01 9.25676e-01 9.25676e-01 1.25250e+00 1.32510e+00 1.57310e+00 1.10111e+0 1.111146e-08 1.111146e-08 1.25667e+00 1.235392e+00 1.93751e+00 1.95550e+00 1.95	34e-09 -3.27636e-09
1.23453e+00       1.23453e+00       3.01487e+00       1.54010e+00       -4.11794e-09       -4.11156e-09       -1.00555e-08       -5.1         1.13560e+00       1.53001e+00       3.22492e+00       1.57640e+00       -5.01351e-09       -1.00555e-08       -5.1         1.135500e+00       1.73143e+00       3.32309e+00       1.5640e+00       -5.01351e-09       -1.00555e-08       -5.3         1.13550e+00       1.92078e+00       1.7544e-09       -5.01351e-09       -1.10813e-08       -5.1         2.08148e+00       2.19728e+00       3.32389e+00       1.82212e+00       -6.42370e-09       -1.11874e-08       -6.6         2.20854e+00       2.19728e+00       3.32803e+00       1.93761e+00       -6.42370e-09       -1.11011e-08       -6.6         2.20854e+00       2.19778e+00       3.22954e+00       1.93761e+00       -7.7344e-09       -1.10111e-08       -6.6         2.30191e+00       3.23539e+00       1.93761e+00       -7.73476e-09       -1.10111e-08       -6.7         2.30191e+00       3.23539e+00       1.93761e+00       -7.73476e-09       -1.07434e-08       -6.7         2.305566+00       2.30191e+00       3.23539e+00       1.99456e+00       -7.14401e-09       -7.13448e-08       -6.7         2.305566+00       2.301756+00 </td <td>42e-09 -4.42005e-09</td>	42e-09 -4.42005e-09
1.50600e+00       1.50301e+00       3.22432e+00       1.67074e+00       -5.01351e-09       -1.0752e-08       -5.5         1.73559e+00       1.73143e+00       3.32309e+00       1.75640e+00       -5.7344e-09       -1.11874e-08       -5.6         1.73559e+00       1.73143e+00       3.33209e+00       1.75640e+00       -5.73370e-09       -5.111874e-09       -5.6         2.081457e+00       2.0971e+00       3.33209e+00       1.88059e+00       -5.73370e-09       -1.11746e-08       -5.6         2.20854e+00       2.19971e+00       3.33507e+00       1.93761e+00       -6.43370e-09       -7.11814e-08       -6.6         2.231250e+00       2.19971e+00       3.32803e+00       1.93761e+00       -7.33746e-09       -1.11746e-08       -6.6         2.331250e+00       2.33537e+00       3.23975e+00       1.93761e+09       -7.3774e-09       -6.7         2.33537e+00       3.23035e+00       2.35176e+00       -7.33746e-09       -1.00743e-08       -6.7         2.357599e+00       2.35176e+00       2.33537ee+09       -7.1340e-09       -1.08749e-08       -7.4         2.35759990       2.33537ee+00       3.13740e+00       2.310771e+00       -8.423737e-09       -1.06751e-08       -7.4         2.55646e+00       2.33537ee+00       2.35336e+	65e-08 -5.13723e-09
1.73569e+00       1.7143e+00       3.32209e+00       1.75640e+00       -5.75640e+00       -5.75640e+00       -5.8055e-09       -5.77544e-09       -1.10813e-08       -5.8         1.92575e+00       1.93761e+00       3.35807e+00       1.882059e+00       -6.43706-09       -6.40370e-09       -1.11766-09       -6.121166       -6.2031266-09       -1.11766-09       -1.11766-08       -6.1211766-08       -6.1211766-09       -6.12117666-09       -6.12117666-09       -6.12172766-09       -7.137266-09       -7.137266-09       -7.137266-09       -7.137286-09       -7.137286-09       -7.1072856-09       -7.1072856-09       -7.1072856-0	72e-08 -5.57301e-09
1.925786+00       1.925786+00       3.535896+00       3.535896+00       1.8252126+00       -6.43570e-09       -6.44377e-09       -1.11876e-08       -6.8         2.086148e+00       2.19712e+00       3.55097e+00       1.93761e+09       -6.943574e-09       -6.138806-99       -6.14887e-09       -1.111766-08       -6.26374e-08       -6.231260e+09       -7.3746e-09       -1.111611e-08       -6.7         2.086148e+00       2.199712e+00       3.23746e+00       1.937616e+00       -7.3574e-09       -7.111676e-08       -6.7         2.131260e+00       2.30191e+00       3.23746e+00       1.937616e+00       -7.332302e+09       -7.13746e-08       -6.7         2.30789e+00       2.30191e+00       3.23746e+00       1.937616e+00       -7.33302e-09       -7.138746e-08       -6.7         2.46820e+00       2.332302e+00       3.23746e+00       2.107714e+00       -7.33302e+09       -7.134898e-08       -6.7         2.55667e+00       3.19730e+00       2.16739e+00       2.16739e+00       -8.72376e-09       -1.07743e-08       -7.1         2.55667e+00       3.19730e+00       2.16739e+00       2.16739e+00       -8.7377e-09       -1.07548e-08       -7.1         2.55667e+00       3.13740e+00       2.16739e+00       2.16739e+00       -8.73302e-09       -1.07548e-08	13e-08 -5.85874e-09
2.08148e+00       2.07423e+00       3.35007e+00       1.88069e+00       -6.34309e-09       -6.131839e-09       -1.11746e-08       -6.2         2.120864e+00       2.30171e+00       3.35307e+00       1.39756e+00       -7.3574e-09       -1.11011e-08       -6.2         2.120864e+00       2.30171e+00       3.23754e+00       1.93756e+00       -7.3754e-09       -1.11011e-08       -6.4         2.312080e+00       2.30171e+00       3.23754e+00       1.99456e+00       -7.3744e-09       -1.08994e-08       -6.6         2.39750e+00       2.38377e+00       3.23939e+00       2.05150e+00       -7.99551e-09       -1.08951e-08       -6.7         2.35566e+00       2.45379e+00       3.23036e+00       2.10771e+00       -8.23333e-09       -8.18438e-09       -1.08743e-08       -7.7         2.55567e+00       3.13740e+00       2.16739e+00       2.21478e+00       -8.42737e-09       -8.3758e-09       -1.06651e-08       -7.1         2.55567e+00       3.13740e+00       2.57444e-09       -8.52815e-09       -1.085518e-08       -7.1         2.55676e+00       2.55747e+00       3.13740e+00       2.57444e-09       -8.57815e-09       -1.085518e-08       -7.1         2.557499e+00       2.55749e+09       3.11048e+00       2.31068e+00       -8.55891e-09	74e-08 -6.07793e-09
Z.20864e+00         Z.30724e+00         1.93761e+00         1.93761e+00         1.11011e-08         -6.4           Z.31260e+00         Z.30191e+00         3.23754e+00         1.99456e+00         -7.71401e-09         -1.010994e-08         -6.4           Z.31260e+00         Z.38537e+00         3.23754e+00         1.99456e+00         -7.71401e-09         -1.00994e-08         -6.6           Z.3537e+00         Z.38537e+00         3.23055e+00         2.10771e+00         -8.23303e-09         -1.08739e-08         -7.6           Z.46820e+00         Z.51011e+00         3.23055e+00         Z.16739e+00         -8.23303e-09         -8.18498e-09         -1.06751e-08         -7.6           Z.57499e+00         Z.51011e+00         3.13730e+00         Z.16739e+00         -8.23303e-09         -8.18498e-09         -1.06651e-08         -7.6           Z.57499e+00         Z.55509e+00         Z.51478e+00         2.51478e+00         -8.53858e-09         -1.06651e-08         -7.6           Z.57499e+00         Z.55530e+00         Z.56434e+00         Z.56434e+00         -8.53891e-09         -1.06552e-08         -7.05556-08         -7.05556-08         -7.05556-08         -7.05556-08         -7.05556-08         -7.05556-08         -7.05556-08         -7.05556-08         -7.055556-08         -7.05556-08         -7.05556	46e-08 -6.27330e-09
2.31260e+00       2.30131e+00       3.23754+00       1.995456+00       -7.71401e-09       -7.67836e-09       -1.09934e-08       -6.6         2.397289e+00       3.36339a+00       3.26339a+00       2.05156e+00       -7.33303e-09       -7.356676e-09       -1.08734e-08       -6.7         2.468278e+00       2.51011e+00       3.19730e+00       2.07714e+00       -8.132303e-09       -8.13734e-08       -7.2         2.55667e+00       3.19730e+00       2.10771e+00       -8.53303e-09       -8.13734e-08       -7.2         2.55567e+00       3.19730e+00       2.16239e+00       -8.53303e-09       -8.13734e-08       -7.2         2.5557499e+00       2.55667e+00       3.13740e+00       2.16239e+00       -8.53230e-09       -1.06551e-08       -7.2         2.55567e+00       3.13740e+00       2.16339e+00       -8.5484e-09       -8.52615e-09       -1.06551e-08       -7.2         2.55574e+00       2.55677e+00       3.11478e+00       2.56434e+00       2.56434e+00       2.67474e+00       -1.045652e-08       -7.1         2.65584e+00       2.56734e+00       2.11046e+00       2.31568e+00       -8.73552e-09       -1.035552e-08       -7.1         2.65784e+00       2.66734e+00       2.31686e+00       2.335352e-09       -8.65721e-08       -7.16720e-08 <td>11e-08 -6.46317e-09</td>	11e-08 -6.46317e-09
2.33789e+00       2.35537e+00       3.26339e+00       2.05150e+00       -7.9851e-09       -1.08873e-08       -6.8         2.46820e+00       2.137071e+00       -8.23303-09       -7.13734-08       -7.6         2.52646e+00       2.51011e+00       3.13730e+00       2.10771e+00       -8.42373-09       -8.13439e-09       -1.07734-08       -7.6         2.552646e+00       2.55607e+00       3.116534e+00       2.1478e+00       -8.42737-09       -8.37285e-09       -1.06651e-08       -7.3         2.55749e+00       2.55607e+00       3.116534e+00       2.12478e+00       -8.47236-09       -8.37285e-09       -1.06651e-08       -7.3         2.61564e+00       2.555607e+00       3.116634e+00       2.26434e+00       -8.72484e-09       -8.65701e-09       -1.06651e-08       -7.3         2.61564e+00       2.55747e+00       3.11048e+00       2.31068e+00       -8.353891e-09       -8.7652e-08       -7.3         2.64384e+00       2.657747e+00       3.11048e+00       2.31068e+00       -8.353891e-09       -1.033555e-08       -7.3         2.64384e+00       2.657747e+00       3.11048e+00       2.300560-09       -1.033555e-08       -7.3         2.77376e+00       2.657747e+00       3.10648e+00       2.300560-09       -1.033555e-08       -7.3 <td>94e-08 -6.65315e-09</td>	94e-08 -6.65315e-09
2.46820e+00     2.145379e+00     3.23005e+00     2.10771e+00     -8.23303e-09     -8.18498e-09     -1.07743e-08     -7.6       2.52546e+00     2.516011e+00     3.19730e+00     2.16239e+00     -8.42737e-09     -8.3758e-09     -1.066518e-08     -7.1       2.55567e+00     2.55667e+00     3.11654e+00     2.121788e+00     -8.55856e-09     -1.065518e-08     -7.1       2.55549e+00     2.55567e+00     3.13740e+00     2.2444e+00     -8.58391e-09     -8.5701e-09     -1.046521e-08     -7.1       2.615549e+00     2.555784e+00     2.57434e+00     2.7444e+09     -8.72484e+09     -8.76424e-09     -1.035558e-08     -7.1       2.649384e+00     2.657747e+00     3.11048e+00     2.31068e+00     -8.353891e-09     -8.76429e-09     -1.03755e-08     -7.1       2.649384e+00     2.657747e+00     3.11048e+00     2.31068e+00     -8.353891e-09     -8.76429e-09     -1.03755e-08     -7.1       2.657374e+00     2.657747e+00     2.56774e+00     2.303755e-00     -8.353891e-09     -1.03755e-08     -7.1       2.77376e+00     2.657747e+00     2.567747e+00     2.567740e     2.933752e-09     -1.02751e-08     -7.1       2.77376e+00     2.567747e+00     2.303752e-09     -8.35352e-09     -1.02751e-08     -7.1	73e-08 -6.84306e-09
2.578646+00 2.51011e+00 3.19730e+00 2.16239e+00 -8.47737e-09 -8.37285e-09 -1.06651e-08 -7.2 2.57499e+00 2.55667e+00 3.16544e+00 2.21478e+00 -8.58926e-09 -8.52185e-09 -1.065618e-08 -7.2 2.615464e+00 2.59530e+00 3.13740e+00 2.26434e+00 -8.83891e-09 -8.6721e-09 -1.04552e-08 -7.1 2.64384e+00 2.62747e+00 3.11048e+00 2.31068e+00 -8.83891e-09 -8.76429e-09 -1.03755e-08 -7.1 2.57874e+00 2.65731e+00 3.01048e+00 2.30558e+00 -8.93532e-09 -8.85391e-09 -1.02521e-08 -7.1	43e-08 -7.03058e-09
2.57499e+00 2.55567e+00 3.1654e+00 2.21478e+00 -8.58926e-09 -8.52811e-09 -1.05638e-08 -7.3 2.61564e+00 2.59530e+00 3.13740e+00 2.26434e+00 -8.72454e-09 -8.65701e-09 -1.046522e-08 -7.5 2.6494e+00 2.62747e+00 3.11048e+00 2.13088e+00 -8.33591e-09 -8.67429e-09 -1.037555e-08 -7.5 2.77874e+00 2.65747e+00 3.08759e+00 2.35586+00 -8.35532e-09 -8.853392e-09 -1.037555e-08 -7.4 2.77876e+00 2.65775c-00 2.65775c-00 3.08709e+00 2.35088e+00 -8.35532e-09 -1.037555e-08 -7.4	51e-08 -7.21295e-09
2.61564e400 2.59530e400 3.13740e400 2.26434e400 -8.72484e-09 -8.65701e-09 -1.04652e-08 -7.5 2.64984e400 2.62747e400 3.11044e400 2.31086e400 -8.83891e-09 -8.7642e-09 -1.03755e-08 -7.7 2.67874e400 2.65431e400 3.08599e400 2.35358e400 -8.93532e-09 -8.85538e409 -1.02221e-08 -7.5 2.73757e400 2.65575e400 3.657726400 3.08554e400 2.35358e400 -8.93552e409 -8.85538e400 4.93555e68 -7.5	18e-08 -7.38773e-09
2.64984e+00 2.62747e+00 3.11048e+00 2.31068e+00 -8.83891e-09 -8.76429e-09 -1.03755e-08 -7.7 2.67874e+00 2.65431e+00 3.08499+00 2.3358e+00 -8.95352e+09 -8.85382e+09 -1.022921e-08 -7.7 2.722556-00 7.55755.00 3.627372.00 7.000 4.03145.00 7.000 4.031462.00 7.000	52e-08 -7.55304e-09
2.67874-400 2.654312+00 3.085494-00 2.53586+00 -8.955322-09 -8.85388-09 1.022912-08 -7.8 7.202056.00 2.55575.00 3.082349-400 3.03549-00 1.022912-08 -7.8	55e-08 -7.70761e-09
	21e-08 -7.85069e-09
(1)	46e-08 -7.98197e-09
2.72413e+00 2.6953e+00 3.04065e+00 2.42876e+00 -9.08674e-09 -8.99133e-09 -1.01425e-08 -8.j	Z5e-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 -8.2	52e-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 -8.5	22e-08 -8.30628e-09
2.77026e+00 2.73330e+00 2.98382e+00 2.51603e+00 -9.24061e-09 -9.12398e-09 -9.5296e-09 -8.3	96e-09 -8.39259e-09

