Pulse Propagation along close Conductors

Numerical solution of the telegraph equations using Mathematica

Andreas Dieckmann
Physikalisches Institut der Uni Bonn
Elektronik Abteilung
Oktober 2000 - September 2001

Abstract

The propagation and reflection of arbitrarily shaped pulses on non-dispersive parallel conductors of finite length with user defined cross section is simulated employing the discretized telegraph equation. The geometry of the system of conductors and the presence of dielectric material determine the capacities and inductances that enter the calculation. The values of these parameters are found using an iterative Laplace equation solving procedure and confirmed for certain calculable geometries including the line charge inside a box. The evolving pulses and the resulting crosstalk can be plotted at any instant and – in the *Mathematica* notebook version of this report (http://www.physik.uni-bonn.de/~dieckman/) – be looked at in an animation. As an example a differential pair of microstrips as used in the ATLAS vertex detector is analysed.

Contents

Introduction

Electrical pulses travelling on a conductor will induce voltages and currents onto the neighbor conductors.To estimate the size of this influence called *crosstalk* either simulation and/or measurement of the effect is needed. Fig. 1 shows the basic schematic of coupled lines (**Ref. 1**, p.196):

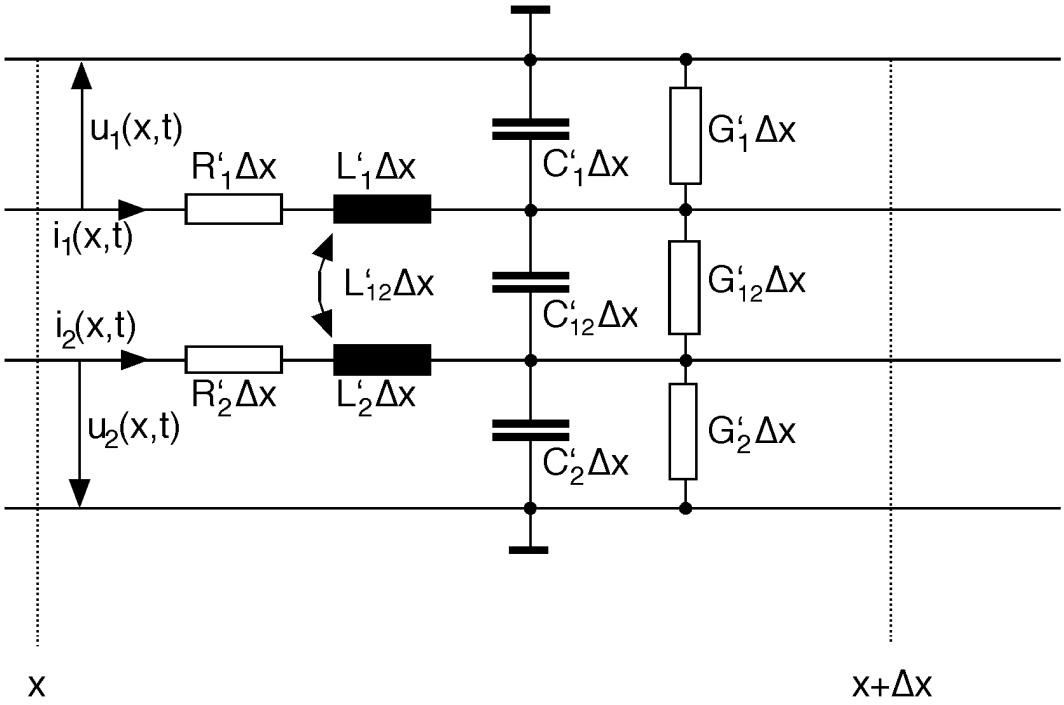


Figure 1

Figure 1. Two coupled pairs of conductors with parameters per unit length ('), voltages and currents on a section of wires ranging from x to x+Δx

The parameters R' (Resistance/m along the conductor), L' (Inductance/m of a circuit), C' (effective Capacity/m between two conductors) and G' (Conductivity/m from one conductor to the next) are normalized to the length of the wire and their index denotes the conductor they refer to. Their numerical values can be found by measurement or calculation (see below).

At frequencies not too high the pulses are electromagnetic waves guided by the conductors. The state of the pulses may be calculated by numerically integrating their evolution in space and time from a given *initial state* of voltages and currents, subjected to the *boundary conditions* at either end of the conductor. The voltages u_j and currents i_k change according to the following set of one dimensional partial differential equations called *telegraph equations* that can be read off Fig. 1:

$$\begin{aligned}
\partial_x\, u_1 &= -R_1\,'\, i_1 - L_1\,'\, \partial_t\, i_1 - L_{12}\,'\, \partial_t\, i_2 \\
\partial_x\, u_2 &= -R_2\,'\, i_2 - L_2\,'\, \partial_t\, i_2 - L_{12}\,'\, \partial_t\, i_1 \\
\partial_x\, i_1 &= -C_1\,'\, \partial_t\, u_1 + C_{12}\,'\, \partial_t\, u_{21} - G_1\,'\, u_1 + G_{12}\,'\, u_{21} \\
\partial_x\, i_2 &= -C_2\,'\, \partial_t\, u_2 - C_{12}\,'\, \partial_t\, u_{21} - G_2\,'\, u_2 - G_{12}\,'\, u_{21}
\end{aligned} \qquad (1)$$

with $u_{21} = u_2 - u_1$. For numerical integration these equations and their boundary conditions will be cast into another form that allows the calculation of discrete increments in time (Δt) and space (Δx).

The end of the next section contains a flip page animation of a model pulse travelling back and forth including its behavior at the wire ends. In a second section a method is presented to calculate the capacities and inductances of a given set of linear conductors (including ground planes and dielectrics) by numerically solving the Laplace equation. Code is given that allows precision tests against known geometries. The last section shows then the application of all this to the "pigtail", a part of the ATLAS vertex detector for high

energy particles being assembled at CERN. This is a connection made of microstrip lines on kapton foil carrying signals from the inner part to an optoconverter further outside.

This document is an interactive notebook that may be used to carry out the calculation or to view the animations on a computer. All programming is done in *Mathematica* (**Ref. 8**) – code is shown in grey boxes –, so the input may be easily adapted to get results on other geometries.

Integration of the Telegraph Equations

coupled Telegraph Equations

Rearrangement of the Equations

The scheme in Fig. 1 shows a generic situation of coupling, where everything is reduced to the essentials. It is represented by the set of equations (1). To describe any 'real' setup, it is best to leave Fig. 1 and equations (1) as they are and to project the properties of the situation at hand on to Fig. 1.

Later on the coupling of two differential circuits will be considered, so all the capacitive coupling will have to be expressed through the capacity C_{12}' (45) in Fig. 1. In addition, to simplify matters, we may, as the conductivity of capton is extremely low, put the G matrix safely to zero.

To integrate the telegraph equations the set of equations (1) is rewritten with the time derivative term on the left hand side and currents, voltages and parameters are given in vector and matrix notation (**Ref 2**, p.182f):

$$\begin{aligned} \partial_t \vec{u} &= -[C'^{-1}]\, \partial_x \vec{i} \\ \partial_t \vec{i} &= -[L'^{-1} R']\, \vec{i} - [L'^{-1}]\, \partial_x \vec{u} \end{aligned} \tag{2}$$

with R', L', C' as matrices [ ]:

$$\begin{aligned} [R'] &= \begin{pmatrix} R_1 & 0 \\ 0 & R_2 \end{pmatrix}, \quad [L'] = \begin{pmatrix} L_1 & L_{1,2} \\ L_{1,2} & L_2 \end{pmatrix}, \\ [C'] &= \begin{pmatrix} C_1' + C_{1,2}' & -C_{1,2}' \\ -C_{1,2}' & C_2' + C_{1,2}' \end{pmatrix}, \end{aligned} \tag{3}$$

where the indices point to the pairs of conductors as is evident from Fig. 1. For the following calculations it will be useful to collect $\vec{u}$ and $\vec{i}$ into another state vector $\vec{v}$. Both equations (2) then form (4) a single equation containing the time derivative of $\vec{v}$ on one side and the sum of two matrix operators $\mathcal{L}_1$ and $\mathcal{L}_2$ acting on $\vec{v}$ at the other side :

$$\vec{v} = \begin{pmatrix} \vec{u} \\ \vec{i} \end{pmatrix} \rightarrow$$

$$\partial_t \vec{v} = -\begin{pmatrix} 0 & 0 \\ 0 & [L'^{-1} R'] \end{pmatrix} \vec{v} - \partial_x \begin{pmatrix} 0 & [C'^{-1}] \\ [L'^{-1}] & 0 \end{pmatrix} \vec{v} \tag{4}$$

$$= \mathcal{L}_1(\vec{v}) + \mathcal{L}_2(\vec{v})$$

■ *Mathematica* Code (verify Manipulation of Equations)

If you want to repeat or modify the calculations in the electronic version of this notebook:
select the cell bracket enclosing all grey cells of the corresponding Mathematica Code section and press the evaluate *key*

This cell solves the system of equations (1) for the terms containing the time derivatives:

```
eq1 = r1i1 + l1 di1t + l12 di2t + du1x;
eq2 = r2i2 + l2 di2t + l12 di1t + du2x;
eq3 = c1 du1t - c12 (du2t - du1t) + di1x;
eq4 = c2 du2t + c12 (du2t - du1t) + di2x;
Solve[{eq1 == 0, eq2 == 0, eq3 == 0, eq4 == 0},
  {du1t, du2t, di1t, di2t}] // Simplify
```

$$\left\{\left\{\text{di1t} \to \frac{-\text{du2x l12} + \text{du1x l2} + \text{l2 r1i1} - \text{l12 r2i2}}{\text{l12}^2 - \text{l1 l2}},\right.\right.$$
$$\text{di2t} \to \frac{\text{du2x l1} - \text{du1x l12} - \text{l12 r1i1} + \text{l1 r2i2}}{\text{l12}^2 - \text{l1 l2}},$$
$$\text{du1t} \to -\frac{\text{c2 di1x} + \text{c12 (di1x} + \text{di2x)}}{\text{c12 c2} + \text{c1 (c12} + \text{c2)}},$$
$$\left.\left.\text{du2t} \to -\frac{\text{c1 di2x} + \text{c12 (di1x} + \text{di2x)}}{\text{c12 c2} + \text{c1 (c12} + \text{c2)}}\right\}\right\}$$

The next code example verifies the equality of the first equation of (2) with equations (1):

```
-Inverse[( c1 + c12   -c12
           -c12     c2 + c12 )].{di1x, di2x} // Simplify
```

$$\left\{-\frac{\text{c2 di1x} + \text{c12 (di1x} + \text{di2x)}}{\text{c12 c2} + \text{c1 (c12} + \text{c2)}}, -\frac{\text{c1 di2x} + \text{c12 (di1x} + \text{di2x)}}{\text{c12 c2} + \text{c1 (c12} + \text{c2)}}\right\}$$

It is an easy exercise to check the second equation of (2) in the same way.

■ Discretization of the Telegraph Equations

The state vector $\vec{v}(x,t)$ is represented on a discrete time and space grid as

$$\vec{v}(x_j, t_n) = \vec{v}(j\Delta x, n\Delta t) = \vec{v}_j^{\,n}, \qquad j = 0,..., M, \quad n = 0,..., N, \tag{5}$$

where n (time) and j (space) label the coordinates of the grid and Δx, Δt stand for the grid spacing. The differential operators will be replaced by finite differences, then the state at the time n+1 and at the place j may be expressed by a linear combination of some points at the previous time n. In this way the calculation proceeds in time one Δt after the other. For stability Δt has to be $\leq \Delta x/$(speed of light).

In case the differencing scheme valid for each $\mathcal{L}_i$ *alone* in (4) is known and results in a prescription $\cup_i$ to update $\vec{v}_j^{\,n}$ (in the next two subsections this condition is shown to be fulfilled) :

$$\begin{aligned} &\mathcal{L}_1 : \vec{v}_j^{\,n+1} = \cup_1 (\vec{v}_k^{\,n}, \Delta t) \quad (\text{Damping Term}) \\ &\mathcal{L}_2 : \vec{v}_j^{\,n+1} = \cup_2 (\vec{v}_k^{\,n}, \Delta t) \quad (\text{Wave Equation}), \end{aligned} \tag{6}$$

then the discrete telegraph equations follow with the method of operator splitting (**Ref. 3**, p.847) from (6) by applying the updates one after the other:

$$\vec{v}_j^{\,n+1} = \cup_1 (\cup_2 (\vec{v}_k^{\,n}, \Delta t), \Delta t) \tag{7}$$

■ differencing $\mathcal{L}_2$

The differencing scheme of $\mathcal{L}_2$ is presented in some detail. The time derivative is written as

$$\partial_t \vec{v} = \frac{(\vec{v}_j^{\,n+1} - \vec{v}_j^{\,n})}{\Delta t}, \tag{8}$$

so that with (4)

$$\vec{v}_j^{\,n+1} = \left[\mathbb{1} - \Delta t\, \partial_x \begin{pmatrix} 0 & [C'^{-1}] \\ [L'^{-1}] & 0 \end{pmatrix}\right] \vec{v}_j^{\,n} = [\mathbb{1} - \Delta t\, \partial_x\, [B]]\, \vec{v}_j^{\,n}\ , \tag{9}$$

where $\mathbb{1}$ denotes the identity matrix and [B] is a shorthand for the LC matrix. Next the Lax-Wendroff two-step scheme, which is second order in time, quite accurate and has little distortion if the step size chosen is small enough, will be applied. First the space derivative is written as

$$\partial_x \vec{v} = \frac{(\vec{v}_{j+1}^{\,n} - \vec{v}_j^{\,n})}{\Delta x}, \tag{10}$$

where [B] is considered independent of x, and temporary "half step grid points" are constructed, which (Fig. 2) we put into (9)

$$\begin{aligned} &\frac{\vec{v}_{j+\frac{1}{2}}^{\,n+\frac{1}{2}} - \vec{v}_{j+\frac{1}{2}}^{\,n}}{\frac{\Delta t}{2}} = -[B]\, \frac{(\vec{v}_{j+1}^{\,n} - \vec{v}_j^{\,n})}{\Delta x} \\ &\rightarrow \vec{v}_{j+\frac{1}{2}}^{\,n+\frac{1}{2}} = \vec{v}_{j+\frac{1}{2}}^{\,n} - \frac{\Delta t}{2\,\Delta x}[B]\,(\vec{v}_{j+1}^{\,n} - \vec{v}_j^{\,n}) \end{aligned} \tag{11}$$

and in the same way

$$\vec{v}_{j-\frac{1}{2}}^{\,n+\frac{1}{2}} = \vec{v}_{j-\frac{1}{2}}^{\,n} - \frac{\Delta t}{2\,\Delta x}[B]\,(\vec{v}_j^{\,n} - \vec{v}_{j-1}^{\,n}). \tag{12}$$

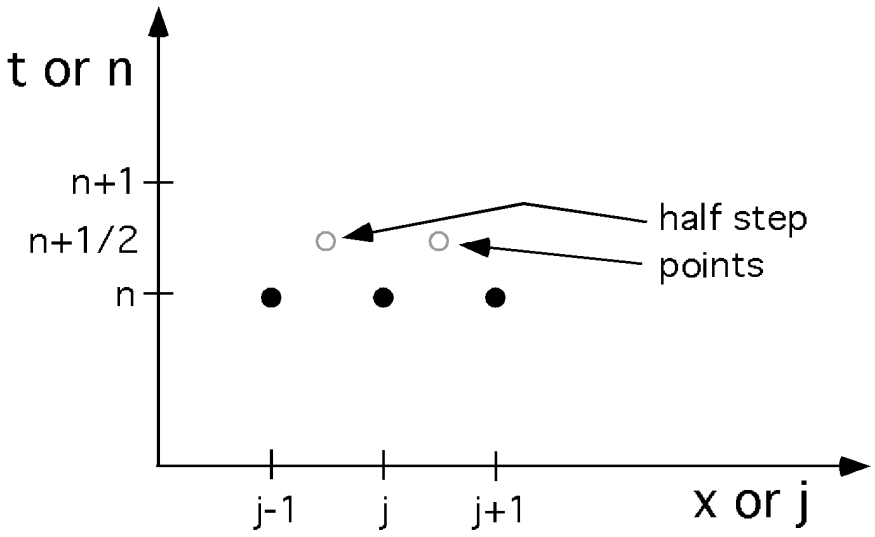


Figure 2

Figure 2. Location of temporary half step points in the Lax-Wendroff Scheme

The $j \pm \frac{1}{2}$ points on the right of (11) and (12) are now replaced by the averages of their left and right neighbors (Lax replacement)

$$v_{j+\frac{1}{2}}^{n} \to \frac{1}{2}\left(v_{j+1}^{n} + v_{j}^{n}\right) \text{ and } v_{j-\frac{1}{2}}^{n} \to \frac{1}{2}\left(v_{j}^{n} + v_{j-1}^{n}\right). \tag{13}$$

In a second step we use the four points connected by the cross in Fig. 3 to calculate v_{j}^{n+1} (**Ref. 3**, p.835):

$$\frac{v_{j}^{n+1} - v_{j}^{n}}{\triangle t} = -[B]\,\frac{\left(v_{j+\frac{1}{2}}^{n+\frac{1}{2}} - v_{j-\frac{1}{2}}^{n+\frac{1}{2}}\right)}{\triangle x}$$