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

Name	Description
$\mathrm{tv}$	time-vector of current (vector 1:m)
$\operatorname{ta}$	number of time-steps of current (scalar)
cv	channel-vector (vector 1:n)
$\mathbf{ca}$	number of channel segments and time-steps of field (scalar)
tend	endtime of calculation in s (scalar)
$\operatorname{Rp}$	distance of fieldpoint in m (scalar)
$\mathbf{v}\mathbf{f}$	upward-propagating return-front-wave speed in m/s (scalar)
$\operatorname{rsmt}$	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)
$\mathbf{E}$	electrical field in $V/m$ (vector 1:n)
Ee	electrostatic field component (vector 1:n)
$\operatorname{Ei}$	electric induction field component (vector 1:n)
$\mathbf{Er}$	electric radiation field component (vector 1:n)
Ed	electric discontinuity field component (vector 1:n)
В	magnetic field in T (vector 1:n)
Bi	magnetic induction field component (vector 1:n)
$\operatorname{Br}$	magnetical radiation field component (vector 1:n)
$\operatorname{Bd}$	magnetic discontinuity field component (vector 1:n)
$\mathrm{tf}$	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:

 $[\text{tend}, \text{ta}, \text{vf}, \text{Rp}, \text{typ}, \text{I0}, \text{T1}, \text{T2}, \text{n}, \text{la}, \text{taub}, \text{tauc}] = \text{enter_file}()$ 

#### 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:

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 (IOt,IOtb,IOtc,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

$$\mathbf{t}_{\mathbf{C}} = \frac{\mathbf{z}' - H}{c} \tag{3.4}$$

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

$$\mathbf{z}^{\prime 2} \cdot \left[ \left( \frac{c}{v_f} \right)^2 - 1 \right] - \mathbf{z}^{\prime} \cdot 2 \frac{c}{v_f} (c \mathbf{t} + \sqrt{R_p^2 + H^2} + H \cdot \frac{c}{v_f}) \\ + \left[ (c \mathbf{t} + \sqrt{R_p^2 + H^2} + H \cdot \frac{c}{v_f})^2 - R_p^2 \right] = 0$$
(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 timevector 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
$\operatorname{tv}$	time-vector of current (vector 1:m)
zvt	height-vector of tall object (vector 1:k)
ZVC	height-vector of channel (vector 1:1)
ZV	merged height-vector (vector 1:n)
tend	endtime of calculation in s (scalar)
nstop	number of reflections calculated
$\operatorname{Rp}$	distance to field-point in m (scalar)
$\mathrm{vf}$	upward-propagating return-front-wave speed in m/s (scalar)
rhotop	reflection coefficient at the top (scalar)
rhobottom	reflection coefficient at the bottom (scalar)
IO	undisturbed current in A (vector 1:m)
$\operatorname{Izt}$	current distribution in A (matrix m:n)
$\mathbf{E}$	electrical field in $V/m$ (vector 1:m)
Ee	electrostatic field component (vector 1:m)
${ m Ei}$	electric induction field component (vector 1:m)
$\mathrm{Er}$	electric radiation field component (vector 1:m)
В	magnetic field in T (vector 1:m)
Bi	magnetic induction field component (vector 1:m)
$\operatorname{Br}$	magnetical radiation field component (vector 1:m)
$\mathrm{tf}$	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:

 $\left[\begin{array}{l} Izt\ ,zv \end{array}\right]\ =\ distribution \left(\begin{array}{l} tv\ ,zvt\ ,zvc\ ,I0\ ,rhotop\ ,rhobot\ ,nstop\ ,\\ vf \end{array}\right)$ 

#### 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  $v_f = 1.3 \cdot 10^8 \frac{\text{m}}{\text{s}}$ . For the MTLE model is  $\lambda = 2000 \text{m}$  and for the DU model is  $\tau_B = \tau_C = 0.1 \mu \text{s}$ .

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  $v_f = 1.3 \cdot 10^8 \frac{\text{m}}{\text{s}}$ . The breakdown discharge time constant is  $\tau_{BD} = 0.6 \mu \text{s}$ and the corona discharge time constant is  $\tau_C = 5 \mu \text{s}$ .

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  $v_f = 1.5 \cdot 10^8 \frac{\text{m}}{\text{s}}$ . 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.







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



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



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



(c) Calculated Magnetic Filed

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



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



(c) Calculated Magnetic Filed

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



(b) Calculated Electrical Field

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



(c) Calculated Magnetic Filed

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



(c) Calculated Magnetic Filed

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



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



(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]



(b) Calculated Current Distribution

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



(c) Calculated Magnetically Field

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



(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

```
1 // v7.1
2
3 // startmodule for calculation of fields for a return stroke
4
5 // written by Andreas F. Dvorak
6 // Vienna University of Technology
7 // Faculty of Electrical Engineering and Information Technology
8 // Institute of Power Systems and Energy Economics
9
10 funcprot(0)
11
12 stacksize('max')
13
14 exec ("start.sci",-1)
15
16 start
```

ground/start.sci

```
1 / / v7.1
 \frac{2}{3} // function to start return stroke model calculation 4
     // written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics
 8
9
10 // functionname start
11
12 // input: none
13
14 // output: none
15
16 function start()
17
         exec ("func_par.sci",-1)
exec ("field.sci",-1)
exec ("func_plot.sci",-1)
exec ("func_save.sci",-1)
18
19
20
21
22
23
24
25
         par=0fie=0
         menu=0
          while menu<>9
    write(%io(2),"
    write(%io(2),"
\frac{26}{27}
                                              ")
")
28
```

```
write(%io(2),"
                                              ")
")
 29
              write(%io(2),")
write(%io(2),"")
write(%io(2),"")
write(%io(2),"")
 30
 31
 32
             33
 34
 35
 \frac{36}{37}
 38
 39
40
             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)," 8 ... interrupt")
write(%io(2)," 8 ... end")
write(%io(2)," ")
write(%io(2)," ")
write(%io(2)," ")
 41
42
 43
 44
 45
 46
 47
48
 \frac{49}{50}
             write(%io(2)," ")
menu=input("Your choice please: ")
write(%io(2)," ")
 51 \\ 52
 53 \\ 54
              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
 55
56
 57
58
                  write (%io(2),"
case 2 then
 59
60
                                                    ")
                     se 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)
 \frac{61}{62}
                     par=1
 63
 64
                      write(%io(2),"
                                                     ")
 65
                  case 3 then
if par==1 then
                         pai==1 then
write(%io(2), "model definition")
write(%io(2), "------")
write(%io(2), " ")
rsmt=input(" model type: 1 for
other
 66
67
 68
 69
                                e(\sqrt{n0}(2), \dots) , ) = input(" model type: 1 for TL, 2 for MTLE, 3 for TCS, 4 for DU, other to break: ")
 70
                         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
transmissiont(" other at the in yes (0 for perc): ")
 71 \\ 72
 73
74
 75
76
77
78
79
80
                            tzoom=input(" zoom at tmax in us (0 for none): ")
write(%io(2)," ")
fn=input(" start figure number. ")
dd
 81
                         [E, Ee, Ei, Er, Ed, B, Bi, Br, Bd, tf]=field (I0t, I0tb, I0tc, tv, tend, cv, vf, rsmt, la
 82
                         83
                         end
fie=1
 84
 85
                     else
                         write(%io(2),"input parameter first!")
 86
                     end
 87
                     write(%io(2)," ")
 88
                  case 4 then
if fie==1 then
 89
 90
 91
                         save_field (E, Ee, Ei, Er, Ed, B, Bi, Br, Bd, tf, tv, vf, rsmt, la, taub, tauc, Rp)
                     else
 92
                          \begin{array}{l} write\,(\,\%io\,(2)\,,"\,\text{calculate one field first"}\,)\\ write\,(\,\%io\,(2)\,,"\,\,\,"\,) \end{array} 
 93
 ^{94}
 95
                     end
                 case 5 then
if par==1 then
 96
97
                         98
 99
100
                                                    Draw figure of fields? 1 for yes, other for no: ")
101
                         fig1 = input("
                         fig2=0
write(%io(2),"
102
                                                      ")
103
104
                          i f
                            f ig1==1
tzoom=input("
                                                         zoom at tmax in us (0 for none): ")
")
105
                             write (%io(2),"
                            input(" 'start figure number: ")
write(%io(2)," ")
106
107
108
```

```
fig2=input(" Draw also single fields with components: 1 for yes,
        other for no: ")
write(%io(2)," ")
d
109
                                                   fig2=input("
110
                                            end
111
                                             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
112
113
114
                                             ,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,
115
                                                          fn+2), end
                                             write (%io(2), "MTLE model:")
write (%io(2), "-----")
116
117
                                             [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,
118
119
                                             tzoom, fn+6), end
write (%io(2), "TCS model:")
write (%io(2), "-----")
120
121
                                             122
123
124
125
126
                                             [\texttt{E\_du},\texttt{Ee},\texttt{Ei},\texttt{Er},\texttt{Ed},\texttt{B\_du},\texttt{Bi},\texttt{Br},\texttt{Bd},\texttt{tf}] = \texttt{field}(\texttt{I0t},\texttt{I0tb},\texttt{I0tc},\texttt{tv},\texttt{tend},\texttt{cv},\texttt{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, and the state of 
127
                                                           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
128
129
                                             fie = 2
130
                                     else
                                             write (\%io(2), "input parameter first!")
131
 132
                                      end
                                      write(%io(2),"
133
                                                                                          ")
 134
                               case 6 then
if fie==2 then
135
                                            save_all_field (E_tl,E_mtle,E_tcs,E_du,B_tl,B_mtle,B_tcs,B_du,tf,tv,vf,
136
                                                           la, taub, tauc, Rp)
137
                                             \begin{array}{l} write\,(\,\%io\,(2)\,, \texttt{"calculate all fields first"}\,) \\ write\,(\,\%io\,(2)\,, \texttt{"}\, \texttt{"}\,) \end{array} 
138
139
140
                                     end
                               case 8 then
write (%io(2),"
141
142
                                                                                             ")
 143
                                       write(%io(2),"
                                                                                              ---> type resume to continue <---")
144
                                      pause
145
                                case 9 then
                                     write(%io(2),"
write(%io(2),"
146
147
                                                                                              !!! Have a good one !!!")
148
                                else
                                      write(%io(2),"
write(%io(2),"
                                                                                              ")
149
                                                                                             No menu entrv !")
150
151
                         \mathbf{end}
                  end
152
153 endfunction
```

### A.1.2 Definition of Parameters

ground/func\_par.sci

```
// v7.1
 2
 \overline{3} // functions for parameter input and calculation (enter_user, enter_file,
            calc_par)
 4
    // written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
 5
 6
 7
 \frac{8}{9}
    // Institute of Power Systems and Energy Economics
10
   // functionname enter_user
11
12
13 // function to define parameter through keyboard
14
   // input: none
15

    16
    17

    // output: tend ... endtime of calculation in s
// ta ... number of timesteps
// vf ... front-wave speed in m/s
// Rp ... distance of fieldpoint in m
18
19
20
```

```
typ ... type of part i current (de: double-exponential, h. heidler)
IO, 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
 ^{21}
 22
23
24
 25
 26
 27
     function [tend, ta, vf, Rp, typ, I0, T1, T2, n, la, taub, tauc] = enter\_user()
 28
 29
          n=0
 30
          write(%io(2)," ")
write(%io(2),"timevector for field definition")
write(%io(2),"-----")
write(%io(2)," ")
tend=input(" Endtime [us]: ")

 31
 32
 33
34
          tend=tend \pm 10^{\circ}(-6)
ta=tend \pm 10^{\circ}(-6)
 35
          ta{=}input(" number of timesteps (best range: 1000 to 5000): ") write (\%io\,(2)\,, " ")
 36
 37
 38
 \frac{39}{40}
           write (%io(2), "channel definition")
 \frac{41}{42}
           write (%io(2), "-----")
write (%io(2), " ")
                              ' upward-propagating return-front-wave speed in 10^8 m/s: ")
           vf=input("
vf=vf*10^8
 43
 44
           write(%io(2),"
                                         ")
 \begin{array}{c} 45 \\ 46 \end{array}
          write(%io(2),"fieldpoint definition")
write(%io(2),"------")
write(%io(2)," ")
Rp=input(" distance of fieldpoint i
 47
 48
 49
                               distance of fieldpoint in km: ")
 50
 51 \\ 52
          Rp=Rp * 10 3
                                        ")
           write(%io(2),"
 53 \\ 54
          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=lo
 55
 56
          a=input(" Hoy
write(%io(2),"
for i=1:a
 57
 58
 59
              b=input(" 1 for Heidler, 2 for Double-Exponetial, 3 Nucci90, 4 DU90 other
 60
             to break: ")
write(%io(2)," ")
 61
 62
 63
              if b>2 then, break, end
 64
 65
              if b==1
                 typ(i)="h"
 66
                 typ(1)="h"
write(%io(2),"
write(%io(2),"
IO(i)=input("
IO(i)=IO(i)*10^3
 67
                                                  Heidler:")
 68
 69
70
71
72
                                                    )
IO [kA]: ")
                 T1(i)=input (" current rise time constant T1 [us]: ")
T1(i)=T1(i)*10^{\circ}(-6)
                 T1(1)=11(1)+10("-0)
T2(i)=input(" current decay time constant T2 [us]: ")
T2(i)=T2(i)*10^(-6)
n(i)=input(" n []: ")
write(%io(2)," ")
 73
74
 75
76
 77
78
79
              \mathbf{end}
              if b==2
 80
                 \operatorname{typ}\left( {{\,\mathrm{i}\,}} \right){\rm{ = "de "}}
                 typ(1)="de"
write(%io(2),"
write(%io(2),"
I0(i)=input("
I0(i)=I0(i)*10^3
 81
                                                  double exponential:")
 82
                                                    ÍO [kA]: ")
 83
 84
 85
                 T1(i) = input (" c
T1(i) = T1(i) * 10^{(-6)}
                                                    current decay time constant T1 [us]: ")
 86
 87
                 T2(i)=input(" c
T2(i)=T2(i)*10^(-6)
                                                    current rise time constant T2 [us]: ")
 88
                  write(%io(2),"
 89
 90
              end
 91
           end
 92
          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)
 93
 94
 95
 96
97
 98
99
              n(1)=2
typ(2)="de"
              IO(2) = -7.5*10^{3}
T1(2) = 100*10^{(-6)}
100
101
102
              T2(2) = 6 * 10^{(-6)}
           end
103
```

```
104
          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
105
 106
107
108
109
110
111
112
113
114
              n(2) = 2
115
116
           end
117
           118
119
120
121
          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)}
122
123
124
125
          126
                                      discharge time constant for corona current in us for DU: ")
127
128
           write(%io(2),"
129
130 endfunction
131
132 // functionname enter_file
133
134 // function to define parameter through file
135
136 // input: none
137
137
138 // output: tend ... endtime of calculation in s
139 // ta ... number of timesteps
140 // vf ... front-wave speed in m/s
141 // Rp ... distance of fieldpoint in m
142 // typ ... type of part i current (de: double-exponential, h: heidler)
143 // I0, T1, T2, n ... parameter for part i of current according to typ(i)
144 // la ... coefficient for MTLE model in m
145 // taub ... breakdown coefficient for DU model in s
146 // tauc ... corona coefficient for DU model in s
147
147
148
 149
      function [tend, ta, vf, Rp, typ, I0, T1, T2, n, la, taub, tauc] = enter_file()
150
           write(%io(2)," ")
151
           write(%io(2), "read from file")
write(%io(2), "read from file")
write(%io(2), "-------")
write(%io(2), " ")
rd=input(" read ff.....
152
153
154
           rd=input("
                                  read definition from file def_ground.txt? 1 for yes, other for no:
155
156
           write(%io(2),"
                                          ")
157
158
           if rd==1 then
159
               errcatch (999, "continue")
               fid=mon("def_ground.txt","r")
if fid==-1 then
write(%io(2)," error: cannot
160
161
162
                                                  error: cannot open file def.txt!")
               else
163
                  header=1
164
165
                  while header = = 1
                      val=mfscanf(1,fid,"%s")
if val="start_of_def" then, header=0, end
166
167
168
                  end
 169
                  data=1
170
                  while data==1
                      val=mfscanf(1,fid,"%s")
 171
                      val=miscanf(1,11d, "%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
pact=1
172
173 \\ 174
 175
176
 177
178
                            next=1
179
                              i = 1
                              while next==1
180
                                 [dummy,typ(i),I0(i),T1(i),T2(i),n(i)]=mfscanf(1,fid,"%s %f %f %f %f %f %f
181
                                 if typ(i)=="x" then
next=0
182
183
                                     typ=typ(1:(i-1))
I0=I0(1:(i-1))
184
185
```

```
T1=T1(1:(i-1))
186
187
                                               T2=T2(1:(i-1))
n=n(1:(i-1))
 188
                                           else
189
                                               I0(i)=I0(i)*10^3
 190
                                               T1(i)=T1(i)*10^{-6}
T2(i)=T2(i)*10^{-6}
191
 192
193
                                                i = i + 1
 194
                                           end
195
                                      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,
 196
197
 198
199
200
                             end
                       end
201
202
                   end
                   mclose (fid)
203
             \operatorname{errcatch}(-1)
end
204
205
206 \\ 207
        endfunction
208
209
210
211 // functionname calc_par
212
213 // function for to calculate vectors and input-current
214
        // input: tend ... endtime of calculation in s
215
                                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)
216
217
218
219
220
221

      221

      222
      // output: tv ... timevector

      223
      // cv ... channelvec

      224
      // IOt=I(0,t) ... cu

      225
      // IOtb=Ib(0,t) ... cu

      202
      // LOtb=Ib(0,t) ... cu

                                   tv ... therefore

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
226
227
228
229
        function [tv, cv, I0t, I0tb, I0tc]=calc_par(tend, ta, vf, Rp, typ, I0, T1, T2, n)
230
231
              exec ("func_rsc.sci",-1)
232
233
              c\!=\!299792458 // speed of light
234
              write(%io(2), "timevector calculation")
write(%io(2), "-----")
write(%io(2), " ")
235
236
              write(%10(2)," ")
ts=tend/ta // timestep
tv=0:ts:tend // build timevector
write(%10(2)," --> length of timestep in us:")
write(%10(2),(ts/10^(-6)),"(7X,F10.5)")
write(%10(2)," ")
237
238
239
240
241
242
243
              \begin{array}{l} \mbox{write} (\%io(2), "channelvector calculation") \\ \mbox{write} (\%io(2), "-----------") \\ \mbox{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=lenth(cv) \end{array} 
244
245
246
247
248
249
250
              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)," ")
251
252
 253
254
 255
256
257
              \label{eq: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
258
259
260
261
              a=length(I0)
              I0t = zeros(tv)
262
263
              for i=1:a
264
265
                   if typ(i)=="h"
266
267
                       I(i,:)=heidler(I0(i),T1(i),T2(i),n(i),tv) // part i of current
```

```
268
269
         end
270
         if typ(i)=="de"
           271
272
                  figur
273
         end
274 \\ 275
         IOt=IOt+I(i,:) // calculate current at origin
       end
276
277
       if a==2
         IOtb=I(1,:) // breakdown current for DU model
IOtc=I(2,:) // corona current for DU model
278
279
      else
I0tb=0
280
281
282
         {\rm I}\,0\,{\rm t}\,{\rm c}=0
       end
283
284
       write(%io(2),"figure of current")
write(%io(2),"-----")
write(%io(2)," ")

285
286
287
       288
289
290
291
       if fig==1
        tzoom=input(" zoom at tmax in us (0 for none): ")
write(%io(2)," ")
fn=input(" start figure number: ")
292
293
                          start figure number: ")
")
294
         fn=input (
295
         write(%io(2),"
296
297
         f1=scf(fn) // seclect figure
clf(f1) // clear if exist
tmax=find(tv>=tend,1)
298
299
300
         if a > 1
           301
             plot2d((tv(1:tmax)*10^6),(I(i,1:tmax)*10^(-3)),style=(i+1)) // plot part
302
                  i of current
303
           end
         end
304
         plot2d ((tv(1:tmax)*10^6),(I0t(1:tmax)*10^(-3)), style=1) // plot complete
305
               current
         f1. figure_size = [1500 \ 1000]
306
307
         f1.children.grid=[1 1]
         select a case 1, h1=legend (txt1(1), 4),
308
309
           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),"$current \ at \ origin$"
310
311
312
                ,4<sup>)</sup>,
313
         end
         xtitle("","$t \; / \; \mu s$","$I \; / \; kA$")
314
315
         316
317
318
           clf(f2) // clear if exist
tzoom=tzoom*10^(-6)
319
320
           tzoom=find(tv>=tzoom,1) // find next index according to zoom given
           if a > 1
for i=1:a
321
322
               plot2d ((tv(1:tzoom)*10^6),(I(i,1:tzoom)*10^(-3)), style=(i+1)) // plot
323
                     part i
                            of zoomed curr
324
              \mathbf{end}
325
           end
326
           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]
327
328
           329
330
331
332
333
334
335
336
         end
337
338
         fig=input(" Draw figures of current derivative? 1 for yes, other for no: ") write (<math display="inline">\%io\,(2) ," ")
339
340
         if fig==1
341
```

```
342
343
344
345
           f3=scf((f1.figure_id+2)) // select figure
346
           clf(f3) // clear if exist
plot2d((tv(1:tmax)*10^6),(dI0t(1:tmax)*10^(-9)),style=1) // plot complete
347
348
349
           f3.figure_size = [1500 1000]
           f3.children.grid=[1 1]
xtitle("","$t \ \; / \; \mu s$","$\frac{\mathrm{d}I}{\mathrm{d}t} \; / \; \
frac{kA}{\mu s}$")
350
351
352
           if tzoom>0
353
354
             f4=scf((f1.figure_id+3)) // select figure
             clf(f4) // clear if exist
tzoom=tzoom*10^(-6)
355
356
             tzoom=tzoom=t0 (-0)
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
357
358
                   current derivative
             359
360
361
362
          end
        end
363
364
      end
365
366 endfunction
```

### ground/func\_rsc.sci

```
1 / / v7.1
 2
 \overline{\mathbf{3}} // 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
 5
 6
 8
9
10
11
12
    // functionname heidler
13
14 \\ 15
    // function to calculate current at channel origin
// as a heidler function
\frac{16}{17}
17 // input: I0, T1, T2, n (scalar)
18 // tv ... timevector (vector 1:m)
19
20 // output: I0t=I(0,t) ... current at channel origin (vector 1:m) 21
22 function IOt=heidler(IO,T1,T2,n,tv)
23
          \begin{array}{l} 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) \end{array}
24
25
26
27
28
    endfunction
29
\begin{array}{c} 30\\ 31 \end{array}
32 // functionname dexp
33
    // function to calculate current at channel origin // as a double-exponential function
34
35
36
    \begin{array}{cccc} // & input: \ I0 \ , \ T1 \ , \ T2 \ (scalar \ ) \\ // & tv \ \dots \ timevector \ (vector \ 1:m) \end{array}
37
38
39
    // output: I0\!t\!=\!I\left(0\,,t\,\right) ... current at channel origin (vector 1:m)
40
41
    function I0t=dexp(I0, T1, T2, tv)
42
43
          I0t = I0 * (exp(-tv/T1) - exp(-tv/T2))
44
45
46 endfunction
```