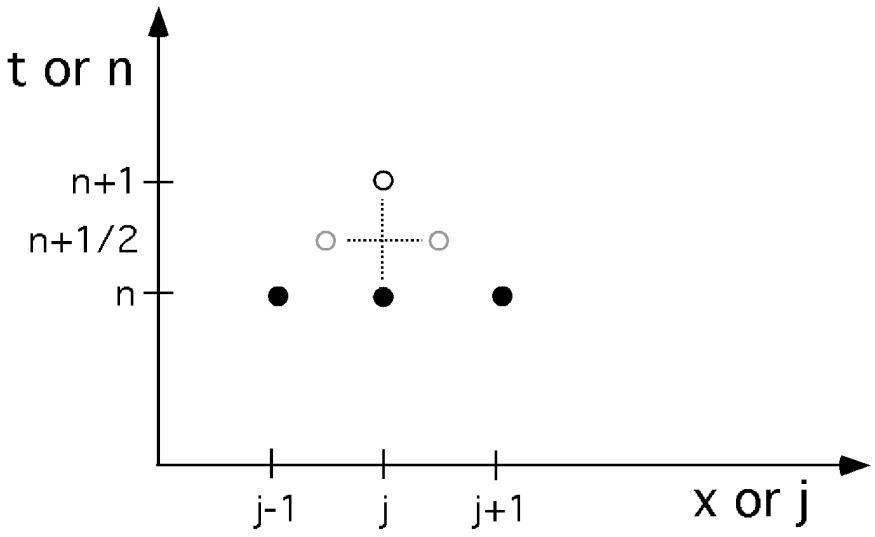


Figure 3

Figure 3. The new point on top is calculated using the staggered leapfrog scheme, it contains information from the three black ones

With (11) and (12) the explicit result for $\cup_2$ is

$$\begin{aligned}
\vec{v}_j^{n+1} &= \cup_2 (\vec{v}_k^n, \Delta t) \\
&= \vec{v}_j^n - \frac{\Delta t}{2\,\Delta x}[B]\left(\vec{v}_{j+1}^n - \vec{v}_{j-1}^n - \frac{\Delta t}{\Delta x}[B](\vec{v}_{j+1}^n - 2\,\vec{v}_j^n + \vec{v}_{j-1}^n)\right) \\
&= \left[\mathbb{1} - \left(\frac{\Delta t}{\Delta x}\right)^2 [B]^2\right]\vec{v}_j^n + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 [B]^2 (\vec{v}_{j+1}^n + \vec{v}_{j-1}^n) - \\
&\quad \frac{\Delta t}{2\,\Delta x}[B](\vec{v}_{j+1}^n - \vec{v}_{j-1}^n)
\end{aligned} \tag{14}$$

- **differencing $\mathcal{L}_1$**

The differencing scheme of $\mathcal{L}_1$ is somewhat simpler, because there is no space derivative. A second shorthand [A] is introduced for the LR matrix:

$$\vec{v}_j^{n+1} = \cup_1 (\vec{v}_k^n, \Delta t) = \left[\mathbb{1} - \Delta t \begin{pmatrix} 0 & 0 \\ 0 & [L'^{-1} R'] \end{pmatrix}\right]\vec{v}_j^n = [A]\,\vec{v}_j^n. \tag{15}$$

Now $\cup_1$ and $\cup_2$ are combined as in (7) :

$$\begin{aligned}
\vec{v}_j^{n+1} &= [A]\left[\mathbb{1} - \left(\tfrac{\Delta t}{\Delta x}\right)^2 [B]^2\right]\vec{v}_j^n + \\
&\quad \tfrac{1}{2}\left(\tfrac{\Delta t}{\Delta x}\right)^2 [A][B]^2 (\vec{v}_{j+1}^n + \vec{v}_{j-1}^n) - \tfrac{\Delta t}{2\,\Delta x}[A][B](\vec{v}_{j+1}^n - \vec{v}_{j-1}^n)
\end{aligned} \tag{16}$$

From (16) we get, through inserting the parameter matrices for [A] and [B] and separating for voltages and currents, at last to the discrete telegraph equations as shown in the next section.

- **the discrete Telegraph Equations**

$$\begin{aligned}
\vec{u}_j^{n+1} &= \left[\mathbb{1} - \left(\frac{\Delta t}{\Delta x}\right)^2 [C'^{-1} L'^{-1}]\right]\vec{u}_j^n + \\
&\quad \frac{1}{2}\left(\left(\frac{\Delta t}{\Delta x}\right)^2 [C'^{-1} L'^{-1}](\vec{u}_{j+1}^n + \vec{u}_{j-1}^n) - \frac{\Delta t}{\Delta x}[C'^{-1}]\left(\vec{i}_{j+1}^n - \vec{i}_{j-1}^n\right)\right) \\
\vec{i}_j^{n+1} &= [\mathbb{1} - \Delta t\,[L'^{-1} R']]\left(\left[\mathbb{1} - \left(\frac{\Delta t}{\Delta x}\right)^2 [L'^{-1} C'^{-1}]\right]\vec{i}_j^n + \right. \\
&\quad \left.\frac{1}{2}\left(\left(\frac{\Delta t}{\Delta x}\right)^2 [L'^{-1} C'^{-1}]\left(\vec{i}_{j+1}^n + \vec{i}_{j-1}^n\right) - \frac{\Delta t}{\Delta x}[L'^{-1}](\vec{u}_{j+1}^n - \vec{u}_{j-1}^n)\right)\right)
\end{aligned} \tag{17}$$

There are now two coupled (voltage/current) grids; Point (n+1, j) is calculated from the three points on the line n below {(n, j+1), (n, j), (n, j-1)} in a left right symmetric way, the computational molecule is an isosceles triangle (cf. Fig. 3).

■ **special treatment of the boundary**

At the edges (j = 1 or M) the outside points j = 0, M+1 are not available, so we have to modify our method. Mean values of differences in space and time (18) are used to calculate the edge points - to understand this look at Fig. 4 . As computational molecule there is now a rectangular box - (**Ref. 4**, p.107):

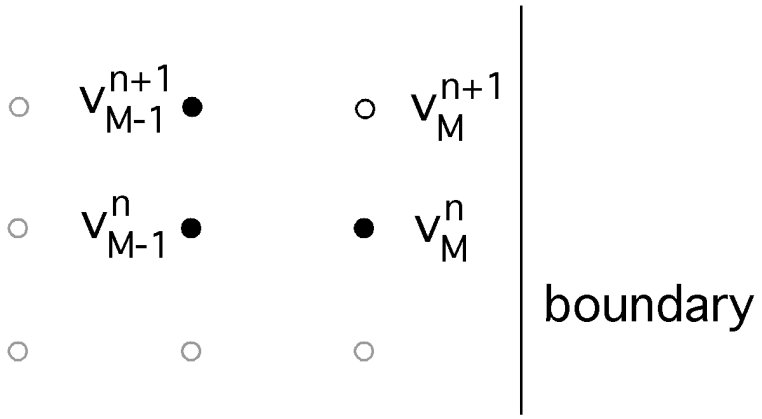


Figure 4

Fig. 4 The unknown point in the upper right edge is calculated from the three black ones by relating average differences in time and space via the matrices [A] and [B].

These calculations are called implicit, because the term to be solved for shows up on the right and the left hand side in the equations (18).

$$\begin{aligned}
&\frac{\vec{v}_M^{n+1}-\vec{v}_M^{n}+\vec{v}_{M-1}^{n+1}-\vec{v}_{M-1}^{n}}{2\,\Delta t} = -[B]\,\frac{\vec{v}_M^{n+1}-\vec{v}_{M-1}^{n+1}+\vec{v}_M^{n}-\vec{v}_{M-1}^{n}}{2\,\Delta x}\\
&\quad\rightarrow \vec{v}_M^{n+1} = \vec{v}_{M-1}^{n} + [\mathbb{I}+\tfrac{\Delta t}{\Delta x}[B]]^{-1}[\mathbb{I}-\tfrac{\Delta t}{\Delta x}[B]]\left(\vec{v}_M^{n}-\vec{v}_{M-1}^{n+1}\right)\\
&\frac{\vec{v}_1^{n+1}-\vec{v}_1^{n}+\vec{v}_2^{n+1}-\vec{v}_2^{n}}{2\,\Delta t} = -[B]\,\frac{\vec{v}_2^{n+1}-\vec{v}_1^{n+1}+\vec{v}_2^{n}-\vec{v}_1^{n}}{2\,\Delta x}\\
&\quad\rightarrow \vec{v}_1^{n+1} = \vec{v}_2^{n} + [\mathbb{I}-\tfrac{\Delta t}{\Delta x}[B]]^{-1}[\mathbb{I}+\tfrac{\Delta t}{\Delta x}[B]]\left(\vec{v}_1^{n}-\vec{v}_2^{n+1}\right)
\end{aligned} \tag{18}$$

After formation of the inverse matrix and multiplication with [A] we get these expressions:
right side:

$$\begin{aligned}
\vec{u}_M^{n+1} &= \vec{u}_{M-1}^{n} + \left[\mathbb{I}-(\tfrac{\Delta t}{\Delta x})^2[C'^{-1}][L'^{-1}]\right]^{-1}\left[\mathbb{I}+(\tfrac{\Delta t}{\Delta x})^2[C'^{-1}][L'^{-1}]\right]\left(\vec{u}_M^{n}-\vec{u}_{M-1}^{n+1}\right)-\\
&\quad 2\,\tfrac{\Delta t}{\Delta x}\left[\mathbb{I}-(\tfrac{\Delta t}{\Delta x})^2[C'^{-1}][L'^{-1}]\right]^{-1}[C'^{-1}]\left(\vec{i}_M^{n}-\vec{i}_{M-1}^{n+1}\right),\\
\vec{i}_M^{n+1} &= [\mathbb{I}-\Delta t[L'^{-1}\,R']]\left(\vec{i}_{M-1}^{n}-2\,\tfrac{\Delta t}{\Delta x}\left[\mathbb{I}-(\tfrac{\Delta t}{\Delta x})^2[L'^{-1}][C'^{-1}]\right]^{-1}[L'^{-1}]\left(\vec{u}_M^{n}-\vec{u}_{M-1}^{n+1}\right)+\right.\\
&\quad \left.\left[\mathbb{I}-(\tfrac{\Delta t}{\Delta x})^2[L'^{-1}][C'^{-1}]\right]^{-1}\left[\mathbb{I}+(\tfrac{\Delta t}{\Delta x})^2[L'^{-1}][C'^{-1}]\right]\left(\vec{i}_M^{n}-\vec{i}_{M-1}^{n+1}\right)\right)
\end{aligned} \tag{19}$$

and left side:

$$\begin{aligned}
\vec{u}_1^{n+1} &= \vec{u}_2^{n} + \left[\mathbb{I}-(\tfrac{\Delta t}{\Delta x})^2[C'^{-1}][L'^{-1}]\right]^{-1}\left[\mathbb{I}+(\tfrac{\Delta t}{\Delta x})^2[C'^{-1}][L'^{-1}]\right]\left(\vec{u}_1^{n}-\vec{u}_2^{n+1}\right)+\\
&\quad 2\,\tfrac{\Delta t}{\Delta x}\left[\mathbb{I}-(\tfrac{\Delta t}{\Delta x})^2[C'^{-1}][L'^{-1}]\right]^{-1}[C'^{-1}]\left(\vec{i}_1^{n}-\vec{i}_2^{n+1}\right),\\
\vec{i}_1^{n+1} &= [\mathbb{I}-\Delta t[L'^{-1}\,R']]\left(\vec{i}_2^{n}+2\,\tfrac{\Delta t}{\Delta x}\left[\mathbb{I}-(\tfrac{\Delta t}{\Delta x})^2[L'^{-1}][C'^{-1}]\right]^{-1}[L'^{-1}]\left(\vec{u}_1^{n}-\vec{u}_2^{n+1}\right)+\right.\\
&\quad \left.\left[\mathbb{I}-(\tfrac{\Delta t}{\Delta x})^2[L'^{-1}][C'^{-1}]\right]^{-1}\left[\mathbb{I}+(\tfrac{\Delta t}{\Delta x})^2[L'^{-1}][C'^{-1}]\right]\left(\vec{i}_1^{n}-\vec{i}_2^{n+1}\right)\right)
\end{aligned} \tag{20}$$

- **explicit boundary conditions**

Now the boundary conditions may be specified in the following ways:
open → set i = 0, leave u as in (19) or (20);
short → set u = 0, leave i as in (19) or (20);
termination with z right/left → set u = i * (+/- z), leave i as in (19) / (20).

Mathematica Code (Gaussian Pulse with Reflections)

RAM requirements: Front End 40 MB, Kernel 10 MB.

As an example the propagation of gaussian pulses placed in the middle of each of two wires is presented. In order to demonstrate clearly the propagation and reflection properties the mutual coupling, which disturbs the shape of the pulses, is set to zero.

- **Definitions**

```
npl = 100;(* number of plots,
 may be set to smaller number if memory is tight *)
dt = 1. * 10^-12; (* 1 psec *)
dx = 0.004; (* 4 mm *)
(* dx/dt >> speed of propagation for stability,
 trade-off with computational speed *)
μ0 = 4. π * 10^-7 ;
ϵ0 = 10^-9 / (35.95 π);
m = 250; (* array dimension → space distance m*dx = 1 m *)
nmax = 4000;
(* simulation time steps, time considered: nmax * dt  *)
pl = Round[nmax / npl];
(* snapshot after every pl timesteps *)
k = 2;(* 2 wires → dimension of matrices *)
```

- **Parameters per Unit Length**

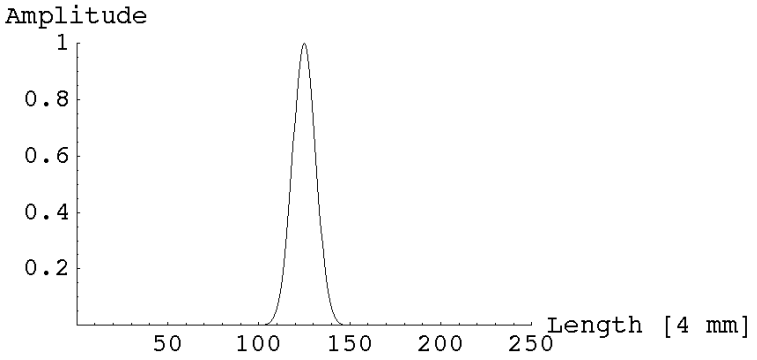
```
res = 2.;  (* resistance in Ω per meter *)
rm = {{res, 0}, {0, res}};
cap = ϵ0 * π / 2; (* capacitance in F per meter  *)
cm = {{cap, 0}, {0, cap}}; (* no mutual coupling *)
ci = Inverse[cm];
ind = 2 * μ0 / π; (* inductance in H per meter  *)
lm = {{ind, 0}, {0, ind}}; (* no mutual coupling *)
li = Inverse[lm];
zl = 0.5 Sqrt[ind / cap]; (* too low Termination left  *)
zr = 2.0 Sqrt[ind / cap];(* too high Termination right  *)
```

- **Initialization and Setup of Coefficient Matrices**

```
id = IdentityMatrix[k];
u1 = u0 = Table[0., {m}, {k}]; i1 = i0 = Table[0., {m}, {k}];
m1 = id - (dt/dx)^2 ci.li; m2 = 1/2 (dt/dx)^2 (ci.li); m3 = -1/2 (dt/dx) ci;
m4 = (id - dt li.rm).(id - (dt/dx)^2 li.ci);
m5 = 1/2 (dt/dx)^2 (id - dt li.rm).(li.ci);
m6 = -1/2 (dt/dx) (id - dt li.rm).li;
m8 = Inverse[id - (dt/dx)^2 ci.li].(id + (dt/dx)^2 ci.li);
m9 = 2 dt/dx Inverse[id - (dt/dx)^2 ci.li].ci; m10 = id - dt li.rm;
m11 = 2 dt/dx (id - dt li.rm).Inverse[id - (dt/dx)^2 li.ci].li;
m12 =
  (id - dt li.rm).Inverse[id - (dt/dx)^2 li.ci].(id + (dt/dx)^2 li.ci);
```

- **Static Initial Condition**

```
u0 = Transpose[u0];
u0[[1]] = u0[[2]] = Table[ Exp[-(i - 125)^2 / 80], {i, 1, m}];
u0 = Chop[Transpose[u0]];
p1 = ListPlot[Transpose[u0][[1]],
   PlotRange → {{0, m}, {0, 1}}, PlotJoined → True,
   AxesLabel -> {"Length [4 mm]", "Amplitude"}];
```



A gaussian pulse is placed in the middle of the wires, it is going to divide into two separate components.

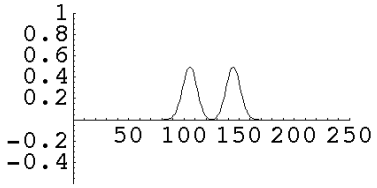
- **Calculate Evolution in Time and produce Plots**

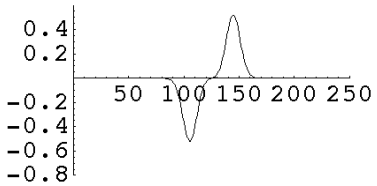
The output of the next cell shows the pulses at progressing time intervals. If their enclosing cell bracket is selected and the Menu Command "Animate selected Graphics" is chosen, they display an animation.