### A.1.3 Calculation of fields

### ground/field.sci

```
1 // v7.1
 2
 3 // module to calculate the field and its components
 4

4
5 // written by Andreas F. Dvorak
6 // Vienna University of Technology
7 // Faculty of Electrical Engineering and Information Technology
8 // Institute of Power Systems and Energy Economics

 9
10
    // functionname field
11
    // 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
// current for calculation
12
13
14
15
16
                       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
17
18
19
20
21
22
23
24
25
    // output: 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
\frac{26}{27}
28
29
30
31
32 function [E, Ee, Ei, Er, Ed, B, Bi, Br, Bd, tf]=field(IOt, IOtb, IOtc, tv, tend, cv, vf, rsmt, la, taub, tauc, Rp)
33
           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
34
35
36
37
38
39
           c\!=\!299792458 // speed of light
40
           tf=Rp/c // starttime of field at fieldpoint
41
42
           write(%io(2)," ")
43
44
45
           timer()
46
47
           write\,(\,\%io\,(2)\,," ... calculating radiation component") write\,(\,\%io\,(2)\,," ")
48
           [Er,Br]=r_field (I0t,I0tb,I0tc,tv,cv,vf,rsmt,la,taub,tauc,Rp) // calculate
49
                       adiation component
50
            write(%io(2),timer())
write(%io(2),"")
timer()
51 \\ 52
53 \\ 54
           write ( \%io\left(2\right) , " ... calculating induction component " ) write ( \%io\left(2\right) , " " )
55
56
           [Ei,Bi]=i_field (I0t,I0tb,I0tc,tv,cv,vf,rsmt,la,taub,tauc,Rp) // calculate
57
                    induction component
58
              write(%io(2),timer())
write(%io(2),"")
timer()
59
60
61
62
63
           64
65
                    electrostatic component)
66
               write(%io(2),timer())
write(%io(2),"")
67
68
69
70
71
72
73
74
75
              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
76
77
                 Ed=0
                Bd=0
78
           end
```

79
80 // write(%io(2),timer())
81 write(%io(2)," ")
82
83 E=Ee+Ei+Er+Ed
84 B=Bi+Br+Bd
85
86 endfunction

#### ground/e\_field.sci

```
1 / / v7.1
  2
  3 // function to calculate the electrostatic component of the field
   4
                                                 with perfectly conducting ground in height z=0
  5
        // by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics
  \frac{6}{7}
   8
10
       // functionname e_field
11 \\ 12
        // input: I0t=I(0,t) ... current at channel origin (vector 1:k)
// I0tb=Ib(0,t) ... breakdown current at channel origin for DU model (
 13
14
                                               1:k)
15 //
                                         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)
16
 17
18
 19
20
21
22
23
24
25
26
        // output: Ee(t) ... electric induction component
27
28
        function Ee=e_field (IOt, IOtb, IOtc, tv, cv, vf, rsmt, la, taub, tauc, Rp)
                 29
30
31
32
33
34
35
36
                    efac=(2*cv.^2-Rp^2)./(R.^5*(2*%pi*eps0)) // factor electrostatic component
37
38
                   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
39
40
                                               at starttime
                                     1/2
41
                             e // 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
                    else
42
43
                            at starttime
dIotbe=(Iotb(ta)-IOtb(ta-1))/tv(2) // derivative of current IOtb an the end
dIotb=splin(tv,IOtb,"clamped",[0 dIOtbe]) // derivatives of breakdown
current IOtb with 0 at starttime
dIotce=(IOtc(ta)-IOtc(ta-1))/tv(2) // derivative of current IOtc an the end
dIotc=splin(tv,IOtc,"clamped",[0 dIOtce]) // derivatives of corona current
IOtc with 0 at starttime
44
45
46
47
48
                   end
49
                    dFe=zeros(ca,ca) \ // \ build \ inital \ field \ matrix \\ Ee=zeros(1,ca) \ // \ build \ inital \ vector \ of \ Ee \ electrostatic \ component \ of \ E \ electrostatic \ component \ of \ electrostatic \ electros
50
51
 52
53
                    for j=1:ca
54 \\ 55
                    select rsmt
56
57
                             58
59
60
61
62
                                               dFe(j,i) = dFe(j,(i-1)) + tv(2) * (Izt(i-1) + Izt(i)) / 2 // calculate Q
63
                                       end
64 \\ 65
                             66
```

```
index=find (Izt <>0,1)
 67
                         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
 68
 69
70
71
72
73
74
75
76
77
                         \mathbf{end}
                   case 3 then //\ {\rm TCS} model
                         ts tart = cv (j)/vf+cv (j)/c // start time 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
 78
79
                         Izt = [zeros(1, (j-1)) Izt(1:(ca-j+1))]
index=find(Izt <>0,1)
                         if index == [], break, end
for i=index:ca
 80
 81
                         for
                             dFe(j,i)=dFe(j,(i-1))+tv(2)*(Izt(i-1)+Izt(i))/2 // calculate Q
 82
                         end
 83
 84
85
                   case 4 then // DU model
                        se 4 then // DO 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
 86
87
 88
 89
 90
                         dtb=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
 91
 92
 93
 94
                               spline interpolation
dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
 95
                              dib=exp(-tv(l:length(ts))/taub) // decrease factor breakdown
Iztc=interp(ts,tv,I0tc,dI0tc,"by_zero") // calculate current as
    spline interpolation
dfc=exp(-tv(l:length(ts))/tauc) // decrease factor corona
Izt=Iztb-Iztb(l).*dfb+Iztc-Iztc(l).*dfc
 96
 97
 98
 99
                         \mathbf{end}
                         Izt = [zeros(1, (j-1)) Izt(1:(ca-j+1))]
index=find(Izt <>0,1)
100
101
                           f index == [], break, end
102
103
                         for
                                i=index:ca
104
                             dFe(j,i)=dFe(j,(i-1))+tv(2)*(Izt(i-1)+Izt(i))/2 // calculate Q
105
                         end
106
                   \mathbf{end}
             end
107
108
             for i = 1:ca
109
110
                  Ee(i) = inttrap(cv(1:i), (dFe(1:i,i)'.*efac(1:i))) // calculate field
                            integrate over the channel
111
             end
112
113 endfunction
                                                                     ground/i_field.sci
   1 / / v7.1
      // function to calculate the induction components of the field // with perfectly conducting ground in height z{=}0
  3
   5
   6
      // by Andreas F. Dvorak
            Vienna University of Technology
Faculty of Electrical Engineering and Information Technology
Institute of Power Systems and Energy Economics
   7
   \frac{8}{9}
  10
 11
      // functionname i_field
  12
      // input: I0t=I(0,t) ... current at channel origin (vector 1:k)
// I0tb=Ib(0,t) ... breakdown current at channel origin for DU model (
 13
 14
                             1:k)
 15
                          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)
 16 \\ 17
 18
19
                          .... apward-propagating return-front-wave speed (s
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)
 20
21
 \frac{22}{23}
 24
```

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

```
27
28
      function [Ei, Bi]=i_field (I0t, I0tb, I0tc, tv, cv, vf, rsmt, la, taub, tauc, Rp)
29
            30
31
32
33
34
35
36
37
             \texttt{efac} = (2*\texttt{cv}.^2 - \texttt{Rp}^2) \, . \, / \, (\,\texttt{c}*\texttt{R}.^4*(2*\%\texttt{pi}*\texttt{eps}0)\,) \ / / \ \texttt{factor} \ \texttt{electric} \ \texttt{induction}
38
39
             bfac=(mu0*Rp)./(2*%pi*R.^3) // factor magnetic induction component
                     \begin{array}{l} (rsmt<>4) \mid (10tb==zeros\,(10tb\,)) \ // \ 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 \ dI0te] \ . \end{array} 
40
41
42
                                at starttime
                    e // 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
43
44
45
                   at startime
dIotbe=(Iotb(ta)-Iotb(ta)/tv(2) // derivative of current IOtb an the end
dIotb=splin(tv,IOtb,"clamped",[0 dIOtbe]) // derivatives of breakdown
current IOtb with 0 at starttime
dIOtce=(IOtc(ta)-IOtc(ta-1))/tv(2) // derivative of current IOtc an the end
dIOtc=splin(tv,IOtc,"clamped",[0 dIOtce]) // derivatives of corona current
IOtc with 0 at starttime
\begin{array}{c} 46 \\ 47 \end{array}
48
49
50
             end
51
             dFi{=}zeros\,(ca\,,ca) // build inital field matrix Ei{=}zeros\,(1\,,ca) // build inital line vector of Ei electric induction component
52
\overline{53}
                       of E
             Bi=zeros(1,ca) // build inital line vector of Bi magnetic induction component
54
                       of E
55
56
             for j = 1:ca
57
                    select rsmt
58
59
                   \begin{array}{c} {\tt case \ 1 \ then \ // \ TL \ model} \\ {\tt Izt \!=\! [\tt zeros \, (1\,, (j-1)) \ I0t \ (1\!:\! (ca\!-\!j\!+\!1)) \ ]} \end{array}
60
61
62
                          \mathrm{d}\,\mathrm{F}\,\mathrm{i}\,(\,\mathrm{j}\,\,,\,:\,)\,{=}\,\mathrm{I}\,\mathrm{z}\,\mathrm{t}
63
                    case 2 then // MTLE model
64
                          Izt=I0t*exp(-cv(j)/la)
Izt=[zeros(1,(j-1)) I0t(1:(ca-j+1))*exp(-cv(j)/la)]
65
66
67
                          d \operatorname{Fi}(j,:) = I z t
                   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
68
69
70
71
72
73
                          Izt = [zeros(1, j-1) Izt(1:(ca-j+1))]
dFi(j,:)=Izt
\frac{74}{75}
76
77
78
                   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
79
80
81
                                 Ist=interp(ts,tv,IOt,dIOt,"by_zero") // calculate current as spline
interpolation by_zero: if out-of-bound finish extrapolation with
82
                                 dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
83
84
                                 Izt=Izt-Izt(1).*dfb
                          else
85
                                Iztb=interp(ts,tv,I0tb,dI0tb,"by_zero") // calculate current as
    spline interpolation by_zero: if out-of-bound finish
    extrapolation with 0
86
                                 dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
87
                                 Iztc=interp(ts,tv,IOtc,dIOtc,"by_zero") // calculate current as
spline interpolation by_zero: if out-of-bound finish
88
                                           extrapolation with 0
89
90
                                dfc=exp(-tv(1:length(ts))/tauc) // decrease factor corona
Izt=Iztb-Iztb(1).*dfb+Iztc-Iztc(1).*dfc
91
92
                           end
                          Izt = [zeros(1, j-1) Izt(1:(ca-j+1))]
93
                          d \operatorname{Fi}(j,:) = I z t
                   end
94
             end
95
96
```

### ground/r\_field.sci

```
1 // v7.1
  2
  3 // function to calculate the radiation components of the field 4 // $ with perfectly conducting ground in height z=0 $
   5
  6
       // by Andreas F. Dvorak
          // Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
   8
        // Institute of Power Systems and Energy Economics
  9
 10
11
       // functionname r_field
 12
13 // input: I0t=I(0,t) ... current at channel origin (vector 1:k)
14 // I0tb=Ib(0,t) ... breakdown current at channel origin for DU model (
                                               1 \cdot \mathbf{k}
15
                                        IOtc=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)
tauh breakdown coefficient for DU model (scalar)
16
17
18
19
20
21
22
23
                                         taub ... breakdown coefficient for DU model (scalar)
taub ... corona coefficient for DU model (scalar)
Rp ... distance of fieldpoint (scalar)
24
25
        // output: Er(t) ... electric radiation component (vector 1:n) // Br(t) ... magnetical radiation component (vector 1:n)
26
27
28
29
        function [Er,Br]=r_field(I0t,I0tb,I0tc,tv,cv,vf,rsmt,la,taub,tauc,Rp)
                  30
31
32
33
34
35
36
37
                    efac=(-1)*(Rp^2)./(c^2*R.^3*(2*%pi*eps0)) // factor electric radiation
38
                    bfac=(mu0*Rp)./(2*%pi*c*R.^2) // factor magnetic radiaion component
39
                   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
40
41
42
43
                             e \ // \ 2 \ currents form DU model \\ dI0te=(I0t(ta)-I0t(ta-1))/tv(2) \ // \ derivative of current I0t and the end \\ dI0t=splin(tv,I0t,"clamped",[0 \ dI0te]) \ // \ derivatives of current I0t with 0 \ dI0t=0 
                    else
44
45
                            at starttime
dIOtbe=(IOtb(ta)-IOtb(ta-1))/tv(2) // derivative of current IOtb an the end
dIOtb=splin(tv,IOtb,"clamped",[0 dIOtbe]) // derivatives of breakdown
current IOtb with 0 at starttime
dIOtce=(IOtc(ta)-IOtc(ta-1))/tv(2) // derivative of current IOtc an the end
dIOtc=splin(tv,IOtc,"clamped",[0 dIOtce]) // derivatives of corona current
IOtc with 0 at starttime
 46
47
48
49
50
                    \mathbf{end}
51
                   52
53
54 \\ 55
56
57
                    for j = 1: ca
58
59
                    select rsmt
                            case 1 then // TL model
dFr(j,:)=[zeros(1,j-1) dIOt(1:(ca-j+1))]
60
61
62
                             case 2 then // MTLE model
63
                                       \begin{array}{l} direct = 0 & direct + j \\ direct = 0 & t \\ direct = (Izt (ta) - Izt (ta - 1)) / tv (2) \\ direct = splin (tv, Izt, "clamped", [0 & dIzte]) \end{array} 
64 \\ 65
66
```