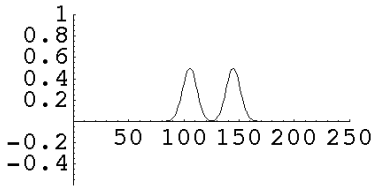
```
(* this cell produces npl plots, view as animation *)
Timing[
 Do[
  {(* main loop, calculate line n+1 in u and i grid *)
   Do[{jp1 = j + 1; jm1 = j - 1;
     u1[[j]] = m1.u0[[j]] +
       m2.(u0[[jp1]] + u0[[jm1]]) + m3.(i0[[jp1]] - i0[[jm1]]);
     i1[[j]] = m4.i0[[j]] + m5.(i0[[jp1]] + i0[[jm1]]) +
       m6.(u0[[jp1]] - u0[[jm1]])}, {j, 2, m - 1}];
   (* edges *)
   u1[[m]] = u0[[m - 1]] +
     m8.(u0[[m]] - u1[[m - 1]]) - m9.(i0[[m]] - i1[[m - 1]]);
   i1[[m]] = m10.i0[[m - 1]] - m11.(u0[[m]] - u1[[m - 1]]) +
     m12.(i0[[m]] - i1[[m - 1]]);
   u1[[1]] = u0[[2]] + m8.(u0[[1]] - u1[[2]]) +
     m9.(i0[[1]] - i1[[2]]);
   i1[[1]] = m10.i0[[2]] + m11.(u0[[1]] - u1[[2]]) +
     m12.(i0[[1]] - i1[[2]]);
   (* static boundary conditions  open: set i=0,
    short: set u=0 *)
   u1[[m, 2]] = zr i1[[m, 2]]; (* termination right with z *)
   u1[[1, 2]] = -zl i1[[1, 2]];
   (* termination left with -z *)
   u1[[1, 1]] = 0.; (* wire 1 left short circuit *)
   i1[[m, 1]] = 0.; (* wire 1 right open end *)
   u0 = u1; i0 = i1; (* advance one step in time *)
   (* do a snapshot of the voltages *)
   If[Mod[n, pl] == 0,
    {(* plot voltage wire 1 *)
     p1 = ListPlot[
       Transpose[u1][[1]], PlotRange → {{0, m}, {-0.6, 1}},
       PlotJoined → True, DisplayFunction → Identity];
     (* plot current wire 1 *)
     p2 = ListPlot[250 Transpose[i1][[1]],
       PlotRange → {{0, m}, {-0.8, 0.6}},
       PlotJoined → True, DisplayFunction → Identity];
     (* plot voltage wire 2 *)
     p3 = ListPlot[
       Transpose[u1][[2]], PlotRange → {{0, m}, {-0.6, 1}},
       PlotJoined → True, DisplayFunction → Identity];
     Show[GraphicsArray[{{p1}, {p2}, {p3}}],
      DisplayFunction → $DisplayFunction];
     (* get rid of small numbers to
       accelerate calculation *)
     u0 = Chop[u0]; i0 = Chop[i0]; Print[n]}]}, {n, nmax}]
]
Print["Fertig !"]
```

■ 3 selected Plots from the Animation

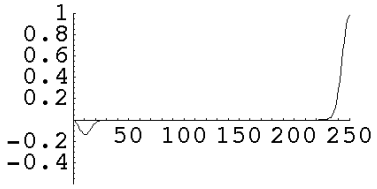
Two wires without coupling are simulated, each of which is carrying a left and right travelling component of the initial pulse; the first two diagrams show u and i of the same, namely the first wire, the third diagram displays the voltage of the second wire. The current is scaled up by a factor that it numerically gets roughly the same size as the voltage. The pulses have broken up in two parts that run apart. Edges of pulses with gradients of voltages and currents of equal sign move right, those with gradients of opposite sign move left.

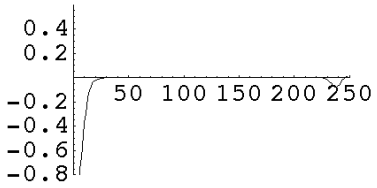


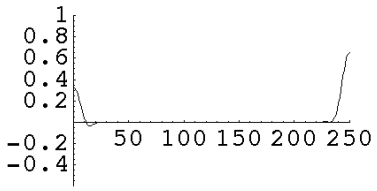




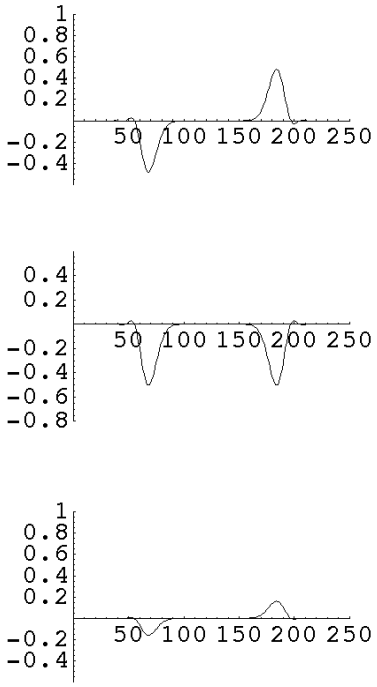
The upper wire has a short at the left and is open at the right end, the second wire (diagram at the bottom) is terminated left with $Z_l = Z_0/2$ and right with $Z_r = 2\,Z_0$, where Z_0 is the impedance $\sqrt{L/C}$ of the wire. These boundary conditions cause reflection and transmission at either end of the wires.







Now the reflected pulses have changed direction, the amplitude of the pulses on the second wire is reduced to $(Z_x - Z_0) / (Z_x + Z_0)$ of the original.



The simulated pulses demonstrate the expected behaviour including reflection at the wire boundaries. To understand any real situation the parameters entering the telegraph equation, specifically the capacities and inductances of the configuration of conductors should be known precisely (the calculations of the next sections reach an accuracy of a few % – tested with coax and other geometries –).

Numerical Calculation of Capacities and Inductances

■ Capacities

■ Definition of Capacitance, Coefficients of Induction and Coefficients of Capacity

In a system of n conductors the equations relating the 'induced' charge on one conductor to the potentials (U or ϕ) of all others (**Ref. 5**, p.318) read

$$\begin{aligned} Q_1 &= \tilde{C}_{11} * U_1 + \tilde{C}_{12} * U_2 + ... + \tilde{C}_{1n} * U_n \\ Q_2 &= \tilde{C}_{21} * U_1 + \tilde{C}_{22} * U_2 + ... + \tilde{C}_{2n} * U_n \\ &\vdots \\ Q_n &= \tilde{C}_{n1} * U_1 + \tilde{C}_{n2} * U_2 + ... + \tilde{C}_{nn} * U_n , \end{aligned} \tag{21}$$

where the $\tilde{C}_{ij} = \tilde{C}_{ji}$ (i≠j) are called '*coefficients of induction*', $\tilde{C}_{ii}$ is the *capacitance* of conductor i.
The values of the $\tilde{C}$ are constants depending on geometry *only*, that means on the shapes and the arrangement of the conductors. Add

$$0 = \tilde{C}_{i1} * U_i + \tilde{C}_{i2} * U_i + ... + \tilde{C}_{i,i-1} * U_i + \tilde{C}_{i,i+1} * U_i + ... + \tilde{C}_{in} * U_i - \tilde{C}_{i1} * U_i - ... - \tilde{C}_{in} * U_i \tag{22}$$

to equation i of (21), and they can be reshuffled to contain Maxwells 'coefficients of capacity' (*Teilkapazitäten*):

$$\begin{aligned} Q_1 &= C_{10} * U_1 + C_{12} * (U_1 - U_2) + ... + C_{1n} * (U_1 - U_n) \\ Q_2 &= C_{21} * (U_2 - U_1) + C_{20} * U_2 + ... + C_{2n} * (U_2 - U_n) \\ &\vdots \\ Q_n &= C_{n1} * (U_n - U_1) + C_{n2} * (U_n - U_2) + ... + C_{n0} * U_n . \end{aligned} \tag{23}$$

The coefficients of capacity are connected to the coefficients of induction through the following equations:

$$\begin{aligned} C_{i0} &= \sum_{k=1}^{n} \tilde{C}_{ik} \\ C_{ik} &= - \tilde{C}_{ik} \qquad (i \neq k). \end{aligned} \tag{24}$$

C_{i0} is known as 'stray capacity', the part that corresponds to the field that goes to the 'outside', where the fieldlines connect to the surrounding ground. Now the meaning of the term 'capacitance' defined above also becomes clear: the capacitance of conductor i is the sum of its stray capacity plus all its coefficients of capacity with the other conductors of the system.

The procedure outlined below finds the $\tilde{C}_{ik}$ of a given configuration. We then use (24) to calculate the coefficients of capacity, a combination of which (according to the mode of operation) is to be put into (3).

■ Determination of the Coefficients of Induction in a System of Linear Conductors

Consider a system of conductors within a volume V filled with dielectric material ($\epsilon_r \neq 1$) and/or empty space. The potential energy of the conductors, being fixed at certain potentials ϕ_i, may then be written as

$$W = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \tilde{C}_{ij} \phi_i \phi_j . \tag{25}$$

The induced charges will produce an electric field, whose shape is a consequence of the given geometry, and whose energy amounts to

$$W = \frac{1}{2} \int_V \vec{D} \cdot \vec{E} \, dV . \tag{26}$$

The energy of a charged system of conductors is thus stored in the electric field between them. This offers a neat way to calculate the capacitances and coefficients of induction of the configuration from the electric field

without quantitative information about the charge distribution, – once the field is known (**Ref. 6**, p.53). The capacitance per unit length $\tilde{C}_{ii}'$ of the linear conductor i pops out of (25) and (26), if we integrate over the plane perpendicular to the conductor the electric field corresponding to the boundary conditions ϕ_i =1 and all other ϕ_k = 0 :

$$\tilde{C}_{ii}' = \int_A \epsilon_r \, \epsilon_0 \, \vec{E}_i \cdot \vec{E}_i \, dx \, dy \tag{27}$$

After the capacitances the coefficients of induction may be obtained from

$$\tilde{C}_{ij}' = \frac{1}{2} \left(\int_A \epsilon_r \, \epsilon_0 \, \vec{E}_{ij} \cdot \vec{E}_{ij} \, dx \, dy - \tilde{C}_{ii}' - \tilde{C}_{jj}' \right), \tag{28}$$

by considering pairs of conductors, where the indices of $\vec{E}$ point to the two conductors, whose potentials are then set to 1. From (27) and (28) it is clear that the presence of any additional conductor will change the field and will consequently have an effect on every other capacity.

■ Discrete Form of Gauss's Law with Dielectrics

It remains to determine the electrical field. In the volume outside the conductors there are no free charges, here the field obeys this form of Gauss's Law

$$\vec{\nabla} \cdot \vec{D} = \vec{\nabla} \cdot \epsilon \vec{E} = 0 \tag{29}$$

with $\epsilon = \epsilon_r \, \epsilon_0$. As the field is the negative gradient of the potential ϕ, (29) reads with a space dependent ϵ(x,y)

$$\vec{\nabla} \cdot \epsilon \vec{E} = - (\vec{\nabla} \epsilon \cdot \vec{\nabla} \phi + \epsilon \Delta \phi) = 0. \tag{30}$$

The terms containing the first derivatives act as additional sources to the Laplace equation that appears in (30).

The potential ϕ(x,y) is mapped onto a grid with $u_{i,j} = \phi$(i Δx, j Δy). In the iterative relaxation process used for elliptic pde's (**Ref. 3**, p.821) the Laplace operator Δ is discretized by the prescription that each point is updated with the average of its four neighbors. $\epsilon_{i,j}$ is either 1 (for vacuum or air) or equal to the relative dielectric constant of the material at the location of the corresponding grid point.

Discretize first derivatives like (to second order for symmetry reasons, – first order would be shifted to up and right) :

$$\partial_x f_{i,j} = \frac{f_{i,j+1} - f_{i,j-1}}{2 \Delta}, \tag{31}$$

that means, any structure, in order to be correctly included, must comprise at least two grid points. This is the 'resolution' of this simulation.

The following update procedure is applied many times to each grid point starting from actually any initial matrix $u_{i,j}$, until the result no longer changes:

$$u_{i,j} = \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{4} + \frac{\epsilon_{i,j+1} - \epsilon_{i,j-1}}{16 \, \epsilon_{i,j}} (u_{i,j+1} - u_{i,j-1}) + \frac{\epsilon_{i+1,j} - \epsilon_{i-1,j}}{16 \, \epsilon_{i,j}} (u_{i+1,j} - u_{i-1,j}) \tag{32}$$

If the values at the boundaries are completely specified, a unique solution for u will be found slowly propagating into the plane from the fixed boundaries. The number of iterations and hence the calculation time is proportional to the difference of the arbitrary initial matrix and the final result. The distribution of the $u_{i,j}$, the relaxation converges to, is characterized by the boundary values and the Laplace operator, which enforces the sum of the curvatures (∼ to second derivative in space) in x and y direction to be zero. That means, either both are zero (a piece of a plane surface) or both have opposite sign and equal size (saddle form). The presence of dielectrics reveals itself by a sudden jump of the gradient to lower values (see code example below).

■ *Mathematica* Code (Parallel Plate Capacitor)

RAM requirements: Front End 10 MB, Kernel 10 MB

A grid of 80 x 40 points is set up. In all of the following we have to keep in mind, how a grid matrix is mapped onto a rectangular coordinate system: the column index j marks the x-axis, increasing row numbers i proceed in negative y-direction.

Two conducting planes forming a capacitor are placed within a grounded rectangle, between the planes a dielectric area is defined.

■ set up matrix:

m defines a matrix of operators that tell, how the point i,j on the grid is to be updated (32), v1 and v2 define the fixed potential on conductor 1and 2 (the warnings on Part specification are harmless...).

```
v1=1.;v2=0.;(* potential of the electrodes 1 and 2 *)
nz=40; (* # of rows of the grid, divisible by 2 *)
ns=80; (* # of columns of the grid, divisible by 2 *)
u=.;m=.;m=Array[u,{nz,ns}];a=Array[u,{nz,ns}];
mb=Table[0,{nz},{ns}];
ϵ=Table[1,{i,1,nz},{j,1,ns}];
eps=4;
exl=20;exr=39;eyd=20;eyu=28; (* dielectric area *)
Do[Do[ϵ[[i,j]]=eps,{i,eyd,eyu}],{j,exl,exr}];
Do[Do[a[[i,j]]=u[[i,j]],{j,1,ns}],{i,1,nz}]
Do[Do[
m[[i,j]]:=Evaluate[0.25(a[[i-1,j]]+a[[i,j-1]]+
a[[i+1,j]]+a[[i,j+1]])+0.0625/ϵ[[i,j]]((ϵ[[i,j+1]]-
ϵ[[i,j-1]])(a[[i,j+1]]-a[[i,j-1]])+(ϵ[[i+1,j]]-
ϵ[[i-1,j]])(a[[i+1,j]]-a[[i-1,j]]))],
{j,2,ns-1}],{i,2,nz-1}]
```

– *Part::partd : Part specification u⟦1, 1⟧ is longer than depth of object.*

– *Part::partd : Part specification u⟦1, 1⟧ is longer than depth of object.*

– *Part::partd : Part specification u⟦1, 2⟧ is longer than depth of object.*

– *General::stop :*
Further output of Part::partd will be suppressed during this calculation.

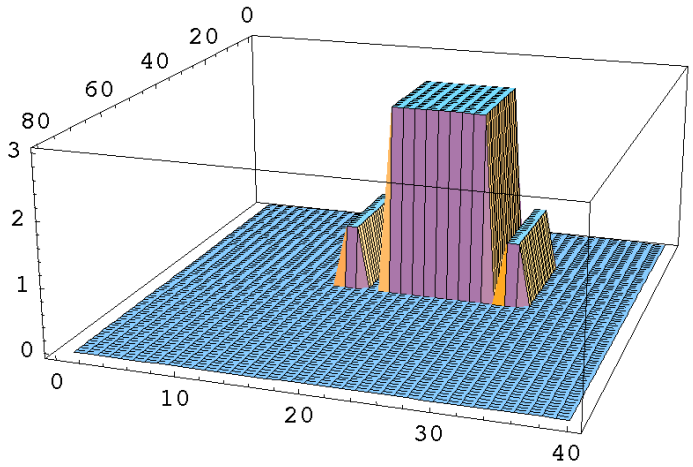
■ boundary conditions:

```
Do[m[[1, j]] := 0., {j, 1, ns}]
Do[m[[nz, j]] := 0., {j, 1, ns}]
Do[m[[i, 1]] := 0., {i, 1, nz}]
Do[m[[i, ns]] := 0., {i, 1, nz}]
```

Set location of the conductors in the operator matrix, also stored in mask mb :

```
xl1=20;xr1=39;yd1=16;yu1=17;xl2=20;xr2=39;yd2=30;yu2=31;
Do[Do[{m[[i,j]]:=v1;mb[[i,j]]=1;},{j,xl1,xr1}],{i,yd1,yu1}]
Do[Do[{m[[i,j]]:=v2;mb[[i,j]]=1;},{j,xl2,xr2}],{i,yd2,yu2}]
```

```
p1 = ListPlot3D[ϵ - 1,
   ViewPoint → {3., 1., 1.}, DisplayFunction → Identity];
p2 = ListPlot3D[mb, ViewPoint → {3., 1., 1.},
   DisplayFunction → Identity];
Show[p1, p2, DisplayFunction → $DisplayFunction]
```



- Graphics3D -

This picture shows the location of the dielectrics and of the conductors in the plane.

■ **calculate potential**

To accelerate the calculation we obtain a rough guess u1 (without dielectrics) on a coarser grid (# points/4 -> faster) as initial matrix. u1 has already some similarity to the true distribution:

```
nz1=nz/2;ns1=ns/2;u1=.;m1=.;
m1=Array[u1,{nz1,ns1}];a1=Array[u1,{nz1,ns1}];
Do[Do[a1[[i,j]]=u1[[i,j]],{j,1,ns1}],{i,1,nz1}]
Do[Do[
m1[[i,j]]:=Evaluate[0.25(a1[[i-1,j]]+a1[[i+1,j]]+a1[[i,j-1]]+
a1[[i,j+1]])],
{j,2,ns1-1}],{i,2,nz1-1}];
Do[m1[[1,j]]:=0.,{j,1,ns1}];
Do[m1[[nz1,j]]:=0.,{j,1,ns1}];
Do[m1[[i,1]]:=0.,{i,1,nz1}];
Do[m1[[i,ns1]]:=0.,{i,1,nz1}];
(* guess for coordinates:old/2;
mb is used to identify location of conductors*)
Do[Do[If[mb[[2i,2j]]==1,m1[[i,j]]=m[[2i,2j]]],{i,nz1}],
{j,ns1}]
```

— *Part::partd : Part specification u1⟦1, 1⟧ is longer than depth of object.*

— *Part::partd : Part specification u1⟦1, 2⟧ is longer than depth of object.*

— *Part::partd : Part specification u1⟦1, 3⟧ is longer than depth of object.*

— *General::stop :*
Further output of Part::partd will be suppressed during this calculation.

Let Laplace do his work...

```
u1=Table[0.,{i,1,nz1},{j,1,ns1}];
Timing[Do[{u1=m1;
If[Mod[i,500]==0,
Print[i," residual error:",Max[Flatten[Abs[u1-m1]]]]]},
{i,1,1000}]]
```

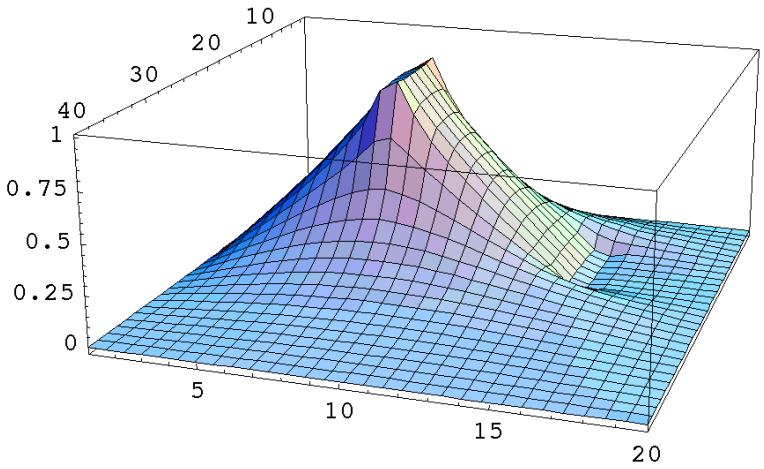
```
500 residual error:8.45628×10^-6

1000 residual error:2.25683×10^-8

{22. Second, Null}
```

The coarse grid u1 is interpolated to calculate intermediate points
for the final higher resolution grid u (j -> x, i -> y)

```
p = ListInterpolation[Transpose[u1]];
```

Control plot of interpolation:

```
Plot3D[p[x, y], {x, 1, ns1}, {y, 1, nz1},
 PlotRange → All, PlotPoints → 30, ViewPoint → {3., 1., 1.}]
```



Control plot of interpolated potential.

Copy coarse to fine grid as input for the main calculation
(near the border the interpolation has to extrapolate a little):

```
u=Table[p[j/2,i/2],{i,1,nz},{j,1,ns}];
```

– *InterpolatingFunction::dmval :*
Input value $\{\frac{1}{2}, \frac{1}{2}\}$ *lies outside the range of data*
in the interpolating function. Extrapolation will be used.
– *General::stop : Further output of*
InterpolatingFunction::dmval will be suppressed during this calculation.

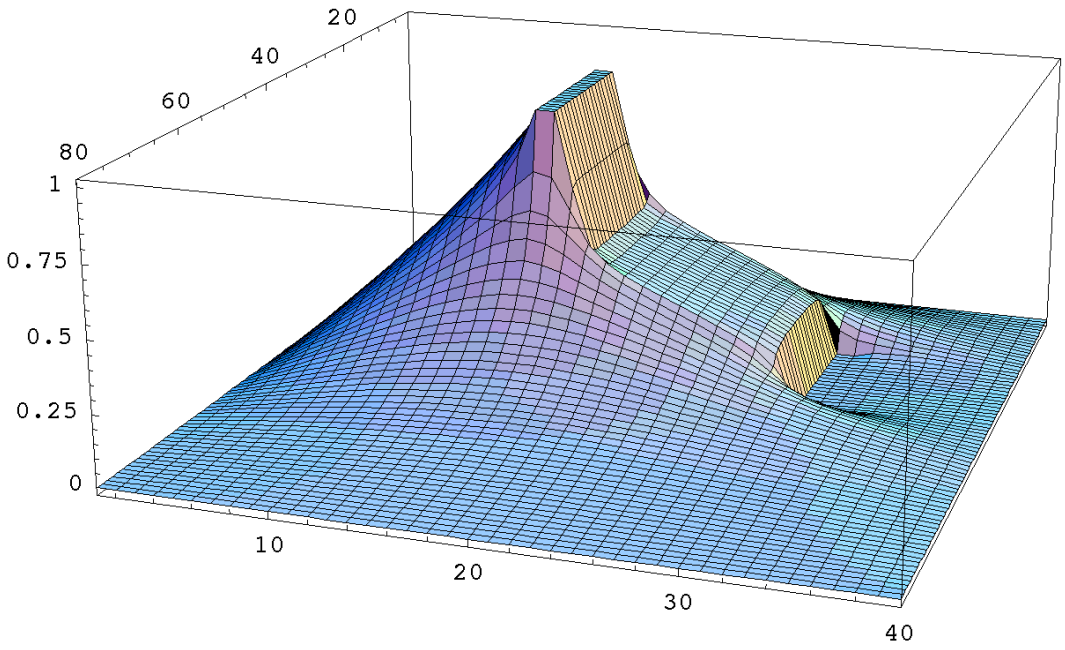
Once again time for Laplace...

```
Timing[Do[{u=m;
If[Mod[i,100]==0,{err=Max[Abs[Flatten[u-m]]];
Print[i," residual error: ",err];
If[err<10^-6,Break[]]}]},{i,1,2000}]]
```

1200 residual error: 7.76124×10^{-7}

{120.533 Second, Null}

```
ListPlot3D[u,
ImageSize->400,PlotRange->All,ViewPoint→{3.,1.,1.}]
```



- SurfaceGraphics -

View of the resulting potential including dielectric material. The dielectric effects on the potential are evident in the abrupt change of the gradient on the flank to the right of the peak.

■ **contourplot with fieldlines**

A contourplot is generated for later display:

```
cont = ListContourPlot[u, ImageSize → 400,
  PlotRange → All, ContourShading → False,
  Contours → 15, DisplayFunction → Identity]
```

```
- ContourGraphics -
```

The electric field vector is given as derivative of an interpolated function of the potential u :

```
wi = ListInterpolation[Transpose[u], InterpolationOrder → 1];
(* column index counts increments in x direction,
 row index counts increments in y direction →
  use Transpose *)
ex = Derivative[1, 0][wi];
ey = Derivative[0, 1][wi];
```

The lines follow the field from the starting point x0, y0 uphill and downhill as far as possible...

```
FieldLine[{ex_InterpolatingFunction, x0_},
  {ey_InterpolatingFunction, y0_}] :=
 Module[{x = x0, y = y0, l = {{x0, y0}},
   xxmin = Part[ex, 1, 1, 1], xxmax = Part[ex, 1, 1, 2],
   yymin = Part[ex, 1, 2, 1],
   yymax = Part[ex, 1, 2, 2], emod10, xn, yn},
  (* search uphill *)
  Do[{emod10 = 10. √((ex[x, y])^2 + (ey[x, y])^2);
    If[emod10 == 0., Break[]];
    xn = ex[x, y] / emod10 + x; yn = ey[x, y] / emod10 + y;
    If[xn < xxmin || xn > xxmax ||
      yn < yymin || yn > yymax, Break[]];
    l = Append[l, {xn, yn}]; x = xn; y = yn}, {n, 1, 5000}];
  x = x0; y = y0;
  (* search downhill *)
  Do[{emod10 = -10. √((ex[x, y])^2 + (ey[x, y])^2);
    If[emod10 == 0., Break[]];
    xn = ex[x, y] / emod10 + x; yn = ey[x, y] / emod10 + y;
    If[xn < xxmin || xn > xxmax ||
      yn < yymin || yn > yymax, Break[]];
    l = Prepend[l, {xn, yn}]; x = xn; y = yn}, {n, 1, 5000}];
  Line[l]]
```

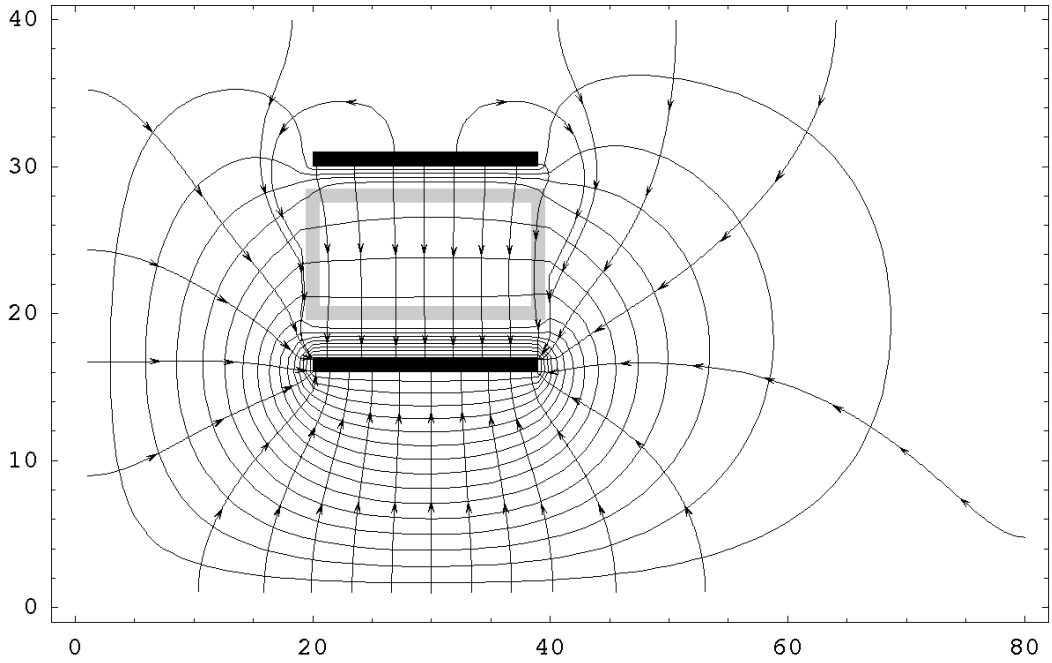
```
Needs["Graphics`Arrow`"]
```

Add directional arrows to the fieldlines (**Ref. 9**, p. 522):

```
AddArrow[Line[opts_], d_, num_: 8] :=
 Module[{arr = {}, n = 0, pts = Chop[opts]},
  Fold[
   If[First[#1] ≥ d && n < num, n++;
     AppendTo[arr, Arrow[Last[#1], #2, HeadScaling → Absolute,
       HeadCenter → 0.5, HeadLength → 4]];
     {0, #2}, {First[#1] + Sqrt[Apply[Plus, (Last[#1] - #2)^2]],
      #2}] &, {0, First[pts]}, Rest[pts]];
  arr]
```

We let n fieldlines run up and down from an ellipse with center xe, ye and half axes a, b. The density of the field lines is weighted with the strength of the field at the starting points. The dielectric area is indicated with a grey line. You have to experiment with the parameters to get a decent picture...

```
{xe, ye} = {30, 16.5}; a = 14; b = 10;
n = 21;(* n/3 is the number of angular bins *)
eps = ListInterpolation[Transpose[ϵ]];
(* sample the field strength on the ellipse *)
ft = Table[eps[a Cos[i 2π / n] + xe, b Sin[i 2π / n] + ye]
    √(ex[a Cos[i 2π / n] + xe, b Sin[i 2π / n] + ye]^2 +
       ey[a Cos[i 2π / n] + xe, b Sin[i 2π / n] + ye]^2), {i, 1, n}];
s = Apply[Plus, ft];
(* nl[[i]] is (number of lines - 1)
  starting in the ith angular bin of the ellipse *)
nl = Round[Apply[Plus, Transpose[Partition[ft, 3]]] / s * n];
start = {};
(* array of angles of the
  starting points on the ellipse *)
Do[Do[AppendTo[start, (2π / (n / 3)) ((i - 1) + j / (nl[[i]] + 1))],
   {j, 0, nl[[i]]}], {i, 1, n / 3}];
(* calculate lines one after the other *)
lines = Table[FieldLine[{ex, a Cos[start[[i]]] + xe},
    {ey, b Sin[start[[i]]] + ye}], {i, 1, Length[start]}];
(* add single lines where appropriate *)
line1 = FieldLine[{ex, 25}, {ey, 34}]; AppendTo[lines, line1];
line2 = FieldLine[{ex, 34}, {ey, 34}]; AppendTo[lines, line2];
arrows = Map[AddArrow[#, 6] &, lines];
(* r1,r2 : electrodes, r3: dielectrics *)
r1 = Rectangle[{xl1, yd1}, {xr1, yu1}];
r2 = Rectangle[{xl2, yd2}, {xr2, yu2}];
r3 = {Line[{{exl, eyd}, {exl, eyu}}],
  Line[{{exl, eyu}, {exr, eyu}}], Line[
   {{exr, eyu}, {exr, eyd}}], Line[{{exr, eyd}, {exl, eyd}}]};
Show[Graphics[{GrayLevel[0.8], Thickness[0.015], r3}],
 cont, Graphics[{lines, arrows, r1, r2}], ImageSize → 400,
 Frame → True, DisplayFunction → $DisplayFunction]
```



- Graphics -

The above picture shows the field and the equipotential lines of two electrodes, the upper one at zero, the lower at 1. All lines display a kink, whenever they enter or leave the dielectric (the greyish bordered area).

- **gradient field...**

The matrix of gradient vectors needs a complicated build: the last row has same the y component as the row above, the last column has the same x component as the column to its left:

```
g = Table[{u[[i, j + 1]] - u[[i, j]], u[[i + 1, j]] - u[[i, j]]},
   {i, 1, nz - 1}, {j, 1, ns - 1}];
lrow = Table[{u[[nz, j + 1]] - u[[nz, j]],
    u[[nz, j]] - u[[nz - 1, j]]}, {j, 1, ns - 1}];
g = Append[g, lrow];
lcol = Append[Table[{u[[i, ns]] - u[[i, ns - 1]],
     u[[i + 1, ns]] - u[[i, ns]]}, {i, 1, nz - 1}],
   {u[[nz, ns]] - u[[nz, ns - 1]], u[[nz, ns]] - u[[nz - 1, ns]]}];
g = Transpose[Append[Transpose[g], lcol]];
```

■ **calculate integral over electric field**

Use (27) to get $\tilde{C}$ in [F/m] from the field:

```
ϵ0 = 10^-9 / (35.95 π);
cap = ϵ0 Sum[ϵ[[i, j]] g[[i, j]].g[[i, j]], {i, 1, nz}, {j, 1, ns}]
```

6.28904×10^{-11}

■ **results for the capacities**

The following results were obtained in three runs with the parameters vi as specified :
$\tilde{C}_{11}$ = 62.9 [pF/m] (v1 = 1, v2 = 0)
$\tilde{C}_{22}$ = 75.7 [pF/m] (v1 = 0, v2 = 1)
$\tilde{C}\tilde{C}$ = 66.1 [pF/m] (v1 = 1, v2 = 1)
$\tilde{C}_{12}$ = 0.5 ($\tilde{C}\tilde{C}$ - $\tilde{C}_{11}$ - $\tilde{C}_{22}$) = - 36.3 [pF/m]

To understand the negative sign of $\tilde{C}_{12}$ look at (21) with U_1 = 1 and all other voltages set to zero. From the sign of the field produced by the charges on both conductors (see picture of field above) it is seen that for example $\tilde{C}_{11}$ and $\tilde{C}_{12}$ must be of opposite sign.

With (24) follow the coefficients of capacity:

C_{10} = 26.6 [pF/m]
C_{20} = 39.4 [pF/m]
C_{12} = 36.3 [pF/m]

There is good agreement with the theory. The formula from the textbook gives $\epsilon_{r,\text{eff}}\,\epsilon_0\,\frac{A}{d\,l} \approx 36.8$ [pF/m] (enter A=19, d=13 and $\epsilon_{r,\text{eff}}$ =(8*4+5*1)/(8+5)=2.85 as parameters). It takes into account the mutual capacity C_{12} of the two conductors and ignores the surrounding ground.
The values $C_{i\,j}$ reflect the *geometry* of the electrodes and may be combined in various ways to model different modes of operating this capacitor :
differential: effective C = C_{12} + C_{10} || C_{20} = 52.2 [pF/m] (|| means here: in series)
electrode 1 grounded (C_{10} =∞) : C = C_{12} + C_{20} = 75.7 [pF/m]
electrode 2 grounded (C_{20} =∞) : C = C_{12} + C_{10} = 62.9 [pF/m]
The differences are sizeable, because the stray capacities are not negligible.