67	dFr(j,:) = [zeros(1,j-1) dIzt(1:(ca-j+1))]
68	anso 2 than // TCS model
70	$t_{start-cy(i)/yf_{cy(i)/c}}$ starttime of timestep-vector
71	$t_{sstat} = c_{sstat} + (c_{ss} + 1) + t_{ss} (2) / (c_{sstat} + 1) + t_{sstat} (2) / (c_{sstat} + 1) + t_$
72	$t_{s}$ = t_{s} t_{s} t_{t} = t_{s} t_{s} = t_{s} t_{s} = t_{s} t_{s} = t_{s} t_{s} = t_{s}
73	Izt = interpolation (2). is and // timestep = vector for interpolation
10	interpolation by zero; if out-of-bound finite extrapolation with (
74	dizt=splin(ts_lzt "monotone") // derivative of current in channel
75	$dFr(i, \cdot) = [zeros(1, i-1), d[zt(1:(za-i+1))]]$
76	
77	case 4 then // DU model
78	$t_{i}$ the table $t_{i}$ and the tabular and tabular
79	tsend=tsstart+(ca-i+1)*ty(2) // endtime of timestep-vector
80	ts = ts tart : ty(2) : ts end //max(ty) // timestep - vector for interpolation
81	if I0th=zeros(I0th) then
82	Izt=interp(ts.tv,I0t,dI0t,"by zero") // calculate current as spline
	interpolation by zero; if out-of-bound finish extrapolation wit
	0
83	dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
84	$Izt = Izt - Izt(1) \cdot * dfb$
85	else
86	Iztb=interp(ts,tv,I0tb,dI0tb,"by_zero") // calculate current as
	spline interpolation $by_zero:$ if $out-of-bound$ finish
	extrapolation with 0
87	dfb=exp(-tv(1:length(ts))/taub) // decrease factor breakdown
88	<pre>Iztc=interp(ts,tv,I0tc,dI0tc,"by_zero") // calculate current as</pre>
	spline interpolation $by_zero:$ if $out-of-bound$ finish
	extrapolation with 0
89	dfc=exp(-tv(1:length(ts))/tauc) // decrease factor corona
90	$Izt = Iztb - Iztb(1) \cdot * dfb + Iztc - Iztc(1) \cdot * dfc$
91	end
92	tsl=length(ts)
93	dIzte = (Izt(tsl) - Izt(tsl - 1)) / tv(2)
94	dIzt = splin(ts, Izt, "clamped", [0 dIzte])
95	dFr(j,:) = [zeros(1,j-1)  dIzt(1:(ca-j+1))]
96	end
97	end
98	
99	
100	for $i=2:ca$
101	$\operatorname{Er}(1) = \operatorname{inttrap}(\operatorname{cv}(1:1), (\operatorname{dFr}(1:1,1)'.*\operatorname{etac}(1:1))) // \operatorname{calculate} field$
100	integrate over the channel
102	Dr(1) = interap(CV(1:1), (drr(1:1,1)), * biac(1:1)))
103	ena;
104	andfunction
1 1 1	

#### ground/d\_field.sci

```
\begin{array}{cccc} 1 & // & v7.1 \\ 2 & \end{array}
 ^2 3 // function to calculate the discontinuity components of the field 4 // with perfectly conducting ground in height z=0
 \overline{5}
 5
6 // by Andreas F. Dvorak
7 // Vienna University of Technology
8 // Faculty of Electrical Engineering and Information Technology
9 // Institute of Power Systems and Energy Economics
10
11 \\ 12
     // 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)
13
14 \\ 15 \\ 16 \\ 17 \\ 18
     // output: Ed(t) ... electric discontinuity component (vector 1:n) // Bd(t) ... magnetic discontinuity component (vector 1:n)
19
20
21
22
23
24
25
26
     function [Ed,Bd] = d_field (I0t, tv, cv, vf, Rp)
           27
28
\frac{29}{30}
            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
31
```

 $dH\!\!=\!\!splin\left(tv\left(1\!:\!ca\right),cv\right) \; // \; \text{derivative of high of discontinuity with respect to} \;$ 32 tim 33 efac=-(Rp^2)./(c^2\*R.^3\*(2\*%pi\*eps0)) // factor electric discontinuity 34bfac=(mu0\*Rp)./(2\*%pi\*c\*R.^2) // factor magnetic discontinuity component 35 36 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 37 38 39  $\label{eq: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$ 40 414243Bd=bfac.\*Iht.\*dH 44 45 endfunction

### A.1.4 Plotting and saving of data

ground/func\_save.sci

```
// v7.1
    \mathbf{1}
   2
   \frac{1}{3} // functions to save the field (save_field, save_all_field)
    4
       // written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics
   6
    8
  10
  11
  12 // functionname save_field
  13
14 // functions to save field and its component
15
16 // input: E=E(t) ... electrical field in V/m
17 // Ee, Ei, Er, Ed ... components of electrical field in V/m
18 // B=B(t) ... magnetic field in T
19 // Bi, Br, Bd ... components of magnetic field in T
20 // tf ... time of arrival of field at field point in s
21 // tv ... timevector
22 // vf ... upward-propagating return-front-wave speed in m/s
23 // rsmt ... return-stroke-model type (1: TL, 2: MTLE, 3: TCS, 4: DU)
24 // la ... coefficient for MTLE model in m
25 // taub ... breakdown coefficient for DU model in s
26 // Rp ... distance of fieldpoint in m
28
  14 // functions to save field and its components
 29 // output: none
  30
       \texttt{function} \hspace{0.2cm} [] = \texttt{save_field} \hspace{0.2cm} (\texttt{E}, \texttt{Ee}, \texttt{Ei}, \texttt{Er}, \texttt{Ed}, \texttt{B}, \texttt{Bi}, \texttt{Br}, \texttt{Bd}, \texttt{tf}, \texttt{tv}, \texttt{vf}, \texttt{rsmt}, \texttt{la}, \texttt{taub}, \texttt{tauc}, \texttt{Rp})
  31
  32
                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=""
 33
             select rsmt
  34
 35
  36
 37
 38
  39
             end
  40
  41
             if length(Ed) < length(E) then
  42
              Ed=zeros(E)
Bd=zeros(E)
  43
  44
             end
 45
46
             47
48
 sav=input(" save field and components to file? 1 for yes, other for no: ") write (\%io(2)," ")
             51 \\ 52
 53 \\ 54
                 fnw=input(" filename:
fid=mopen(fnw, "w")
 55
56
                 if fid == -1 then
                      write (%io(2),"
                                                              Error: Cannot open file for writing!")
 57
58
                  else
                                                             ... writing data to file: "+fnw) ")
                     write(%io(2),"
write(%io(2),"
  59
```

```
mfprintf(fid, "%s \n",txt)
mfprintf(fid, "%s \3.2f %s \n", "field and components at fieldpoint: ",(Rp
 *10^(-3)), "&m")
mfprintf(fid, "%s \t \%s \t %s 
       60
      61
       62
       63
         64
       65
       66
                                                                                        mfprintf(fid , "%s \n", "Data:")
for i=1:length(E)
       67
       68
                                                                                                        mfprintf(fid, "%1.5e \t %1.5e \t %
       69
                                                                                                                                                  i),Bd(i))
      70
                                                                                      end
      71
72
                                                                                      mclose(fid)
                                                                     end
                                                 \operatorname{errcatch}(-1) end
      73 \\ 74
      75
76 endfunction
      77
78
       79
       80 // functionname save_all_field
       81
       82
                             // function to save fields of all models
       83
                          // 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
// tr ...
       84
       85
       86
       87
       88
                                                                                                                       v. ... upward-propagating return-front-wave speed
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
       89
       90
       91
       92
       93
       94
                             // output: none
       95
                           function [] = save_all_field (E_tl, E_mtle, E_tcs, E_du, B_tl, B_mtle, B_tcs, B_du, tf, tv, vf
       96
                                                                         , la , taub , tauc , Rp)
      97
                                                 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)+"
      98
       99
100
101
                                                                                           s '
102
                                                           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), " ")
103
 104
 105
 106
 107
                                                            if sav==1 then
fnw=input("
 108
 109
                                                                                                                                                                                                                filename: ","s")
                                                                                          fid=mopen(fnw, "w")
110
                                                                                        if fid == -1 then
 111
                                                                                                                  write (\%io(2), "
112
                                                                                                                                                                                                                                                                            Error: Cannot open file for writing!")
                                                                                                                                                                                                                                                                        ... writing data to file: "+fnw)
")
                                                                                          else
 113
                                                                                                                    write(%io(2),"
114
                                                                                                                  115
116
   117
118
   119
120
                                                                                                                    mfprintf(fid, "%s \t %s \
121
                                                                                                                  ")

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 \
 122
123
   124
 125
   126
 127
 128
                                                                                                                                              mfprintf(fid, "%1.5e \t %1.5e \t %
 129
130
                                                                                                                  end
```