Inductances

Definition of Coefficients of Self- and Mutual Inductance

There is a far reaching *duality* between electric and magnetic fields, voltages and currents, or capacities and inductances. Because of that, the calculation of the inductance of a circuit proceeds on a terrain already paved by the previous section.

In a system of n circuits we have the equations

$$\begin{aligned} \varphi_1 &= L_{11} * I_1 + L_{12} * I_2 + \ldots + L_{1n} * I_n \\ \varphi_2 &= L_{21} * I_1 + L_{22} * I_2 + \ldots + L_{2n} * I_n \\ &\vdots \\ \varphi_n &= L_{n1} * I_1 + L_{n2} * I_2 + \ldots + L_{nn} * I_n \end{aligned} \tag{33}$$

where φ_i is the magnetic flux, I_i the current through circuit i and the $L_{ij} = L_{ji}$ are called 'coefficients of mutual inductance', L_{ii} is the self-inductance of circuit i.

Determination of the Coefficients of Inductance in a System of Linear Circuits

The potential energy of system of linear circuits, carrying certain currents I_i, may then be written as

$$W = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} L_{ij} I_i I_j . \tag{34}$$

There will be a magnetic field, whose shape is a consequence of the given geometry, and whose energy amounts to

$$W = \frac{1}{2} \int_V \vec{B} \cdot \vec{H} \, dV . \tag{35}$$

The energy of a current carrying system of circuits is thus stored in the magnetic field between them. So the coefficients of inductance may be found through the field. We get the self-inductance per unit length L_{ii}' of the linear circuit i out of (34) and (35) after integration over the plane perpendicular to the conductor the field, which corresponds to the boundary conditions $I_i = 1$ and all other $I_k = 0$:

$$L_{ii}' = \mu_0 \int_A \vec{H}_i \cdot \vec{H}_i \, dx \, dy \tag{36}$$

After the self-inductance the coefficients of mutual inductance are obtained from

$$L_{ij}' = \frac{1}{2} (\mu_0 \int_A \vec{H}_{ij} \cdot \vec{H}_{ij} \, dx \, dy - L_{ii}' - L_{jj}'), \tag{37}$$

where the indices of $\vec{H}$ point to the two circuits, whose currents were fixed to 1.

The Magnetic Field $\vec{H}$

The magnetic field $\vec{H}$ can be derived from the vector potential $\vec{A}$:

$$\vec{H} = \vec{\nabla} \times \vec{A} \tag{38}$$

Ampere's Law ($\vec{\nabla} \times \vec{H} = \vec{J}$) leads with (38) to

$$\vec{\nabla} \times \vec{\nabla} \times \vec{A} = \vec{\nabla} (\vec{\nabla} \cdot \vec{A}) - \Delta \vec{A} = \vec{J}, \tag{39}$$

where $\vec{J}$ is the current density. In our case we may orient the coordinate systems z axis parallel to the current. As the vector potential is parallel to the current (**Ref. 7**, p.304), it has only a z coordinate A_z. If in addition the Coulomb Gauge ($\vec{\nabla} \cdot \vec{A} = 0$) is applied, equation (39) changes to a Poisson equation

$$\Delta A_z = - J_z . \tag{40}$$

The distribution of the currents is not known, what will the boundary condition be like ? This is similar to our ignorance of the charge distribution in the electric case above (currents = moving charges after all !). For a way out imagine that currents of magnitude 1 have built up a vector potential according to (40). We cut out all current carrying conductors 1,...,i,...,n of the plane along their outer surfaces and nail down the values v_i of A_z at the cut i, as if the currents were still there. Now the potential satisfies in the current free space a Laplace equation

$$\Delta A_z = 0 \qquad (41)$$

plus certain boundary conditions. At each line of cut the value of A_z will be constant and can be found by trial and error from the requirement

$$I = \oint \vec{H} \cdot d\vec{l} = \oint (\vec{\nabla} \times \vec{A}) \cdot d\vec{l} = \pm 1 \text{ or } 0, \qquad (42)$$

if the integration path l is around the surface cut. The task has now become the same as before, namely numerically solving the Laplace equation with (iteratively determined) boundary conditions.

■ *Mathematica* Code (Coaxial Cable)

RAM requirements: Front End 15 MB, Kernel 25 MB

■ set up matrix:

This code is very similar to the code example of the previous section...

```
nz=130; (* divisible by 2 *)
ns=130; (* divisible by 2 *)
u=.;m=.;
m=Array[u,{nz,ns}];a=Array[u,{nz,ns}];mb=Table[0,{nz},{ns}];
Do[Do[a[[i,j]]=u[[i,j]],{j,1,ns}],{i,1,nz}]
Do[Do[m[[i,j]]:=Evaluate[0.25(a[[i-1,j]]+a[[i+1,j]]+a[[i,j-1]
]+a[[i,j+1]])],{j,2,ns-1}],{i,2,nz-1}]
```

– *Part::partd : Part specification u⟦1, 3⟧ is longer than depth of object.*

– *General::stop :*
Further output of Part::partd will be suppressed during this calculation.

■ boundary conditions:

```
Do[m[[1, j]] := 0., {j, 1, ns}]
Do[m[[nz, j]] := 0., {j, 1, ns}]
Do[m[[i, 1]] := 0., {i, 1, nz}]
Do[m[[i, ns]] := 0., {i, 1, nz}]
```

The value of the vector potential A_z at r_0 -> v1 has to be adjusted according to (42). The vector potential is in this case independent of φ, but changes radially even across the metal. A complete expression is given at the end of this section. Usually one starts with a guess, here we may use the known value (see (43)) for the coax cable (cross section Fig.5) :

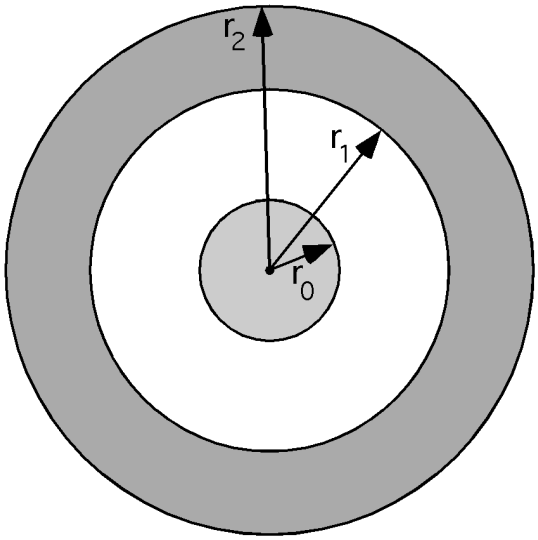


Figure 5

Figure 5. Inner wire and shield of a coax cable with their corresponding radii.

Because of the complete shielding of the coax the addition of a constant doesn't change the shape of the vector potential between r_0 and r_1 . Therefore the vector potential at r_1 can be put to zero:

$$A_z(r) = \frac{1}{4\pi} \text{Log}\left[\frac{r1^2}{r^2}\right]. \qquad (43)$$

```
v1=0.11248766;
v2=0;
```

Set two circular boundaries at r_0 (inner wire) and r_1 (shield):

```
{xr, yr, r0, r1, r2} = {65, 65, 30., 60., 63};
Do[Do[{b = Sqrt[(j - xr)^2 + (i - yr)^2];
   If[b ≥ 0 && b ≤ r0 + 0.0, {m[[i, j]] := v1; mb[[i, j]] = 1;}];
   If[b ≥ r1 - 0.0 , {m[[i, j]] := v2; mb[[i, j]] = 1;}]},
  {i, nz}], {j, ns}]
```

If the boundaries are curved, they can only be mapped approximately to a square grid. In all tests the best way to find the sets of grid line intersection points that belong to the boundary areas and are to be kept fixed during the relaxation process, was just to include or exclude the points according to the radius of curvature of the corresponding boundary (see cell above). The general agreement between theory and numerical calculation (about 2%) could – in the case of coaxial geometry – be somewhat improved (-> 0.5%) by using a radius of curvature that extends the fixed areas by 0.3 units. But this turned out not to be true for other configurations like for instance a pair of circular wires.

- **calculate potential**

Calculate rough guess u1 on coarse grid as input for main calculation:

```
nz1=nz/2;ns1=ns/2; u1=.;m1=.;
m1=Array[u1,{nz1,ns1}];a1=Array[u1,{nz1,ns1}];
Do[Do[a1[[i,j]]=u1[[i,j]],{j,1,ns1}],{i,1,nz1}]
Do[Do[
m1[[i,j]]:=Evaluate[0.25(a1[[i-1,j]]+a1[[i+1,j]]+a1[[i,j-1]]+
a1[[i,j+1]])],
{j,2,ns1-1}],{i,2,nz1-1}];
Do[m1[[1,j]]:=0.,{j,1,ns1}];
Do[m1[[nz1,j]]:=0.,{j,1,ns1}];
Do[m1[[i,1]]:=0.,{i,1,nz1}];
Do[m1[[i,ns1]]:=0.,{i,1,nz1}];
(* guess for coordinates:old/2 *)
Do[Do[If[mb[[2i,2j]]==1,m1[[i,j]]=m[[2i,2j]]],{i,nz1}],
{j,ns1}]
```

– *Part::partd : Part specification u1⟦1, 3⟧ is longer than depth of object.*

– *General::stop :*
Further output of Part::partd will be suppressed during this calculation.

Laplace at work...

```
u1=Table[0.,{i,1,nz1},{j,1,ns1}];
Timing[Do[{u1=m1;
If[Mod[i,500]==0,
err=Max[Flatten[Abs[u1-m1]]];
Print[i," residual error: ",err];
If[err<10^-6,Break[]]},{i,1,4000}]]
```

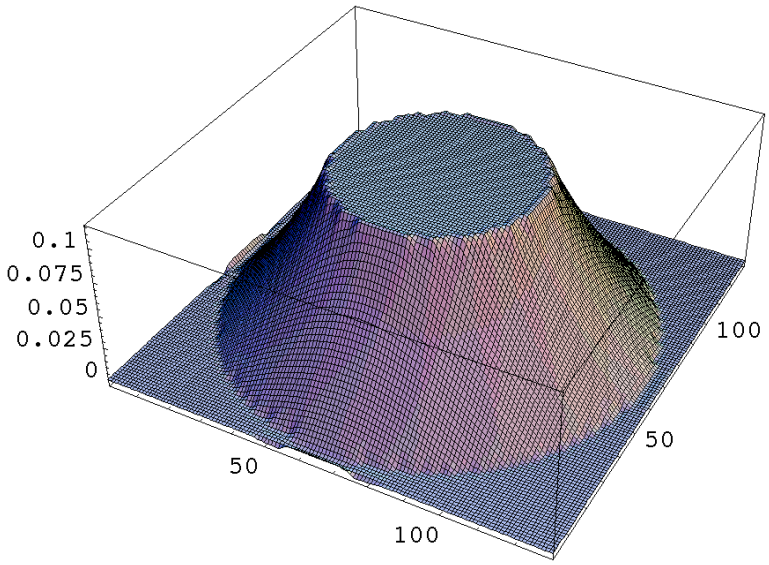
500 residual error: 4.21964×10^{-6}

1000 residual error: 2.79694×10^{-8}

{280.867 Second, Null}

Copy coarse to fine grid:

```
p = ListInterpolation[u1];
u = Table[p[i / 2, j / 2], {i, 1, nz}, {j, 1, ns}];
ListPlot3D[u, PlotRange -> All];
```

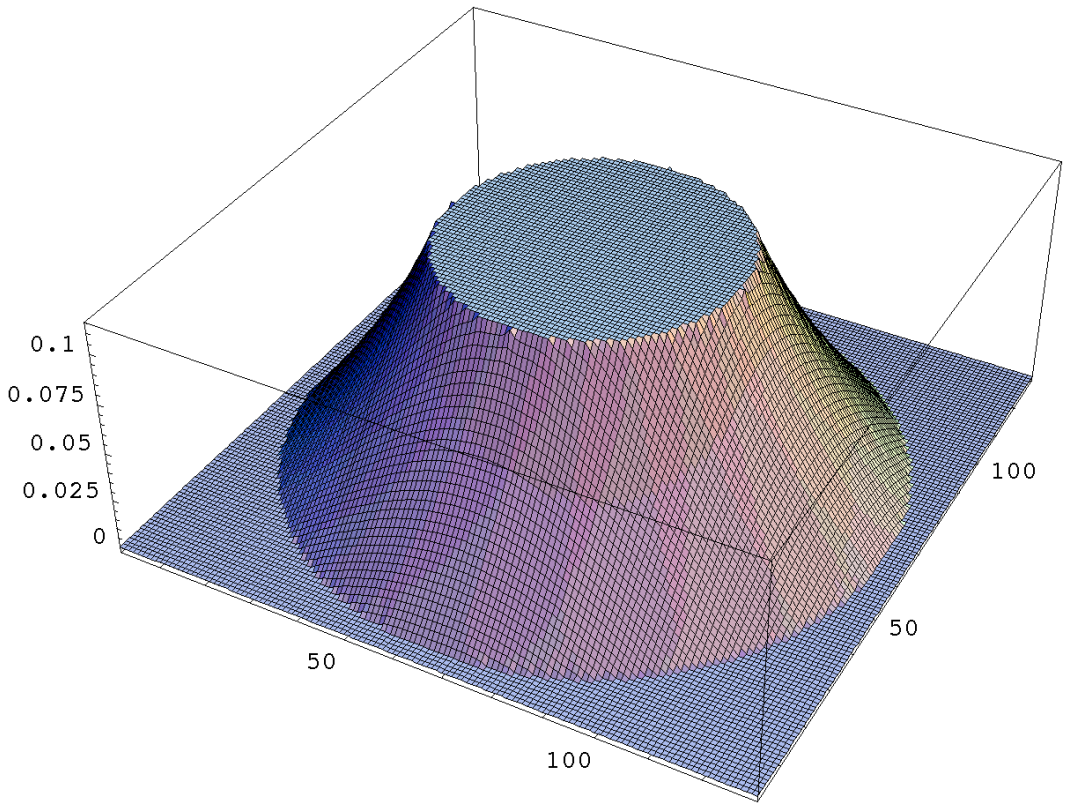


Control plot of interpolated vector potential.

restless Laplace...

```
Timing[Do[{u=m;If[Mod[i,100]==0,
{err=Max[Abs[Flatten[u-m]]];
Print[i," residual error: ",err];
If[err<10^-6,Break[]]}]},{i,1,7000}]]
```

```
ListPlot3D[u,ViewPoint->{1.300, -2.400, 2.000},
ImageSize->400,PlotRange->All]
```



The picture shows the resulting vector potential with its logarithmic drop between r_0 and r_1.

- **gradient field...**

```
g = Table[{u[[i, j + 1]] - u[[i, j]], u[[i + 1, j]] - u[[i, j]]},
   {i, 1, nz - 1}, {j, 1, ns - 1}];
lrow = Table[{u[[nz, j + 1]] - u[[nz, j]],
    u[[nz, j]] - u[[nz - 1, j]]}, {j, 1, ns - 1}];
g = Append[g, lrow];
lcol = Append[Table[{u[[i, ns]] - u[[i, ns - 1]],
     u[[i + 1, ns]] - u[[i, ns]]}, {i, 1, nz - 1}],
   {u[[nz, ns]] - u[[nz, ns - 1]], u[[nz, ns]] - u[[nz - 1, ns]]}];
g = Transpose[Append[Transpose[g], lcol]];
```

```
μ0 = 4. π * 10^-7;
```

Calculate $\oint \vec{H}\, d\vec{l}$ around inner conductor on a circle with radius r and center (xr,yr);
$\vec{H} \cdot d\vec{l} = (\partial_y A_z, -\partial_x A_z) \cdot (-r\, d\varphi \sin\varphi, r\, d\varphi \cos\varphi)$ and A_z is evaluated at (xr + r cosφ, yr + r sinφ)

```
dphi = π / 100;
i = Table[ Sum[-rr * dphi (
     g[[Round[yr + rr * Sin[phi]], Round[xr + rr * Cos[phi]], 2]] *
       Sin[phi] + g[[Round[yr + rr * Sin[phi]],
         Round[xr + rr * Cos[phi]], 1]] * Cos[phi]),
   {phi, 0, 2 π, dphi}], {rr, 31, 59}]
iMean = Apply[Plus, i] / Length[i]
```

```
{0.997655, 0.99814, 0.997853, 0.995982, 0.993919,
 0.99758, 0.995584, 0.998464, 0.995632, 0.998036, 0.997206,
 0.996704, 1.0006, 0.998764, 0.998503, 0.999453, 0.999996,
 1.00031, 0.999463, 1.00223, 1.00251, 1.00225, 1.00258,
 1.00298, 1.00618, 1.00468, 1.00418, 1.00123, 1.01134}
```

```
1.
```

The above cell shows the mean of many loop integrals that all should give the same result as long as the radius rr does not touch other currents. v1 in the cell at the start of this section must be varied such that iMean becomes 1. The following result is only correct, if the number for iMean is close enough to 1.

- **result:**

With $\vec{H} = (\partial_y A_z, -\partial_x A_z)$ and (36) the external inductance (between r_0 and r_1) is in [H/m] :

$$L_{ii}' = \mu_0 \int_A (\partial_y A_z)^2 + (\partial_x A_z)^2 \, dx\, dy, \tag{44}$$

leading to:

```
ind = μ0 * Sum[g[[i, j]].g[[i, j]], {i, 1, nz}, {j, 1, ns}]
```

```
1.40587 × 10^-7
```

This is in nice agreement with the theoretical value:

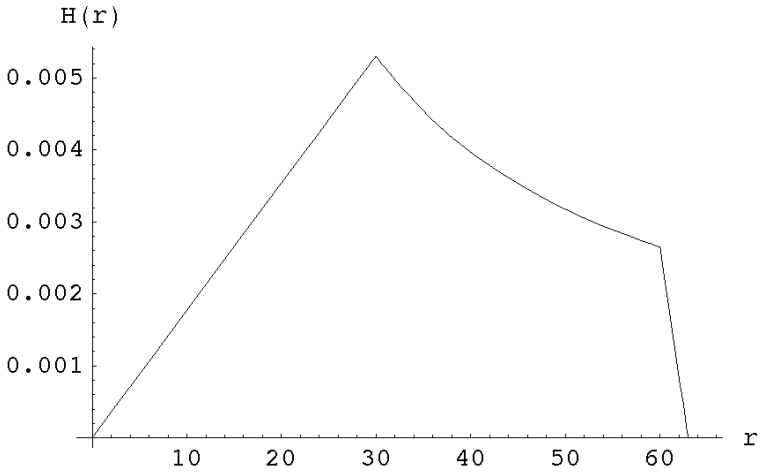
```
μ0 / (2 π) Log[r1 / r0]
```

```
1.38629 × 10^-7
```

For the complete coax including inner wire and shield we have (compare Fig. 5 with r_0 = 30, r_1 = 60, r_2 = 63; current in the wire =+1, current in the shield =-1) :

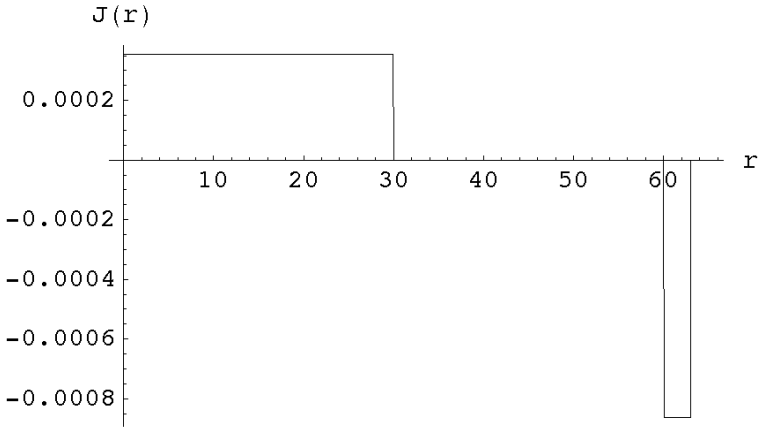
• the magnetic Field as function of r (**Ref. 7**, p.321f) :

```
h[r_] := Which[r < r0, r/(2 π r0^2), r > r2,
  0., r > r1, 1/(2 π r) (r2^2 - r^2)/(r2^2 - r1^2), True, 1/(2 π r)]
Plot[h[r], {r, 0, ns / 2}, AxesLabel -> {"r", "H(r)"}];
```



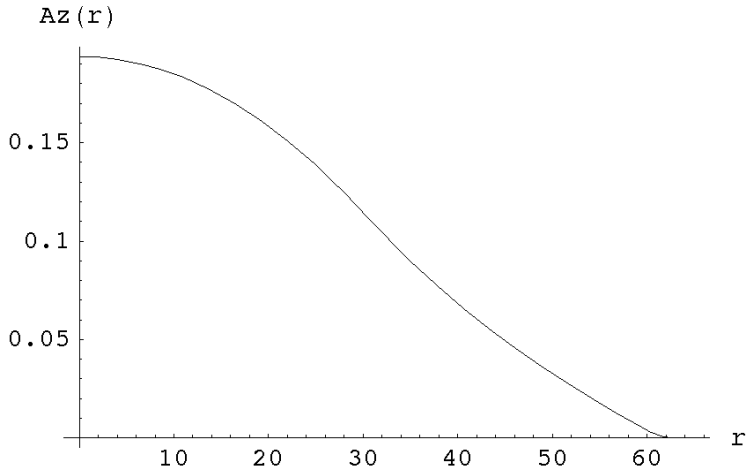
• the current density :

```
jz[r_] :=
 Which[r < r0, 1/(π r0^2), r > r2, 0., r > r1, 1/(π (r1^2 - r2^2)), True, 0]
Plot[jz[r], {r, 0, ns / 2}, AxesLabel -> {"r", "J(r)"}];
```



• the corresponding vector potential az(r) ($A_z(r_2)$ is forced to 0) :

```
az[r_] := Which[
  r < r0, 1/(4 π) (-r^2/r0^2 + Log[r1^2/r0^2] + r2^2/(r2^2 - r1^2) Log[r2^2/r1^2]),
  r > r2, 0.,
  r > r1, 1/(4 π) (-(r2^2 - r^2)/(r2^2 - r1^2) + r2^2/(r2^2 - r1^2) Log[r2^2/r^2]),
  True, 1/(4 π) (Log[r1^2/r^2] - 1 + r2^2/(r2^2 - r1^2) Log[r2^2/r1^2])]
Plot[az[r], {r, 0, ns/2}, AxesLabel -> {"r", "Az(r)"}];
```



The coax geometry allows easy comparison of the theoretical value of its self-inductance with the one that was numerically calculated, because the boundary of the relaxation area has finite extension. Many of the other configurations of conductors with known self-inductance have boundary values of the vector potential that are 'zero at infinity' and require larger grids to reach good accuracy.

■ Some results with other geometries:

▪ Coefficients of capacity of a pair of wires above a grounded plane:

Parameters used in the relaxation:
grid size : 240 * 150, center of wire 1 at (90, 40), center of wire 2 at (150, 40), height h above ground = 40, distance of wires a = 60, diameter d, s = $\sqrt{1+4h^2/a^2}$, t = $4h/d$, all lengths in grid units, the expressions for the capacities are found in (**Ref. 7**, p. 118), capacities in Table 1 are given in [F/m]:

d	$\frac{2\pi\epsilon_0}{\ln(s*t)}$	$C_{10} = C_{20}$ (num.)	$\frac{2\pi\epsilon_0 \ln s}{(\ln t)^2 - (\ln s)^2}$	C_{12} (num.)
2	$1.14 * 10^{-11}$	$1.2 * 10^{-11}$	$1.5 * 10^{-12}$	$1.15 * 10^{-12}$
4	$1.32 * 10^{-11}$	$1.38 * 10^{-11}$	$2.13 * 10^{-12}$	$1.58 * 10^{-12}$
6	$1.47 * 10^{-11}$	$1.56 * 10^{-11}$	$2.7 * 10^{-12}$	$2.06 * 10^{-12}$

Table 1. Analytical (zero at infinity) and numerical capacities of a pair of wires [F/m]

The main contribution to the large differences between the numerical relaxation and the formula from the textbook stems from the fact that the relaxation, however large a grid is chosen, imposes its boundary condi-

tions at a finite distance, namely the border of the grid, while the formula uses a potential, which is 'zero at infinity'. To better check the accuracy of the relaxation, a new formula was developed that returns the exact potential of a charge in a box through the method of images with 'infinite reflection'. If a charge Q is placed inside a (2 – dimensional, rectangular) box, an infinite number of image charges of alternating sign will appear in x– and y– direction (Fig. 6). Every remote image adds a term to the potential, which remains finite because of the alternating sign. They 'compress' the potential function in such a way that its value at the borders of the box is constant.

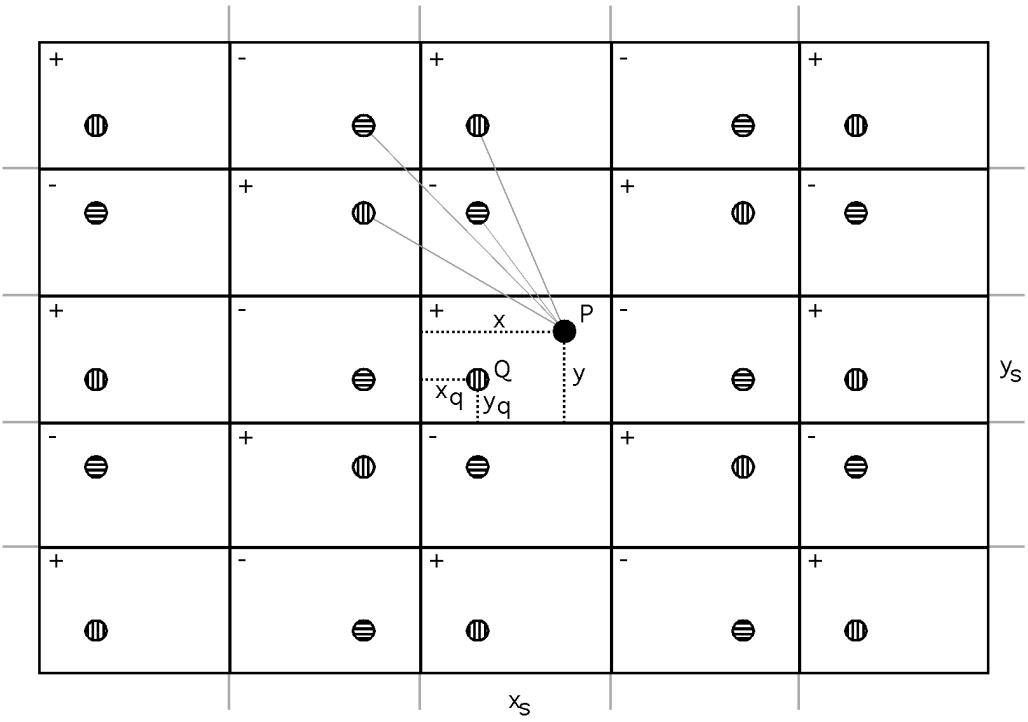


Figure 6

Figure 6. 'Mirror Cabinet' of a charge and its images in a 2 – dimensional box, the indicated group of 4 charges constitutes a term (n = 0, m = 1) in the sum below

The summation and reordering of all terms then gives the potential function at the point x, y within the box as a closed expression:

ϕ (x,y) = $\frac{Q}{2\pi\epsilon_0 l} f(x, y, x_s, y_s, x_q, y_q) + k$, where

$$f = \sum_{m,n=-\infty}^{\infty} \ln\left(\frac{\sqrt{(2 n x_s - x + x_q)^2 + (2 m y_s - y - y_q)^2}\ \sqrt{(2 n x_s - x - x_q)^2 + (2 m y_s - y + y_q)^2}}{\sqrt{(2 n x_s - x + x_q)^2 + (2 m y_s - y + y_q)^2}\ \sqrt{(2 n x_s - x - x_q)^2 + (2 m y_s - y - y_q)^2}} \right),$$

and k is a reference point of the potential on the border. Indices m,n = 0 denote the term in the middle next to the original charge, each square root represents the distance of a charge to the point P. The meaning of the variables is explained in Fig. 6. The double sum converges well, only a few tens of terms are needed in x and y direction. In addition the correct boundary conditions ϕ (0, y) = ϕ (x_s, y) = ϕ (x, 0) = ϕ (x, y_s) = k are fulfilled at the edges of the box. For the x – direction this can be seen by writing out a few terms of the sum in the index n: at x = x_s terms n=0 and n=1 cancel, then n=-1 and n=2 and so on, at x = 0 the parts of term n=0 cancel, then -1 and 1, -2 and 2 etc. A similar argument is valid for the y – direction. The boundary conditions are also evident from the symmetry in Fig. 6 with respect to (x = 0 or x = x_s) or (y = 0 or y = y_s).

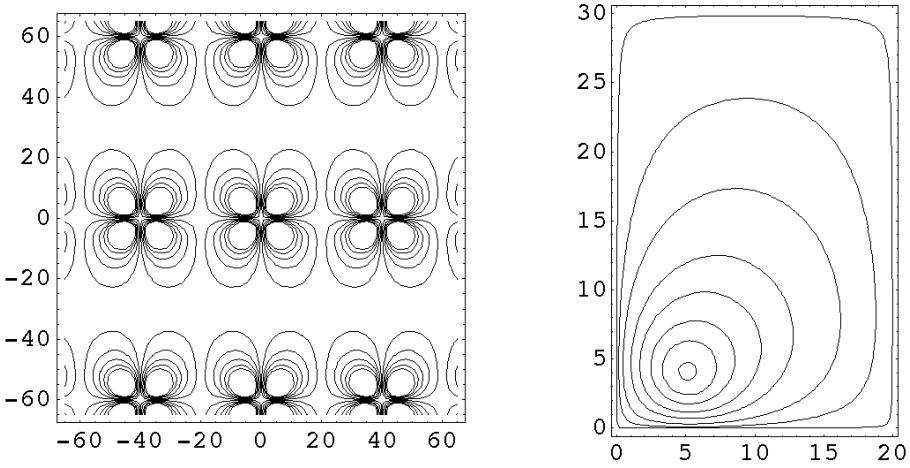
The above double infinite sum is equal to (**Ref. 10**)

$$f = \frac{1}{2}\,\ln\left(\frac{|\vartheta_1(\pi(\frac{x+xq}{2xs}+i\,\frac{ys}{xs}\,\frac{y-yq}{2ys}),e^{-\pi ys/xs})|^2 * |\vartheta_1(\pi(\frac{x-xq}{2xs}+i\,\frac{ys}{xs}\,\frac{y+yq}{2ys}),e^{-\pi ys/xs})|^2}{|\vartheta_1(\pi(\frac{x+xq}{2xs}+i\,\frac{ys}{xs}\,\frac{y+yq}{2ys}),e^{-\pi ys/xs})|^2 * |\vartheta_1(\pi(\frac{x-xq}{2xs}+i\,\frac{ys}{xs}\,\frac{y-yq}{2ys}),e^{-\pi ys/xs})|^2}\right),$$

with the first elliptic theta function defined as (**Ref. 8**, p.778)
$\vartheta_1(u, q) = 2\sum_{n=0}^{\infty}(-1)^n\, q^{(n+1/2)^2}\,\sin[(2n+1)u]$. (The sum converges *very* fast...)

A control plot of f displays the typical double periodic behaviour (as in Fig. 6) :

```
xs = 20; ys = 30; xq = 5.1; yq = 4.1; q = Exp[- π ys / xs];
ContourPlot[1 / 2 Log[Chop[
    (EllipticTheta[1, π ((x + xq)/(2 xs) + I (y - yq)/(2 xs)), q]
      EllipticTheta[1, π ((x + xq)/(2 xs) - I (y - yq)/(2 xs)), q]
      EllipticTheta[1, π ((x - xq)/(2 xs) + I (y + yq)/(2 xs)), q]
      EllipticTheta[1, π ((x - xq)/(2 xs) - I (y + yq)/(2 xs)), q]) /
     (EllipticTheta[1, π ((x + xq)/(2 xs) + I (y + yq)/(2 xs)), q]
      EllipticTheta[1, π ((x + xq)/(2 xs) - I (y + yq)/(2 xs)), q]
      EllipticTheta[1, π ((x - xq)/(2 xs) + I (y - yq)/(2 xs)), q]
      EllipticTheta[1, π ((x - xq)/(2 xs) - I (y - yq)/(2 xs)), q])]],
 {x, -65, 65}, {y, -65, 65},
 PlotPoints → 100, ContourShading -> False]
```



Each one of the four leaves of the "flowers" in the above picture to the left represents the exact distribution of the potential of a line charge inside a conducting box. The picture to the right shows a section containing just the central leaf, plotted is the area of the containing box with sides xs and ys.

The coefficients of capacity of two wires 1 and 2 with diameter d at the positions cited at the beginning of this section may then be found according to (23) and (**Ref. 7**, p. 118) with

$f(x_1, y_1 + d/2, x_s, y_s, x_1, y_1) = f_{11}$,
$f(x_1, y_1 + d/2, x_s, y_s, x_2, y_2) = f_{12}$,
$f(x_2, y_2 + d/2, x_s, y_s, x_1, y_1) = f_{21}$,
$f(x_2, y_2 + d/2, x_s, y_s, x_2, y_2) = f_{22}$

from

$C_{10}' = \frac{2\pi\epsilon_0 (f_{22} - f_{12})}{f_{11} f_{22} - f_{12} f_{21}}$, $C_{20}' = \frac{2\pi\epsilon_0 (f_{11} - f_{21})}{f_{11} f_{22} - f_{12} f_{21}}$, $C_{12}' = \frac{2\pi\epsilon_0 f_{12}}{f_{11} f_{22} - f_{12} f_{21}}$, $C_{21}' = \frac{2\pi\epsilon_0 f_{21}}{f_{11} f_{22} - f_{12} f_{21}}$:

d	new **formula**	$C_{10} = C_{20}$ **(num.)**	new **formula**	$C_{12} = C_{21}$ **(num.)**
2	$1.2 * 10^{-11}$	$1.2 * 10^{-11}$	$1.22 * 10^{-12}$	$1.15 * 10^{-12}$
4	$1.41 * 10^{-11}$	$1.38 * 10^{-11}$	$1.76 * 10^{-12}$	$1.58 * 10^{-12}$
6	$1.56 * 10^{-11}$	$1.56 * 10^{-11}$	$2.26 * 10^{-12}$	$2.06 * 10^{-12}$

Table 2. Analytical (zero at boundary of the box) and numerical capacities of a pair of wires [F/m]

The agreement (cf. Table 2) is much better for the stray capacities and reasonable for the mutual capacity.