131 mclose (fid) 132 end

```
132 e
133 end
```

```
134
135 endfunction
```

#### ground/func\_plot.sci

```
1 // v7.1
    2
     3 // functions to plot the field (plot_field, plot_all_field)
     4
               // written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics
     5
     6
      8
     9
 10
 11
 12
              // functionname plot_field
 13
 14 // function to plot field and its components
 15
              // 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
// tend ...
 16
 17
  18
 19
                                                                          B1, B7, Bd ... components of
tend ... endtime of calculation
tv ... time-vector
tzoom ... zoomtime in us
fn ... startnumber for figure
20
21
22
23
\frac{24}{25}
               // output: none
26
27
              function [] = plot_field (rsmt, E, Ee, Ei, Er, Ed, B, Bi, Br, Bd, tend, tv, tzoom, fn)
28
29
                                                    t_{zoom} = t_{zoom} * 10^{(-6)}
30
31
                                                      wide = 1000
32
33
                                                     high=500
tit=""
 34
                                                      select rsmt
case 1 then
    tit="Transmission Line Model"
35
36
37
38
39
                                                      case 2 then
                                                                          tit="Modified Transmission Line Model"

    40 \\
    41

                                                      case 3 then
    tit="Travelling Current Source Model"
                                                      case 4 then
tit="Diendorfer Uman Model"
42 \\ 43
\frac{44}{45}
                                                     end
                                                           f1=scf(fn) \ // \ select \ figure \\ clf(f1) \ // \ clear \ if \ exist \\ tmax=find(tv>=tend,1) \ // \ find \ next \ index \ according \ to \ endtime \ given \\ log Direction (from the tend of tend
46
47
 48
 49
                                                      if Ed<>0
                                                                    Ed<>0
plot2d ((tv(1:tmax)*10^6)',[Ee(1:tmax)' Ei(1:tmax)' Er(1:tmax)' Ed(1:tmax)' Ed(1:tma
50
                                                                                          )'
\frac{51}{52}
                                                       else
                                                                      plot2d ((tv(1:tmax)*10^6)', [Ee(1:tmax)' Ei(1:tmax)' Er(1:tmax)' E(1:tmax)
                                                                                                     '], style = \begin{bmatrix} 2 & 13 & 5 & 1 \end{bmatrix})
                                                       end
 53
                                                      f1.figure_size=[wide high]
f1.children.grid=[1_1]
 54
                                                      55
 56
 57
                                                                hl=legend("$electrostatic \ component$","$induction \ component$","
$radiation \ component$","$discontinuity \ component$","$electrical \
field$",-1)
58
 59
                                                       else
                                                              \label{eq:hless} \begin{array}{l} hl = legend ("\ensuremath{\$} electrostatic \ component\ensuremath{\$}","\ensuremath{\$} induction \ component\ensuremath{\$}","\ensuremath{\$} electrical \ field\ensuremath{\$}",-1) \end{array}
60
61
                                                      end
 62
                                                       xtitle(tit,"$t \ / \ \mu s$","$E \ / \ \frac{V}{m}$")
 63
                                                      64
65
                                                                       \label{eq:linear} \begin{array}{c} 12-501 \mbox{ (linear inglated ingle)} \\ 12-501 \mbox{ (linear inglated inglated ingle)} \\ 12-501 \mbox{ (linear inglated ingl
\frac{66}{67}
68
69
                                                                                   Ed <> 0
                                                                                        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])
```

```
70
                                      else
                                             plot2d((tv(1:tzoom)*10^6)',[Ee(1:tzoom)' Ei(1:tzoom)' Er(1:tzoom)' E
(1:tzoom)'],style=[2 13 5 1])
  71
  72
                                     end
  73
74
75
                                     f2.figure_size=[wide high]
                                     f2. children.grid = \begin{bmatrix} 1 & 1 \end{bmatrix}
f2. children.margins(2) = 0.3
                                    if Ed<>0
    h1=legend("$electrostatic \ component$","$induction \ component$","
    $radiation \ component$","$discontinuity \ component$","
    $electrical \ field$",-1)
  \frac{76}{77}
  78
                                            \label{eq:hl=legend} \begin{array}{c} \mbox{legend} ( \mbox{"}\mbox{electrostatic } \mbox{component}\ ,\ \mbox{"}\ \mbox{sinduction } \mbox{component}\ ,\ \mbox{"}\ \mbox{sinduction } \mbox{component}\ ,\ \mbox{"}\ \mbox{sinduction}\ \mbox{component}\ \mbox{sinduction}\ \mbox{component}\ \mbox{sinduction}\ \mbox{component}\ \mbox{sinduction}\ \mbox{component}\ \mbox{sinduction}\ \mbox{component}\ \mbox{sinduction}\ \mbox{component}\ \mbox{sinduction}\ \mbox{sinduction}\ \mbox{sinduction}\ \mbox{component}\ \mbox{sinduction}\ \mbox
  79
  80
                                     \mathbf{end}
                                     xtitle(tit,"$t \ / \ \mu s$","$E \ / \ \frac{V}{m}$")
  81
  82
                            end
  83
                            f3=scf(f1.figure_id+2) // select figure
clf(f3) // clear if exist
if Bd<>0
  84
  85
  86
87
                                    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])
  88
                             else
                                   se
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])
  89
  90
                            end
                            f3. figure_size = [wide high]
f3. children.grid = \begin{bmatrix} 1 & 1 \end{bmatrix}
f3. children.margins (2) = 0.3
  91
  92
  93
  94
                             if Bd<>0
                                 hl=legend("$induction \ component$","$radiation \ component$","
$discontinuity \ component$","$magnetic \ field$",-1)
  95
  96
                             else
                                h1 = legend ("\$induction \ \ component\$","\$radiation \ \ component\$","\$magnetic
  97
                                               \ field$",-1)
  98
                            end
                            xtitle(tit,"$t \ / \ \mu s$","$B \ / \ \mu T$")
  99
100
 101
                             if tzoom>0
                                     fd=scf(f1.figure_id+3) // select figure
clf(f4) // clear if exist
if Bd<>0
102
 103
104
                                          plot2d((tv(1:tzoom)*10^6)',[(Bi(1:tzoom)*10^6)' (Br(1:tzoom)*10^6)' (Bd(1:tzoom)*10^6)' (B(1:tzoom)*10^6)'], style=[2 13 5 1])
105
106
                                             plot2d((tv(1:tzoom)*10<sup>6</sup>)',[(Bi(1:tzoom)*10<sup>6</sup>)' (Br(1:tzoom)*10<sup>6</sup>)' (
B(1:tzoom)*10<sup>6</sup>)'], style=[2 13 1])
107
108
                                     end
                                     f4. figure_size = [wide high]
f4. children.grid = \begin{bmatrix} 1 & 1 \end{bmatrix}
f4. children.margins (2) = 0.3
109
110
111
                                     if Bd<>0
112
                                             hl=legend("$induction \ component$","$radiation \ component$","
$discontinuity \ component$","$magnetic \ field$",-1)
113
114
                                      else
                                             \label{eq:hlegend} \begin{array}{l} h1 = legend ("\$induction \ component\$","\$radiation \ component\$","\$magnetic \ field\$",-1) \end{array}
115
116
                                     end
                                      xtitle(tit,"$t \ / \ \mu s$","$B \ / \ \mu T$")
117
                            end
118
119
120 endfunction
121
122
123 // functionname plot_all_field
124
125 // function to plot fields of all models
 126
          // 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
127
128
129
                                        tend ...
 130
                                        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
131
 132
133
 134
135
 136
          // output: none
137
 138
         function [] = plot_all_field (E_tl, E_mtle, E_tcs, E_du, B_tl, B_mtle, B_tcs, B_du, tend, tv,
139
                        tzoom, fn, la, taub, tauc)
140
```

```
tzoom=tzoom*10^{(-6)}
141
142
143
           wide = 1000
144
          \rm hig\,h\!=\!500
145
          txt1="$TL \ model$"
txt2="$MTLE \ model: \ \lambda="+string(la)+"m $"
txt3="$TCS \ model:"
if tauc==0 then
txt4="$DU \ model: \ \tau ="+string((taub*10^6))+"\mu s$"
146
147
148
149
150
151
           else
            sise
txt4="$DU \ model: \ \tau_{b} ="+string((taub*10^6))+"\mu s, \ \tau_{c} ="+
    string((tauc*10^6))+"\mu s$"
152
153
          end
154
          155
156
157
158
          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", "$t \ / \ mu s$", "$E \ / \ \frac{V}{m}$")
159
160
161
162
163
164
           if tzoom>0
165
             166
167
168
169
170
171
              f2.children.grid=[1 1]
f2.children.margins(2)=0.3
xtitle("All Models","$t \ / \mu s$","$E \ / \ \frac{V}{m}$")
172
173
174
175 \\ 176
           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","$t \ / \ mu s$","$B \ / \ mu T$")
177
178
179
180
181
182
183
184
185
186
           if tzoom>0
              f4=scf(f1.figure_id+3) // select figure clf(f4) // clear if exist
187
              r4=sci(r1.rigure_id+3) // select rigure
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.children.grid=[1 1]
f4.children.margins(2)=0.3
bl=bcood(txt1 txt2 txt2 txt4 1)
188
189
190
191
192
          h1=legend(txt1,txt2,txt3,txt4,-1)
xtitle("All Models","$t \ / \ mu s$","$B \ / \ \mu T$")
end
193
194
195
196
197 endfunction
```

### A.1.5 Definitionfile

ground/def\_ground.txt

definitionfile for ground remarks: \*\*\*\*\* start\_of\_def tend ta vf 1.3 $\operatorname{Rp}$ currents h 0.73 $^{13}_{7}$ h 

```
x
                           0
                                                     0
                                                                               0
                                                                                                         0
  x
x
                           0
                                                     0
                                                                               0
                                                                                                         0
                           0
                                                     Ő
                                                                               Ő
                                                                                                          õ
   modelparameter
  la
                                           2000
  taub
                                           0.1
 tauc
                                           0.1
 end_of_def
    *****
  format of data:
                                          endtime of field calculation in us
number of timesteps
return stroke front wave speed in the channel in 10°8 m/s
  \operatorname{tend}
  ta
vf
 _{\mathrm{Rp}}
                                           distance of field point in km
  currents
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 double fo
 a b
                                                   c d
                                                                                             е
                                                                                                                                  \mathbf{f}
 e ... n (need only for heidler-function)
  modelparameter
                                          exponential decay coefficient for MTLE model in m
breakdown discharge time constant for DU model in us
corona discharge time constant for DU model in us
  taub
 tauc
  for copy/paste:
 Nucci90:
                                                                    0.072
                           -9.9
 h
                                                                                                                   5
                                                                                                                                            2
  de
                          -7.5
                                                                    100
                                                                                                                    6
                                                                                                                                             0
  Diendorfer 90:
                          ^{-13}_{-7}
 ^{\rm h}
                                                                0.73
                                                                                                         \frac{3}{50}
                                                                                                                                       \frac{2}{2}
 h
                                                               \mathbf{5}
```

### A.2 Lightning Strike to a Tall Object

### A.2.1 Startmodule

### tall/start.sce

```
1 / / v3.1
 2
 3
    // startmodule for calculation of fields for a strike on a tall grounded object
 4
    // written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics
 \frac{5}{6}
 \frac{8}{9}
10
        funcprot(0)
11 \\ 12
        stacksize('max')
13 \\ 14
         exec ("def.sci", -1)
        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)
15 \\ 16
17
18
19
20
21
        22
23
24
25
26
27
        d=0 // discontinuity field
```

```
28
      [tv,H,rhotop,rhobot,nstop,vf,Rp,I0] = def(); // input parameter
29
30
      [zvc,zvt] = heightvector(tv,Rp,H,vf); // calculate heightvectors
31
32
      [Izt,zv] = distribution(tv,zvt,zvc,I0,rhotop,rhobot,nstop,vf); // calculate
33
             current distribution
34
35
      [E, Er, Ei, Ee, B, Br, Bi, tf] = field (Izt, zv, tv, Rp); // calculate fields
36
      37
38
39
           [E,Ed,B,Bd] = d_field(Izt,zv,zvc,tv,Rp,E,B) // calculate discontinuity
40
      end
41
      write(%io(2), "figures of fields")
write(%io(2), "------")
write(%io(2), " ")
fig=input(" Draw figures of fields? 1 for yes, other for no: ")
42
43
44
45
      ...g=input(" Draw
write(%io(2)," ")
if fig==1
tzoom
\frac{46}{47}
                            ")
48
49
         tzoom=input("
                             zoom at tmax in us (0 for none): ")
         write (%io(2),"
                                )
50 \\ 51
        fn=input("
write(%io(2),"
                              start figure number: ")
                                ")
52
53
54
55
         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);
\frac{56}{57}
         else
              plot_field (E, Ee, Ei, Er, B, Bi, Br, tv, tzoom, fn, vf, Rp, Hi);
\frac{58}{59}
         end
60
      end
61
      save_field (E, Ee, Ei, Er, B, Bi, Br, tf, tv, H, rhotop, rhobot, vf, Rp);
62
```

### A.2.2 Definition of Parameters

### tall/def.sci

```
1 / / v3.1
 2
 \mathbf 3 // function to define parameter and plot IO
 4
    // written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics
 \mathbf{5}
 6
 8
 9
10 // functionname def
11
12
    // input: none
13
          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
IO ... undisturbed current in A
14
15
16

    \begin{array}{c}
      17 \\
      18 \\
      19
    \end{array}

20
21
22
23
     function [tv,H,rhotop,rhobot,nstop,vf,Rp,I0] = def()
24
         exec ("rsc.sci",-1) // load return stroke current functions
25
26
27
         c\!=\!299792458 // speed of light
28
29
         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: "
30
31
32
                                       ")
         write(%io(2),"
33
34
35
         if rd==1 then
            fid=mopen("def_tall.txt","r")
if fid==-1 then
36
37
38
                 write(%io(2),"
30
                                               error: cannot open file def.txt!")
```

```
40
                else
 \frac{41}{42}
                   header=1
                    while header==1
val=mfscanf(1,fid,"%s")
if val=="start_of_def" then, header=0, end
 43
  44
 45
                    end
 46
                    data=1
 \begin{array}{c} 47 \\ 48 \end{array}
                    while data==1
val=mfscanf(1,fid,"%s")
                       49
 50
 51
 52
53
 54
 55
56
57
 58
59
 62
63
                                I=zeros(tv)
i=1
 \frac{64}{65}
                                 n e x t = 1
                                 while next==1
                                     [dummy,typ,I0,T1,T2,n]=mfscanf(1,fid,"%s %f %f %f %f %f %f %f ")
if typ="x" then
 \frac{66}{67}
 68
69
                                         next=0
                                     else
                                         \begin{array}{c} 1 \text{ se} \\ 10 = 10 * 10^{3} \\ \text{T1} = \text{T1} * 10^{-6} \\ \text{T2} = \text{T2} * 10^{-6} \\ \text{if typ} = \text{"h" then} \\ \end{array} 
 70 \\ 71 \\ 72 \\ 73 \\ 74 \\ 75
                                             76
                                         end
  77
                                          if typ=="de" then
                                             I (j,:)=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
 78
 79
 80
                                         end
 81
                                         i\!=\!i\!+\!1
                                    end
 82
                            end
case "end_of_def", data=0,
 83
 84
 85
                        end
                   \mathbf{end}
 86
 87
88
                \mathbf{end}
                mclose(fid)
 89
90
                \operatorname{errcatch}(-1)
               lse
  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<sup>(</sup>(-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)," ")
 91
            else
 92
 93
 94
 95
 96
 97
 98
 99
100
101
102
103
104
105
106
107
               write(%io(2)," ")
write(%io(2)," ")
write(%io(2),"tall objekt")
write(%io(2),"------")
write(%io(2)," ")
H=input(" Height [m]: ")
mrite(%ic(2),"
108
109
110
111
112
113
                write(%io(2),"
rhotop=input("
114
                                                    current reflection coefficient at the top []: ")
115
               write(%io(2),"
rhobot=input("
116
                                                  current reflection coefficient at the bottom []: ") ")
117
               write(%io(2),
nstop=input("
118
                                                 max. number of reflections to calculate []: ")
119
```

```
120
               write(%io(2),"
                                               ")
121
               write(%io(2)," ")
write(%io(2)," ")
write(%io(2)," channel")
write(%io(2)," channel")
write(%io(2)," ")
vf=input(" return stroke front speed [10^8 m/s]: ")
vf=vf*10^(8)
write(%io(2) " ")
122
123
124
125
126
127
128
               write(%io(2),"
                                               ")
129
130
               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) " ")
131
132
133
134
135
136
137
                write (\%io(2), "
                                                 ")
138
139
               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)," ")
140
141
142
143
144
145
146
147
               148
149
150
151
152
                   if b>2 then, break, end
153
154
                    if b==1
                        write(%io(2),"
                                                     Heidler:")
IO [kA]: ")
155
                       I0=input("
I0=I0*10^3
156
157
158
                                                     T1 [us]: ")
                        T1=input (
                        T1=T1*10^{(-6)}
159
                       T_{2}=T_{1}*10 ("
T_{2}=T_{2}*10^{-}(-6)
160
                                                     T2 [us]: ")
161
                       T2=12*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 firmer
162
163
164
165
                                 // for figure
166
                   end
167
168
                    if b==2
                        169
                       I0=input("
I0=I0*10^3
170
171
                       T_{1}=10 * 10^{-3}

T_{1}=input ("

T_{1}=T_{1}*10^{-}(-6)

T_{2}=input ("

T_{2}=T_{2}*10^{-}(-6)
172
                                                     T1 [us]: ")
173
174
                                                     T2 [us]: ")
175
                       II = 12 + 10 (-0)
write(%io(2), " ")
I(i,:) = dexp(10,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 $" // for figure
176
177
178
179
                   end
               end
180
181
               if b==3 // define current used by Nucci et al. (1990)
I(1,:)=heidler(-9900,0.072*10^(-6),5*10^(-6),2,tv)
txt1(1)="$Heidler: \ I_{0}=-9.9kA, \ T_{1}=0.072\mu s,\ T_{2}=5\mu s,\ n=2$
182
183
184
                   I(2,:)=dexp(-7500,100*10^(-6),6*10^(-6),tv)
txt1(2)="$double \ exp: \ I_{0}=-7.5kA, \ T_{1}=100\mu s, \ T_{2}=6\mu s$"
185
186
187
               end
188
               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)="$Heidler: \ I_{0}=-7kA, \ T_{1}=5\mu s, \ T_{2}=50\mu s, \ n=2$"
ord
189
190
191
192
193
               end
194
195
           end
196
197
           a=length(I)/length(tv) // real number of functions given
198
```

```
ta\!=\!length(tv) // number of timesteps of current I0\!=\!zeros(1,ta) // build inital matix of current
199
200
201
       for i=1:a
202
          IO=IO+I(i,:) // calculate current at origin
203
       end
204
205
       if H = = 0
206
207
          rhotop=-1 // set rhottop that IO equals I(0,t)
       end
208
209
       write(%io(2),"figure of current")
write(%io(2),"-----")
write(%io(2)," ")
210
211
212
       fig=input(" Draw
write(%io(2)," ")
213
                               figures of current? 1 for yes, other for no: ")
214
215
216
       if fig == 1
         217
218
219
          fn=input(" start figure number: ")
write(%io(2)," ")
220
221
         f1=scf(fn) // seclect figure
clf(f1) // clear if exist
tmaxl=length(tv)
222
223
224
          if a>1
for i=1:a
225
226
              plot2d((tv(1:tmax1)*10^6),(I(i,1:tmax1)*10^(-3)),style=(i+1)) // plot
227
                   part i of current
228
            \mathbf{end}
229
          end
          plot2d((tv(1:tmax1)*10^6),(I0(1:tmax1)*10^(-3)),style=1) // plot complete
230
231
          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),"$short-circuit \
232
233
234
235
236
                 current$",4),
237
          end
          xtitle("","$t \ / \ \mu s$","$I \ / \ kA$")
238
239
240
          if tmax2>0
            f2=scf((f1.figure_id+1)) // select figure clf(f2) // clear if exist
241
242
243
244
245
            tmax2=tmax2*10^{(-6)}
246
247
            tmax2=find(tv>=tmax2,1) // find next index according to zoom given
248
            if a > 1
for i=1:a
249
250
251
                 plot2d ((tv(1:tmax2)*10^6),(I(i,1:tmax2)*10^(-3)),style=(i+1)) // plot
                       part i of zoomed curren
252
              end
253
            end
254
            plot2d((tv(1:tmax2)*10^{6}),(I0(1:tmax2)*10^{(-3)}),style=1) \ // \ plot \ zoomed
            f2.figure_size = [750 \ 500]
255
            256
257
258
259
260
261
262
          end
263
          \label{eq:fig} \begin{array}{ccc} fig=\!input(" & Draw figures of current derivative? 1 for yes, other for no: ") \\ write(\%io(2)," & ") \end{array}
264
265
266
267
          if fig == 1
268
            269
270
271
272
            dI0=splin(tv,I0)
273
            f3=scf((f1.figure_id+2)) // select figure
274
            clf(f3) // clear if exist
plot2d((tv(1:tmax1)*10^6),(dI0(1:tmax1)*10^(-9)),style=1) // plot complete
275
                  current derivative
```

```
f3.figure_size = [750 500]

xtitle("","$t \ / \ \mu s$","$\frac{\mathrm{d}I}{\mathrm{d}t} \ / \ \frac{

kA}{\mu s}$")
276
277
278
279
                 if tmax2>0
                     f4=scf((f1.figure_id+3)) // select figure clf(f4) // clear if exist tmax2=tmax2*10^(-6)
280
281
282
                     tmax2=tmax2*10 (-0)
tmax2=find(tv=tmax2,1) // find next index according to zoom given
plot2d((tv(1:tmax2)*10<sup>6</sup>),(dI0(1:tmax2)*10<sup>(-9)</sup>),style=1) // plot zoomed
283
284
                     f4.figure_size = [750 500]

xtitle ("", "$t \ / \ \mu s$", "$\frac{\mathrm{d}I}{\mathrm{d}t} \ / \ \

frac{kA}{\mu s}$")
285
286
287
                 end
             end
288
289
        end
290 endfunction
```

### tall/rsc.sci

```
1 // v3.1
 2
 \overline{3} // functions to calculate return stroke currents at channel origin
 4
   // written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics
 \mathbf{5}
 6
 9
10
11
12
   // functionname heidler
13
   // function to calculate current at channel origin // as a heidler function
14
15
16
   // input: I0, T1, T2, n (scalar)
// tvi ... timevector (vector 1:m)
17
18
19
20
21
   // output: I0t=I(0,t) ... current at channel origin (vector 1:m)
22
23
   function IOt=heidler(IO,T1,T2,n,tvi)
24
         eta = exp(-T1/T2*(n*T2/T1)^{(1/n)})
25
         \begin{array}{l} \mbox{fracl=(tvi/T1).^n // to reduce prozessing time} \\ \mbox{I0t=I0/eta*(frac1.*(ones(frac1)+frac1).^(-1)).*exp(-tvi/T2)} \end{array} \end{array} 
26
27
28 endfunction
29
\frac{30}{31}
32
33
   // functionname dexp
34 // function to calculate current at channel origin 35 // as a double-exponential function
36
   // input: I0, T1, T2 (scalar)
// tvi ... timevector (vector 1:m)
37
38
39
40 // output: I0t=I(0,t) ... current at channel origin (vector 1:m)
41
42 function I0t=dexp(I0,T1,T2,tvi)
43
        I0t = I0 * (exp(-tvi/T1) - exp(-tvi/T2))
44
45
46 endfunction
```

#### tall/heightvector.sci

1 // v3.1
2
3 // function to calculate the heigthvectors
4
5 // written by Andreas F. Dvorak
6 // Vienna University of Technology
7 // Faculty of Electrical Engineering and Information Technology
8 // Institute of Power Systems and Energy Economics
9
10 // functionname heightvector
11

```
// 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
13
14
15
16
     // output: zvc ... heightvector of channel
// zvt ... heightvector of tall object
17
18
19
20 function [zvc, zvt] = heightvector(tv, Rp, H, vf)
21
22
23
         c = 299792458 // speed of light
         zvt=0
\frac{24}{25}
         write(%io(2)," ")
write(%io(2)," ")
write(%io(2),"calculation of heightvector")
write(%io(2),"-----")
write(%io(2)," ")
26
27
28
29
30
31
        32
33
34 //
            delz=c*tv(2)
zvt=0:delz:(H+delz)
35
36
            zvt=0.defile(zvt>H,1))
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))
37
38
39
40
\frac{41}{42}
            else
                zvt=zvt(1:(find(zvt>H,1)))
\begin{array}{c} 43 \\ 44 \end{array}
            end
         end
45
        Hi=max(zvt)
46
47
     // ---> tall object end <----
\frac{48}{49}
               -> channel begin <-
50
        D=c*(tv+Hi/vf)+sqrt(Rp^2+Hi^2) // auxiliary variable
        \frac{51}{52}
53 \\ 54
         zvc=(-B-sqrt(B.^2-4*A*C))/(2*A) // calculate heightvector of channel
55
\frac{56}{57}
        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)," ")
58
59
60
61
62
63
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), " ")
66
67
68
69
70
71
72
73
74 endfunction
```

#### Calculation of current distribution A.2.3

#### tall/distribution.sci

1 / / v3.12 // function to calculate the current distribution in the tall object and the 3 4 4
5 // written by Andreas F. Dvorak
6 // Vienna University of Technology
7 // Faculty of Electrical Engineering and Information Technology
8 // Institute of Power Systems and Energy Economics

- 9
- 10 // functionname distribution
- 12 // input: tv ... timevector in s

```
zvt ... heightvctor of tall object in m
^{13}
                  zvc ... heightvetor of channel in m
IO ... 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
14
15
16
17
^{18}_{19}
20
21
22
       output: Izt ... current distribution in the channel in A zv ... heightvector in m (tall object and channel)
22 // 23 24 function [Izt,zv] = distribution(tv,zvt,zvc,I0,rhotop,rhobot,nstop,vf)
25
26
       c=299792458 // speed of light
      27
28
29
30
31
32
       Iztt=zeros(zvta,tva) // build inital matix of current distribution in tall
33
       Iztc=zeros(zvca,tva) // build inital matix of current distribution in channel
\frac{34}{35}
      36
37
38
39
\begin{array}{c} 40\\ 41 \end{array}
                               \dot{\ldots} calculating current distribution")
42
43
       // ---> start: current distribution in tall object <---
\frac{44}{45}
       if zvta > 1 then
46
47
                  i=1:zvta
             for
               I\bar{ztt}((zvta-j+1), j:tva) = (1 - rhotop) * I0(1:(tva-j+1)) // calculate
48
                      transmitted wave
49
            end
50
\frac{51}{52}
             stop=0
            m=0
53 \\ 54
            //// ---> start: reflected waves <---
55
56
57
            while stop==0
               m=m+1
58
59
               for i=1:zvta
60
                   stu = (zvta + 2*(m-1)*(zvta - 1)+j-1) // index of starttime of the upward
                                     wave on bottor
61
                   \texttt{std} = (\texttt{zvta} + (2*m-1)*(\texttt{zvta} - 1) + \texttt{j} - 1) \ // \ \texttt{index} \ \texttt{of starttime} \ \texttt{of the downward}
                         running wave on top
                   if (stu>=tva) then
62
63
                     stop=1
                     stop-1
se // upward running
IOi=[zeros(1,stu-1) IO(1:(tva-stu+1))]
64 \\ 65
                   else
66
                     Iztt(j,:)=Iztt(j,:)+(1-rhotop)*rhobot^m*rhotop^(m-1)*I0i // add n-th
reflected wave on bottom
67
                   end
                   if (std>tva) then
68
                  if (std>tva) then
stop=1
else // downward runnig
IOi=[zeros(1,std-1) IO(1:(tva-std+1))]
Iztt((zvta-j+1),:)=Iztt((zvta-j+1),:)+(1-rhotop)*(rhobot*rhotop)^m*
IOi // add n-th reflected wave on top
]
69
70
71 \\ 72
73
74
75
76
               end
                if m = nstop, stop = 1, end \\
77
78
79
            end
80
             //// ---> end: reflected waves <----
81
      end
82
83
       // ---> end: current distribution in tall object <----
84
85
       // ---> start: current distribution in channel <---
86
87
       if length(zvt)>1 then
88
89
         for j=1:zvca
90
```

1 / / v3.1

```
 \begin{array}{l} Iztc\,(j\,,:)=&[zeros\,(1\,,(j-1)) \ IO\,(1:tva-j+1)] \ // \ built \ injected \ wave \\ ti=&(zvc\,(j\,)-zvc\,(1)\,)/c \ // \ starttime \ of \ reflected \ wave \ at \ height \ j \\ IOi=&(-1)*(rhotop)*interp\,((tv-ti\,),tv\,,IO\,,dIO\,,"by_zero") \ // \ built \ reflected \end{array} 
 91
 92
 93
                         \begin{array}{l} \textbf{Iztc}(j,:) = \textbf{Iztc}(j,:) + \textbf{I0i}(1:tva) \ // \ \text{add reflected wave} \\ \textbf{Iztc}(j,:) = [\texttt{zeros}(1,(j-1)) \ (1-\texttt{rhotop})*\textbf{I0}(1:tva-j+1)] \ // \ \text{stop at return} \end{array} 
 94
 95
                 front
 96
               end
 97
                for m=1:nstop
 98
  99
                   for j=1:zvca
ti=(zvc(j)+zvc(1)*(2*m-1))/c // starttime of n-th transmitted wave at
100
                         height j
if ti > (zvc(j)-zvc(1)/vf) then ti = (zvc(j)-zvc(1)/vf); end stop at
101 //
                              -front
                       IOi=(1-rhotop)*(1+rhotop)*rhobot^(m)*rhotop^(m-1)*interp((tv-ti),tv,IO,
dIO,"by_zero") // built n-th transmitted wave
Iztc(j,:)=Iztc(j,:)+IOi(1:tva) // add n-th transmitted wave
102
103
104
                    end
               end
105
\begin{array}{c} 106 \\ 107 \end{array}
               Iztc=triu(Iztc) // Heavyside
108
109
           end
110
            // ---> end: current distribution in channel <-
111
112
           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
113
114
115
116
            else
               zv = zvc
117
118
               Izt=Iztc
119
            end
120
121 endfunction
```