■ **Coefficients of inductance of a pair of wires:**

If the current flowing through a circuit is kept constant the magnetic field is insensitive to whether the vector potential has its zero boundary at infinity or at the border of a box, because the rotation of the vector potential doesn't change – in contrast to the gradient of the electric potential. This is accounted for in the relaxation calculation by the necessary readjustment of the vector potential (42) at the edge of the conductors so that the current stays at 1. Then the formula for the self-inductance ($l = (\mu_0 / \pi)$ `Log[a / r]`) and mutual inductance ($l_{12} = \mu_0 / (2\pi)$ `Log[r14 r23 / (r13 r24)]`) from the textbook (**Ref. 7**, p.316, 325) can be directly compared to the numerical computation. The calculation was done on a grid of 120 * 90 points, the positions of the conductors 1 to 4 lying at (60, 50), (50, 50) and (60, 40), (70, 40) respectively. The loop integrals (42) were taken with radii ranging from 4 to 6 in steps of 0.2, then their mean was used to determine the current. The results for l_1, the self-inductance of circuit 1, (one circuit at current 1, the other at current zero – as if not present –), l_2 and ll (both circuits at current 1) are given below together with the corresponding values of the vector potential.

r = 1 l = 9.21*10^-7 [H/m]:
l_1 = 9.19*10^-7; v1 = 0.370859; v2 = -0.3628862; v3 = -0.06284042; v4 = -0.03036495;
l_2 = 9.16*10^-7; v1 = 0.0259072; v2 = 0.0579781; v3 = 0.352414; v4 = -0.381698;
ll = 1.758*10^-6; v1 = 0.396796; v2 = -0.3049608; v3 = 0.289506 v4 = -0.4120585;

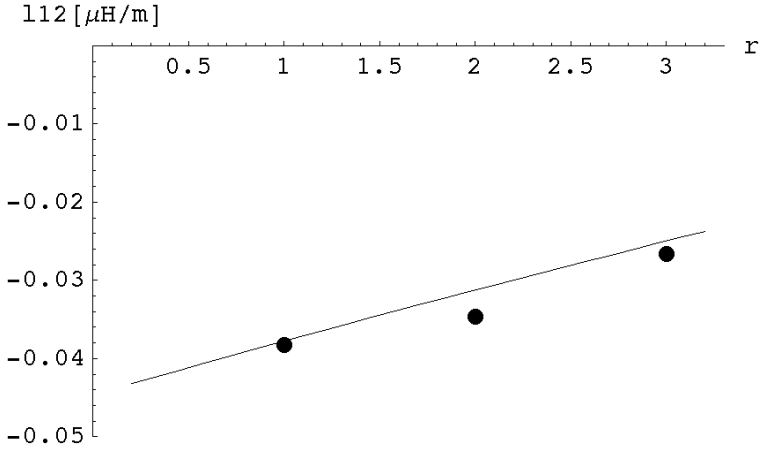
r = 2 l = 6.44*10^-7 [H/m]:
l_1 = 6.65*10^-7; v1 = 0.2702775; v2 = -0.2609418; v3 = -0.0589633; v4 = -0.0294141;
l_2 = 6.64*10^-7; v1 = 0.024736; v2 = 0.053704; v3 = 0.250446; v4 = -0.281441;
ll = 1.26*10^-6; v1 = 0.294926; v2 = -0.2073; v3 = 0.191429; v4 = -0.310924;

r = 3 l = 4.82*10^-7 [H/m]:
l_1 = 4.67*10^-7; v1 = 0.193345; v2 = -0.179803; v3 = -0.0505313; v4 = -0.0272583;
l_2 = 4.66*10^-7; v1 = 0.02257533; v2 = 0.045243; v3 = 0.1695425; v4 = -0.203956;
ll = 8.798*10^-7; v1 = 0.215993; v2 = -0.134556; v3 = 0.118963; v4 = -0231246;

The formula for the mutual inductance had to be modified to include the dependence on the wire radius:

$$l_{12}(r) = \frac{\mu_0}{2\pi} \text{Log}\left[\frac{(r_{14} - r)(r_{23} + r)}{(r_{13} + r)(r_{24} - r)}\right].$$

r_{ij} denotes the distance of conductors i and j. Three values of the mutual inductance for r = 1, 2, 3, calculated with relaxation, are shown as black dots, the thin line represents the formula :



The overall agreement of the relaxation calculation with the theoretical expressions is satisfactory.

The procedures developed until now will be applied to calculate the crosstalk of coupled microstrip lines in the following section.

The Pigtail

The physical dimensions of the pigtail connector for the ATLAS detector are shown in Fig. 7.

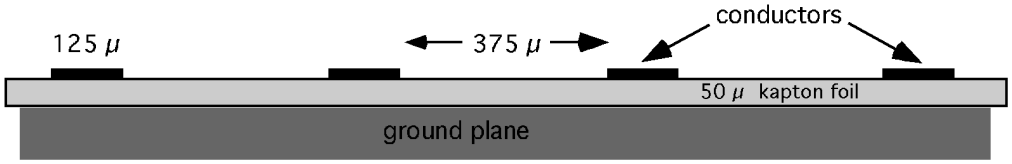


Figure 7

Figure 7. Cross section of the Pigtail connector. The conductors have a thickness of 17μ.

Size and distance of the 4 conductors and the dielectric foil are mapped onto a 446 * 80 grid to calculate the coefficients of capacity and inductance of this system. Pairs of conductors are operated as differential lines to minimize disturbances. The width and distance of the conductors will be 22 and 66 grid squares, the thickness of the conductors and the dielectric (ϵ = 4) will be 3 and 9 squares. The numbers of squares should be chosen as large as program memory allows to minimize quantization error.

■ *Mathematica* Code (Coefficients of Induction for the pigtail)

RAM requirements: Front End 40 MB,Kernel 80 MB

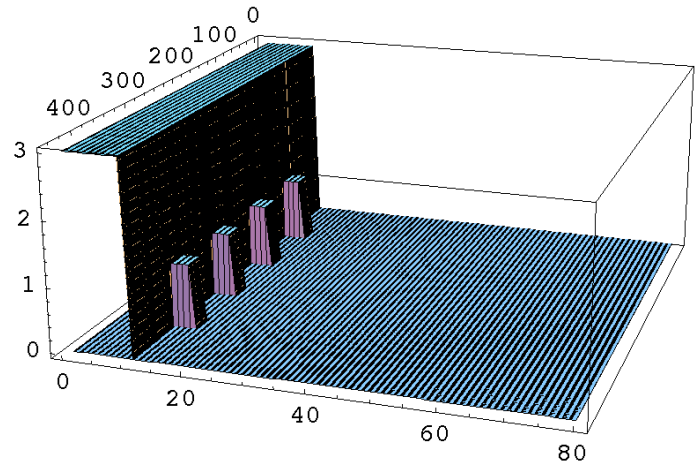
```
v1=1.;v2=1.;v3=0.;v4=0.;(* potential of the electrodes *)
nz=80; (* # of rows of the grid, divisible by 2 *)
ns=446; (* # of columns of the grid, divisible by 2 *)
u=.;m=.;m=Array[u,{nz,ns}];a=Array[u,{nz,ns}];
mb=Table[0,{nz},{ns}];
ϵ=Table[1,{i,1,nz},{j,1,ns}];
eps=4;
exl=1;exr=ns;eyd=1;eyu=10; (* dielectric area *)
Do[Do[ϵ[[i,j]]=eps,{i,eyd,eyu}],{j,exl,exr}];
Do[Do[a[[i,j]]=u[[i,j]],{j,1,ns}],{i,1,nz}]
Do[Do[
m[[i,j]]:=Evaluate[0.25(a[[i-1,j]]+a[[i,j-1]]+a[[i+1,j]]+
a[[i,j+1]])+
0.0625/ϵ[[i,j]]((ϵ[[i,j+1]]-ϵ[[i,j-1]])(a[[i,j+1]]-
a[[i,j-1]])+(ϵ[[i+1,j]]-ϵ[[i-1,j]])(a[[i+1,j]]-a[[i-1,j]]))],
{j,2,ns-1}],{i,2,nz-1}]
Do[m[[1,j]]:=0.,{j,1,ns}]
Do[m[[nz,j]]:=0.,{j,1,ns}]
Do[m[[i,1]]:=0.,{i,1,nz}]
Do[m[[i,ns]]:=0.,{i,1,nz}]
xl1=80;xr1=102;yd1=11;yu1=14;
xl2=168;xr2=190;yd2=11;yu2=14;
xl3=256;xr3=278;yd3=11;yu3=14;
xl4=344;xr4=366;yd4=11;yu4=14;
Do[Do[{m[[i,j]]:=v1;mb[[i,j]]=1;},{j,xl1,xr1}],{i,yd1,yu1}]
Do[Do[{m[[i,j]]:=v2;mb[[i,j]]=1;},{j,xl2,xr2}],{i,yd2,yu2}]
Do[Do[{m[[i,j]]:=v3;mb[[i,j]]=1;},{j,xl3,xr3}],{i,yd3,yu3}]
Do[Do[{m[[i,j]]:=v4;mb[[i,j]]=1;},{j,xl4,xr4}],{i,yd4,yu4}]
(* uncomment next lines to verify coordinates *)
p1=ListPlot3D[ϵ-1,ViewPoint→{3.,1.,1.},
      DisplayFunction→Identity];
p2=ListPlot3D[mb,ViewPoint→{3.,1.,1.},
DisplayFunction→Identity];
Show[p1,p2,DisplayFunction→$DisplayFunction];
```

— *Part::partd : Part specification u⟦1, 1⟧ is longer than depth of object.*

— *Part::partd : Part specification u⟦1, 1⟧ is longer than depth of object.*

— *Part::partd : Part specification u⟦1, 2⟧ is longer than depth of object.*

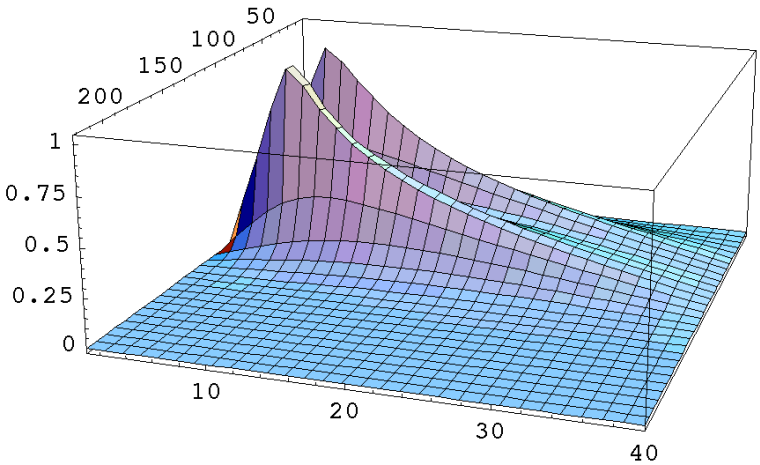
— *General::stop :*
Further output of Part::partd will be suppressed during this calculation.



The picture shows the location of the conductors and the dielectric material.

```
nz1=nz/2;ns1=ns/2;u1=.;m1=.;
m1=Array[u1,{nz1,ns1}];a1=Array[u1,{nz1,ns1}];
Do[Do[a1[[i,j]]=u1[[i,j]],{j,1,ns1}],{i,1,nz1}]
Do[Do[m1[[i,j]]:=Evaluate[0.25(a1[[i-1,j]]+a1[[i+1,j]]+
a1[[i,j-1]]+a1[[i,j+1]])],{j,2,ns1-1}],{i,2,nz1-1}];
Do[m1[[1,j]]:=0.,{j,1,ns1}];
Do[m1[[nz1,j]]:=0.,{j,1,ns1}];
Do[m1[[i,1]]:=0.,{i,1,nz1}];
Do[m1[[i,ns1]]:=0.,{i,1,nz1}];
(* guess for coordinates:old/2;mb:location of conductors *)
Do[Do[If[mb[[2i,2j]]==1,m1[[i,j]]=m[[2i,2j]]],{i,nz1}],
{j,ns1}]
u1=Table[0.,{i,1,nz1},{j,1,ns1}];
Timing[Do[{u1=m1;If[Mod[i,500]==0,
Print[i," residual error:",Max[Flatten[Abs[u1-m1]]]]]},
{i,1,3000}]]
```

```
p = ListInterpolation[Transpose[u1]];
Plot3D[p[x, y], {x, 1, ns1}, {y, 1, nz1},
 PlotRange → All, PlotPoints → 30, ViewPoint → {3., 1., 1.}]
```



Control plot of interpolated potential, two conductors at potential 1.

```
u=Table[p[j/2,i/2],{i,1,nz},{j,1,ns}];
Timing[Do[{u=m;
If[Mod[i,100]==0,{err=Max[Abs[Flatten[u-m]]];
Print[i," residual error: ",err];
If[err<10^-6,Break[]]}]},{i,1,7000}]]
ListPlot3D[u,
ImageSize->400,PlotRange->All,ViewPoint→{3.,1.,1.}];
```

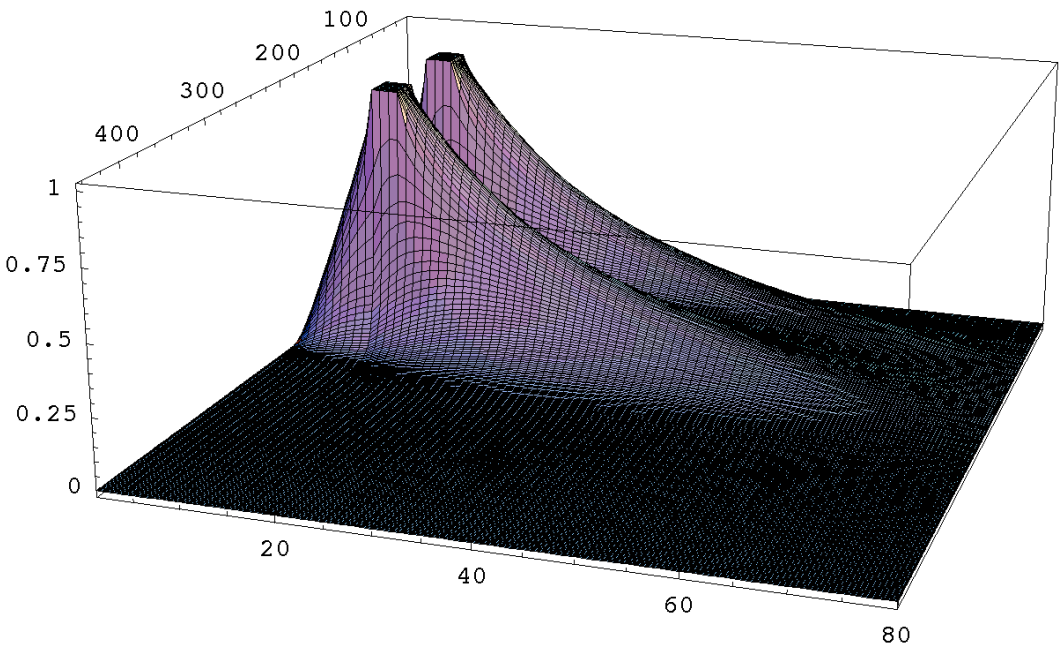
— *InterpolatingFunction::dmval :*
Input value $\{\frac{1}{2}, \frac{1}{2}\}$ *lies outside the range of data in the interpolating function. Extrapolation will be used.*

— *InterpolatingFunction::dmval :*
Input value $\{1, \frac{1}{2}\}$ *lies outside the range of data in the interpolating function. Extrapolation will be used.*

— *InterpolatingFunction::dmval :*
Input value $\{\frac{3}{2}, \frac{1}{2}\}$ *lies outside the range of data in the interpolating function. Extrapolation will be used.*

— *General::stop : Further output of InterpolatingFunction::dmval will be suppressed during this calculation.*

```
100 residual error: 0.000417988

6200 residual error: 1.03883×10^-6

6300 residual error: 9.82263×10^-7

{8891.7 Second, Null}
```