### A.2.4 Calculation, plotting and saving of fields

tall/field.sci

```
2
 \frac{1}{3} // function to calculate the fields
 4
     // written by Andreas F. Dvorak
// Vienna University of Technology
// Faculty of Electrical Engineering and Information Technology
// Institute of Power Systems and Energy Economics
 5
  6
  8
10 // functionname field
11
     // 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
12
13
14
15
\begin{array}{c} 16 \\ 17 \end{array}
      // output:E=E(t) ... vertical electrical field in V/m
// Ee, Ei, Er ... components of vertical electrical field in V/m
// B=B(t) ... horizontal magnetic field in T
// Bi, Br ... components of horizontal magnetic field in T
// tf ... time of arrival of field at field point in s
^{18}_{19}
20
21
22
23
      function [E, Er, Ei, Ee, B, Br, Bi, tf] = field (Izt, zv, tv, Rp)
\frac{24}{25}
           Er=0
26
           Br=0
27
28
           Ei=0
           Bi=0
29
30
           Ee=0
          31
32
33
34
35
36
          tva=length(tv) // number of timesteps
zva=length(zv) // number of channelsteps
37
           \begin{array}{l} tf{=}sqrt\left(H^{2}{+}Rp^{2}\right)/c \ // \ starttime \ of \ field \\ R{=}sqrt\left(Rp^{2}{+}zv\,.^{2}\right) \ // \ distance \ dipole \ - \ field \ point \end{array} 
38
39
40
           write(%io(2)," ")
41
```

```
write(%io(2)," ")
write(%io(2),"calculation of fields")
write(%io(2),"------")
write(%io(2)," ")
 42
 43
 44
 45
 46
     // ----> start: field calculation <----
 47
 48
     //// ---> start: radiation component calculation <---
write(%io(2)," ... calculating radiation field")
write(%io(2)," ")
 49
 50
 51
 52
         efac = (-1)*(Rp^2)./(c^2*R.^3*(2*\%pi*eps0)) // factor electric radiation
 53
         bfac=(mu0*Rp)./(2*%pi*c*R.^2) // factor magnetic radiaion component
 54
        dF=zeros(zva,tva)
 55
           or j=1:zva
dF(j,:)=splin(tv,Izt(j,:))
 56
         for
 57
        end
 58
 59
60
         for i=1:tva
 \frac{61}{62}
           Er(i)=inttrap\left(zv,\left(dF\left(:\,,\,i\right)\,'.*efac\right)\right)\ // intergrate over the channel Br(i)=inttrap\left(zv,\left(dF\left(:\,,\,i\right)\,'.*bfac\right)\right)
 63 \\ 64
        end
     //// ----> end: radiation component calculation <----
 65
        // ---> 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</pre>
 66
 67
 68
 69
         bfac=(mu0*Rp)./(2*%pi*R.^3) // factor magnetic induction component
 70
71
72
73
74
75
76
        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
 77
78
                 -> 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=gares(gua tua)
 79
 80
 81
 82
 83
        dF=zeros(zva,tva)
 84
        for j=2:zva
index=find(Izt(j,:)<>0,1)
 85
 86
            if index == [], break, end
for i=index:tva
 87
 88
 89
                dF(j,i) = dF(j,(i-1)) + tv(2) * (Izt(j,(i-1)) + Izt(j,i)) / 2
 90
           end
 91
92
         end
 93
94
        Ee(i)=inttrap(zv,(dF(:,i)'.*efac)) // intergrate over the channel
        end
 95
 96
                 \rightarrow end: electrostatic component calculation <-
 97
        E = Er + Ei + Ee
 98
 99
        B=Br+Bi
100
101
    // ---> end: field calculation <---
102
103 endfunction
```

tall/plot\_field.sci

1 // v3.1
2
3 // function to plot the fields and its components
4
5 // written by Andreas F. Dvorak
6 // Vienna University of Technology
7 // Faculty of Electrical Engineering and Information Technology
8 // Institute of Power Systems and Energy Economics
9
10 // functionname plot\_field
11
12 // input: E=E(t) ... vertical electrical field in V/m
13 // E=E(t) ... componenets of vertical electrical field in V/m
14 // B=B(t) ... horizontal magnetic field in T
15 // Bi, Br ... componenets of horizontal magnetic field in T
16 // 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  $18 \\ 19$  $^{23}$ // output: none 25 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"  $\frac{27}{28}$ if abs(max(E)) < abs(min(E)) then // allways plot positiv pe=-1 else pe=1 end 35 37 ∠s(ma pb=−1 else if abs(max(B)) < abs(min(B)) then // allways plot positiv 39 pb=1 end  $40 \\
 41$ 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] if tzoom>0 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: protect ((tv(1,t200m)\*10 0), [Le(1,t200m) E1(1,t200m) E1(1,t2 end f3=scf(f1.figure\_id+2) // select figure 62 clf(f3) // clear if exist plot2d((tv(1:tmax)\*10^6)',[(Bi(1:tmax)\*10^6) (Br(1:tmax)\*10^6) (B(1:tmax) piot2a((tv(1:tmax)\*10 0) ',[(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) if tzoom>0 \ field\$",1)
xtitle("\$magnetic \ field \ in \ "+string(Rp\*10^(-3))+"km: \ "+txt1+"\$",
 "\$t \ / \ \mu s\$","\$B \ / \ \mu T\$") end 76 endfunction

### tall/save\_field.sci

 $1 \ // \ v3.1$  // 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
      8
    g
 10
                // functionname save_field
 11
                                       input: E=E(t) ... vertical electrical field in V/m
Ee, Ei, Er ... components of electrical field in V
B=B(t) ... horizontal magnetic field in T
Bi, Br ... components of magnetic field in T
tf ... time of arrival of field at field point in s
ty timevector
  12
                                                                                                                                                                                                                                                                                                                                                                                          field in V/m
 13
  14
15
16
17
                                                                                        H ... time of all val 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
18
19
20
21
22
23
24 // output: none
25
26
27
                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 a
28
29
\frac{30}{31}
                                   sav=input(" save
write(%io(2)," ")
if sav=1 then
errect.'
                                                                                                                                                                field and components to file? 1 for yes, other for no: ")
32
33
                                                 \frac{34}{35}
                                               errcatch (999, "continue")
fnw=input(" filename:
fid=mopen(fnw, "w")
if fid==-1 then
write(%io(2), " Error
else
methe(%io(2))
36
37
38
39
                                                                                                                                                                                      Error: Cannot open file for writing!")
                                                           write(%10(2)," Error: cannot open file for writing. ;
lse
write(%10(2)," ... writing data to file: "+fnw)
write(%10(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 \t %1.5f \n", "rhobot:",rhobot)
mfprintf(fid, "%s \t %3.2f %s \n", "field and components at fieldpoint: ",(Rp
 *10^(-3)), "km")
mfprintf(fid, "%s \t \t %s \t
\frac{40}{41}
 42
 43
 44
 45
 46
 47
 48
 49
50
51
 52
                                                                                          i = 1: length(E)
                                                                            mfprintf(fid, "%1.5e \t %1.5e \t %
53
54
                                                              end
\frac{55}{56}
                                                                mclose(fid)
                                                end
                                \operatorname{errcatch}(-1) end
 57
 58
59
60
                   endfunction
```

### A.2.5 Definitionfile

definitionfile for tall object

### tall/def\_tall.txt

```
remarks:
                    *****
   ******
start_of_def
tend
                        \begin{array}{c} 51 \\ 5000 \end{array}
_{
m H}^{
m ta}
                        168
                        -0.53
rhotop
rhobot
                        0.7
n
vf
                        20
                        1.5
\mathbf{R}\mathbf{p}
                        5
                \begin{array}{c} 0.072\\ 100\\ 0\end{array}
currents
         9.9 \\ 7.5
h
                                      5
                                               2
                                      6
                                               0
de
         0
                                    0
```

```
0
                           0
                                        0
                                                     0
x
x
           0
                          0
                                       0
                                                    0
end_of_def
 ***** **** ********
format of data:
                                  endtime of field calculation in [us]
number of timesteps for field calculation
height of tall object in [m]
current reflection coefficient at the top (tall oblect - channel)
current reflection coefficient at the bottom (tall object - ground)
index of successive multiple reflections
return stroke front wave speed in the channel in [10^8 m/s]
distance of field point [km]
\operatorname{tend}
_{
m H}^{
m t\,a}
 rhotop
 rhobot
_{\rm v\,f}^{\rm n}
\operatorname{Rp}
currents
a b
                         с
                                   d
                                                  е
                                                                 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:
                              \begin{array}{c} 0\,.\,0\,7\,2\\ 1\,00 \end{array}
           -9.9 \\ -7.5
^{\rm h}
                                                     \frac{5}{6}
                                                                     ^{2}_{0}
de
 Diendorfer 90:

\begin{array}{ccc}
-13 & 0.73 \\
-7 & 5
\end{array}

h
                                                     3
                                                                    \frac{2}{2}
                                               50
^{\rm h}
```

# List of Tables

1.1	Values used to calculate the undisturbed current waveforms	
	used by [Nucci et al., 1990] and [Diendorfer and Uman, 1990].	2
1.2	Model specific parameters for generalized current equation	6
2.1	Strike to the ground: List of variables	39
3.1	Strike to a tall object: List of variables	47

# List of Figures

1.1	Undisturbed current N1 and its derivate used by	
	[Nucci et al., 1990]	3
1.2	Undisturbed current D1 and its derivate used by	
	[Diendorfer and Uman, 1990]	4
1.3	Undisturbed current D2 and its derivate used by	
	[Diendorfer and Uman, 1990]	5
1.4	TL model: current along the channel at different times and	
	heights.	8
1.5	TCS model: current along the channel at different times and	
	heights.	10
1.6	Charged areas as base for the DU-model.	11
1.7	Current in the channel at different heights for four engineering	
	models.	12
1.8	Geometry for computation of the remote fields. Adapted from	
	[Thottappillil et al., 1997]	13
1.9	Lightning strike to a tall object. Object and lightning	
	channel are represented by lossless transmission lines con-	
	nected in series with a lumped voltage source. Adapted from	1 5
1 10	[Baba and Kakov, 2005]	15
1.10	Comparison of the process used by [Baba and Rakov, 2005]	
	and in this thesis when a transmitted wave arrives at the	17
1 1 1	Fields in 100km distance in ease of perfecting the discontinu	11
1.11	ity	19
1 1 9	Fields in 5km distance in case of nonlecting the discontinuity	10
1.12	Fields in 100km distance in case of considering the disconti-	19
1.10	nuity	20
1 14	Fields in 5km distance in case of considering the discontinuity	$\frac{20}{21}$
1 15	Fields in 100km distance in case of changing the propagation	
1.10	speed.	22
1.16	Fields in 5km distance in case of changing the propagation	
	speed.	23
	1	

1.17	Comparison of the total fields in 100km distance in all three	
	case	24
1.18	Comparison of the total fields in 5km distance in all three case.	25
2.1	Method of discretization of the channel.	28
2.2	Screenshot: Start of the program to calculate the fields in	
	case of a lightning strike to the ground.	33
2.3	Screenshot: Defining the input parameters via keyboard	34
2.4	Screenshot: Calculation of one Model	36
2.5	Screenshot: Header and first data lines for a one model data-file.	37
2.6	Screenshot: Header and first data lines for an all model data-file.	38
4.1	Comparison of the calculated fields with published results by	
	[Nucci et al., 1990]: Distance 100km, Duration $100\mu s$	52
4.2	Comparison of the calculated fields with published results by	
	[Nucci et al., 1990]: Distance 100km, Duration $5\mu s$	53
4.3	Comparison of the calculated fields with published results by	
	[Nucci et al., 1990]: Distance 5km, Duration $100\mu s$	54
4.4	Comparison of the calculated fields with published results by	
	[Nucci et al., 1990]: Distance 5km, Duration $5\mu s$	55
4.5	Comparison of the calculated fields with published results by	
	[Nucci et al., 1990]: Distance 50m, Duration $100\mu s$	56
4.6	Comparison of the calculated fields with published results by	
	[Nucci et al., 1990]: Distance 50m, Duration $5\mu s$	57
4.7	Comparison of the calculated fields with published results by	
	[Diendorfer and Uman, 1990]: Distance 100km, Duration $5\mu s$	58
4.8	Comparison of the calculated fields with published results by	
	[Diendorfer and Uman, 1990]: Distance 1km, Duration $100 \mu s$	59
4.9	Comparison of the calculated fields with published results by	
	[Diendorfer and Uman, 1990]: Distance $50$ km, Duration $100\mu$ s	60
4.10	Comparison of the calculated fields with published results by	
	[Diendorfer and Uman, 1990]: Distance 200km, Duration $100\mu s$	61
4.11	Comparison of the calculated current at the top and	
	the bottom of the tall object with published results by	
	$[Pavanello et al., 2007] \dots \dots$	62
4.12	Comparison of the calculated Current Distribution with Pub-	
	lished Results by [Pavanello et al., 2007]	63
4.13	Comparison of the Calculated Fields in 100km Distance with	
	Published Results by [Pavanello et al., 2007]	64
4.14	Comparison of the Calculated Fields in 5km Distance with	
	Published Results by [Pavanello et al., 2007]	65
## 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.