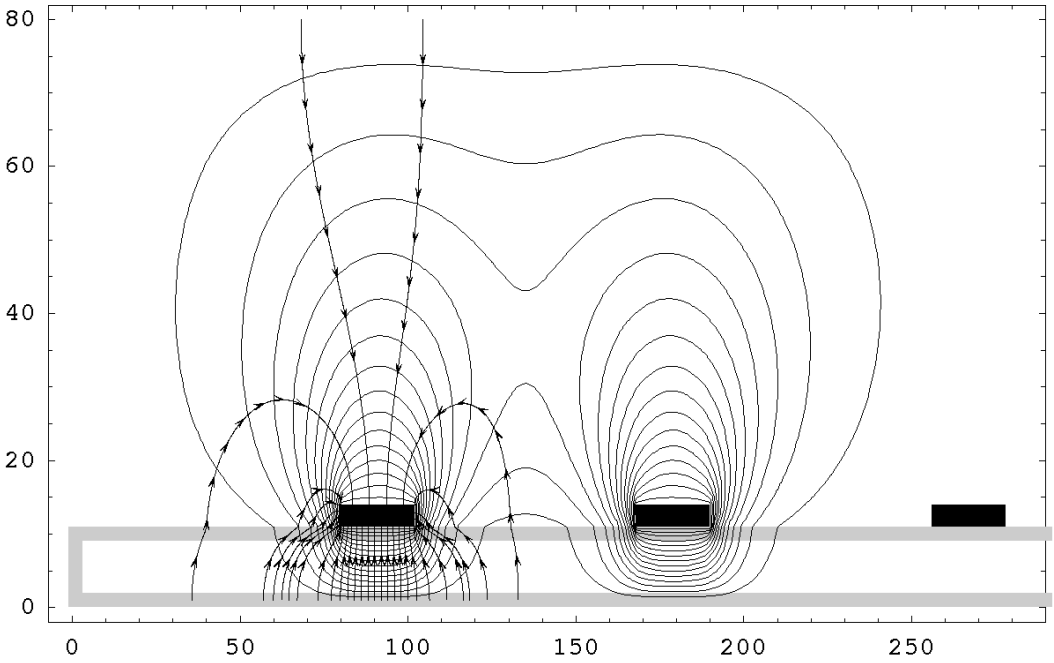
Potential distribution of the pigtail including dielectrics, with the first two conductors set to potential 1.

```
cont = ListContourPlot[u, ImageSize → 400,
  PlotRange → All, ContourShading → False,
  Contours → 15, DisplayFunction → Identity]
wi = ListInterpolation[Transpose[u], InterpolationOrder → 1];
```

```
(* column index counts increments in x direction,
 row index counts increments in y direction →
  use Transpose *)
ex = Derivative[1, 0][wi];
ey = Derivative[0, 1][wi];
FieldLine[{ex_InterpolatingFunction, x0_},
  {ey_InterpolatingFunction, y0_}] :=
 Module[{x = x0, y = y0, l = {{x0, y0}},
   xxmin = Part[ex, 1, 1, 1], xxmax = Part[ex, 1, 1, 2],
   yymin = Part[ex, 1, 2, 1],
   yymax = Part[ex, 1, 2, 2], emod10, xn, yn},
  (* search uphill *)
  Do[{emod10 = 10. √((ex[x, y])^2 + (ey[x, y])^2);
    If[emod10 == 0., Break[]];
    xn = ex[x, y] / emod10 + x; yn = ey[x, y] / emod10 + y;
    If[xn < xxmin || xn > xxmax ||
      yn < yymin || yn > yymax, Break[]];
    l = Append[l, {xn, yn}]; x = xn; y = yn}, {n, 1, 5000}];
  x = x0; y = y0;
  (* search downhill *)
  Do[{emod10 = -10. √((ex[x, y])^2 + (ey[x, y])^2);
    If[emod10 == 0., Break[]];
    xn = ex[x, y] / emod10 + x; yn = ey[x, y] / emod10 + y;
    If[xn < xxmin || xn > xxmax ||
      yn < yymin || yn > yymax, Break[]];
    l = Prepend[l, {xn, yn}]; x = xn; y = yn}, {n, 1, 5000}];
  Line[l]]
Needs["Graphics`Arrow`"]
AddArrow[Line[opts_], d_, num_: 8] :=
 Module[{arr = {}, n = 0, pts = Chop[opts]},
  Fold[If[First[#1] ≥ d && n < num, n++;
     AppendTo[arr, Arrow[Last[#1], #2, HeadScaling → Absolute,
       HeadCenter → 0.5, HeadLength → 4]];
     {0, #2}, {First[#1] + Sqrt[Apply[Plus, (Last[#1] - #2)^2]],
      #2}] &, {0, First[pts]}, Rest[pts]]; arr]
(** 1 **)
{xe, ye} = {91, 12.5}; a = 13; b = 6;
n = 21;(* n/3 is the number of angular bins *)
eps = ListInterpolation[Transpose[ϵ]];
(* sample the field strength on the ellipse *)
ft = Table[eps[a Cos[i 2π / n] + xe, b Sin[i 2π / n] + ye]
    √(ex[a Cos[i 2π / n] + xe, b Sin[i 2π / n] + ye]^2 +
       ey[a Cos[i 2π / n] + xe, b Sin[i 2π / n] + ye]^2), {i, 1, n}];
s = Apply[Plus, ft];
(* nl[[i]] is (number of lines - 1)
  starting in the ith angular bin of the ellipse *)
nl = Round[Apply[Plus, Transpose[Partition[ft, 3]]] / s * n];
start = {};
(* array of angles of the
  starting points on the ellipse *)
Do[Do[AppendTo[start, (2π / (n / 3)) ((i - 1) + j / (nl[[i]] + 1))],
   {j, 0, nl[[i]]}], {i, 1, n / 3}];
(* calculate lines one after the other *)
```

```
lines = Table[FieldLine[{ex, a Cos[start[[i]]] + xe},
    {ey, b Sin[start[[i]]] + ye}], {i, 1, Length[start]}];
(* add single lines where appropriate *)
(*line1=FieldLine[{ex,25},{ey,34}];
 AppendTo[lines,line1];line2=FieldLine[{ex,34},{ey,34}];
 AppendTo[lines,line2];*)
arrows = Map[AddArrow[#, 6] &, lines];
(* r1,r2 : electrodes, r3: dielectrics *)
r1 = Rectangle[{xl1, yd1}, {xr1, yu1}];
r2 = Rectangle[{xl2, yd2}, {xr2, yu2}];
r3 = Rectangle[{xl3, yd3}, {xr3, yu3}];
r4 = Rectangle[{xl4, yd4}, {xr4, yu4}];
r5 = {Line[{{exl, eyd}, {exl, eyu}}],
  Line[{{exl, eyu}, {exr, eyu}}], Line[
   {{exr, eyu}, {exr, eyd}}], Line[{{exr, eyd}, {exl, eyd}}]};
Show[Graphics[{GrayLevel[0.8], Thickness[0.015], r5}],
 cont, Graphics[{lines, arrows, r1, r2, r3, r4}],
 ImageSize → 400, Frame → True,
 DisplayFunction → $DisplayFunction]
```

- ContourGraphics -



- Graphics -

Field lines and equipotential lines of the pigtail, the two conductors to the left on potential 1, the others on 0.

```
g = Table[{u[[i, j + 1]] - u[[i, j]], u[[i + 1, j]] - u[[i, j]]},
   {i, 1, nz - 1}, {j, 1, ns - 1}];
lrow = Table[{u[[nz, j + 1]] - u[[nz, j]],
    u[[nz, j]] - u[[nz - 1, j]]}, {j, 1, ns - 1}];
g = Append[g, lrow];
lcol = Append[Table[{u[[i, ns]] - u[[i, ns - 1]],
     u[[i + 1, ns]] - u[[i, ns]]}, {i, 1, nz - 1}],
   {u[[nz, ns]] - u[[nz, ns - 1]], u[[nz, ns]] - u[[nz - 1, ns]]}];
g = Transpose[Append[Transpose[g], lcol]];
```

2.58572×10^{-10}

The following results were obtained in runs with the parameters vi as specified :

$\tilde{C}_{11}$ = 129.529 [pF/m] (v1 = 1, v2 = 0, v3 = 0, v4 = 0)
$\tilde{C}_{22}$ = 129.527 [pF/m] (v1 = 0, v2 = 1, v3 = 0, v4 = 0)
$\tilde{C}_{33}$ = 129.527 [pF/m] (v1 = 0, v2 = 0, v3 = 1, v4 = 0)
$\tilde{C}_{44}$ = 129.529 [pF/m] (v1 = 0, v2 = 0, v3 = 0, v4 = 1)
$\tilde{C}\tilde{C}_{12}$ = 258.572 [pF/m] (v1 = 1, v2 = 1, v3 = 0, v4 = 0) $\overset{(28)}{\rightarrow}$ $\tilde{C}_{12}$ = - 0.242 [pF/m]
$\tilde{C}\tilde{C}_{23}$ = 258.569 [pF/m] (v1 = 0, v2 = 1, v3 = 1, v4 = 0) $\rightarrow$ $\tilde{C}_{23}$ = - 0.242 [pF/m]
$\tilde{C}\tilde{C}_{34}$ = 258.572 [pF/m] (v1 = 0, v2 = 0, v3 = 1, v4 = 1) $\rightarrow$ $\tilde{C}_{34}$ = - 0.242 [pF/m]
$\tilde{C}\tilde{C}_{13}$ = 259.049 [pF/m] (v1 = 1, v2 = 0, v3 = 1, v4 = 0) $\rightarrow$ $\tilde{C}_{13}$ = - 0.0035 [pF/m]
$\tilde{C}\tilde{C}_{24}$ = 259.049 [pF/m] (v1 = 0, v2 = 1, v3 = 0, v4 = 1) $\rightarrow$ $\tilde{C}_{24}$ = - 0.0035 [pF/m]
$\tilde{C}\tilde{C}_{14}$ = 259.059 [pF/m] (v1 = 1, v2 = 0, v3 = 0, v4 = 1) $\rightarrow$ $\tilde{C}_{14}$ = 0 [pF/m]

Using (24) the coefficients of capacity come out to be

C_{10} = 129.28 [pF/m] , C_{20} = 129.04 [pF/m] , C_{30} = 129.04 [pF/m] , C_{40} = 129.28 [pF/m] ,
C_{12} = 0.242 [pF/m] , C_{23} = 0.242 [pF/m] , C_{34} = 0.242 [pF/m] ,
C_{13} = 0.0035 [pF/m] , C_{24} = 0.0035 [pF/m] ,
C_{14} = 0 [pF/m]

All ten coefficients of capacity form a five node network (Fig. 8). To describe two differential pairs we need to combine the ten into six effective capacities $\hat{C}_{ij}$ between the nodes 1 to 4, to each of which all ten coefficients will contribute. $\hat{C}_{12}$ of the first differential line (C_1 ' in Fig. 1) is the whole capacity from node 1 to 2. $\hat{C}_{34}$ of the second differential line (C_2 ' in Fig. 1) is the total capacity from node 3 to 4. For the coupling capacity (C_{12} ' in Fig. 1) there is a special twist: Any change of the potential on node 1 will couple via $\hat{C}_{13}$ to node 3 and via $\hat{C}_{14}$ to node 4 and is (because of the dominating C_{i0}'s) hardly seen as a voltage on the second pair (common mode rejection of the differential line), the same can be said for changes of potential on node 2; therefore the coupling is given by

$$C_{12}' = \hat{C}_{23} - \hat{C}_{24} - (\hat{C}_{13} - \hat{C}_{14}). \qquad (45)$$

This completes the expression of all three capacities found in Fig. 1 through the $\hat{C}_{ij}$ of the pigtail. Their values will be used in the code example below.

Now in order to find the effective capacity $\hat{C}_{ij}$ between two nodes i and j we reduce the network node by node through replacing successively star-type nodes with box-type nets, which are then parallel to the rest of the network, until there are only two nodes left (**Ref. 7**, p.31f). As you will see, every reduction blows up the resulting expression quite a bit, so the symbolic capability of *Mathematica* comes to good use here.

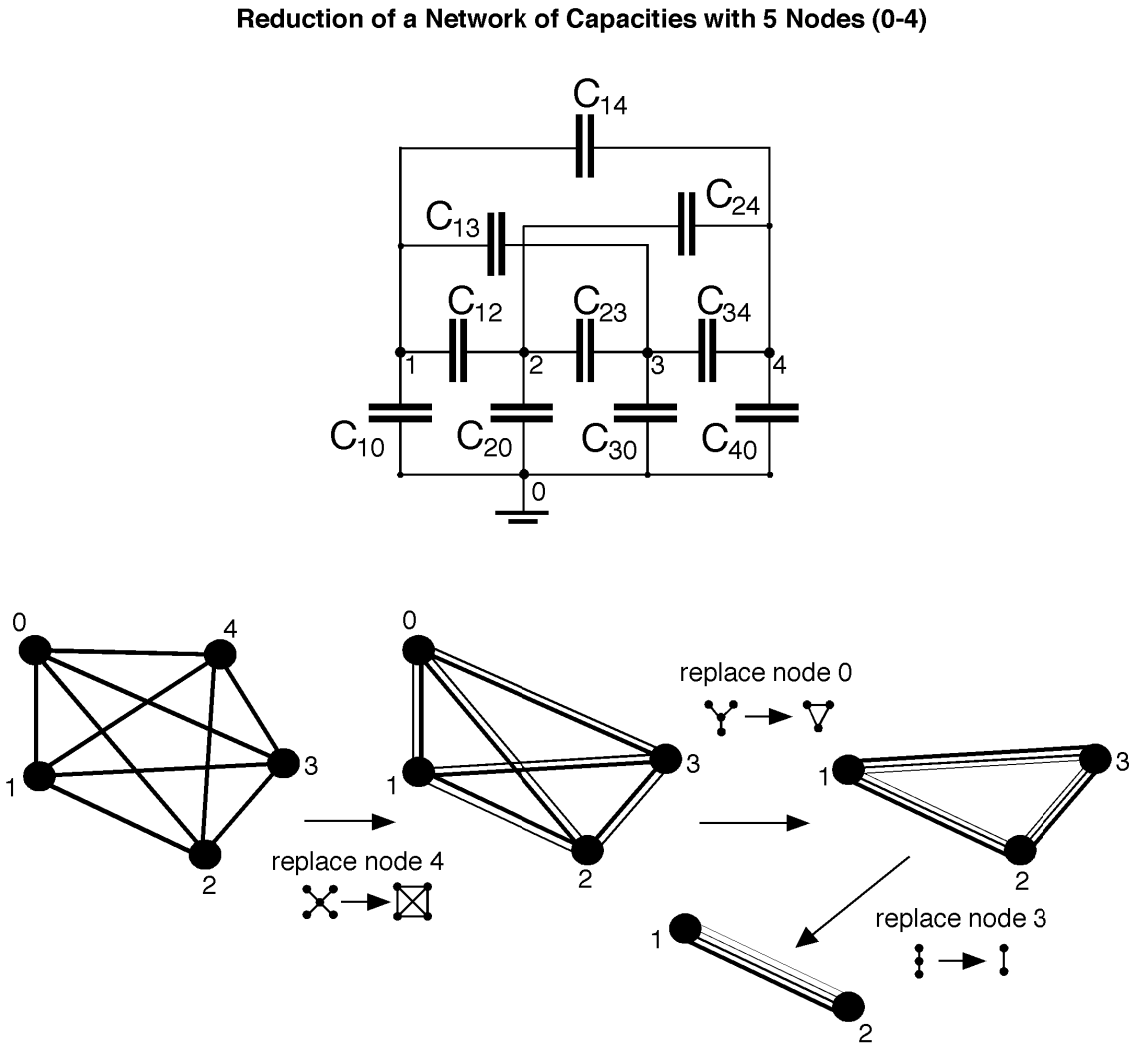


Figure 8

Figure 8. Upper part: Most general capacitive network consisting of 5 nodes (0-4).
Lower part: Successive reduction to two nodes

Equation (45) shows how the new box impedance connecting nodes μ and ν is expressed by all the star legs:

$$Z_{\mu,\nu} = Z_{\mu,r}\, Z_{\nu,r} \sum_{\lambda=1}^{n} \frac{1}{Z_{\lambda,r}}, \tag{46}$$

where n is the number of legs of the star-type node r to be replaced.

The cell below contains an auxiliary function to calculate parallel impedances.

```
par[z1_, z2_] := z1 z2 / (z1 + z2)
```

We get 6, 3 and at last 1 expression corresponding to the thin lines in the lower part of Fig. 8.

Replace node 4 :

```
zz10 = par[z10, z14 z40 (1/z40 + 1/z14 + 1/z24 + 1/z34)] // Simplify
```

$$\frac{z10\,(z24\,z34\,z40 + z14\,(z34\,z40 + z24\,(z34 + z40)))}{z10\,z24\,z34 + z24\,z34\,z40 + z14\,(z24\,z34 + z24\,z40 + z34\,z40)}$$

```
zz20 = par[z20, z24 z40 (1/z40 + 1/z14 + 1/z24 + 1/z34)] // Simplify
```

$$\frac{z20\,(z24\,z34\,z40 + z14\,(z34\,z40 + z24\,(z34 + z40)))}{z24\,z34\,z40 + z14\,(z20\,z34 + z34\,z40 + z24\,(z34 + z40))}$$

```
zz30 = par[z30, z34 z40 (1/z40 + 1/z14 + 1/z24 + 1/z34)] // Simplify
```

$$\frac{z30\,(z24\,z34\,z40 + z14\,(z34\,z40 + z24\,(z34 + z40)))}{z24\,z34\,z40 + z14\,(z34\,z40 + z24\,(z30 + z34 + z40))}$$

```
zz12 = par[z12, z14 z24 (1/z40 + 1/z14 + 1/z24 + 1/z34)] // Simplify
```

$$\frac{z12\,(z24\,z34\,z40 + z14\,(z34\,z40 + z24\,(z34 + z40)))}{(z12 + z24)\,z34\,z40 + z14\,(z34\,z40 + z24\,(z34 + z40))}$$

```
zz13 = par[z13, z14 z34 (1/z40 + 1/z14 + 1/z24 + 1/z34)] // Simplify
```

$$\frac{z13\,(z24\,z34\,z40 + z14\,(z34\,z40 + z24\,(z34 + z40)))}{z24\,(z13 + z34)\,z40 + z14\,(z34\,z40 + z24\,(z34 + z40))}$$

```
zz23 = par[z23, z24 z34 (1/z40 + 1/z14 + 1/z24 + 1/z34)] // Simplify
```

$$\frac{z23\,(z24\,z34\,z40 + z14\,(z34\,z40 + z24\,(z34 + z40)))}{z24\,z34\,z40 + z14\,((z23 + z34)\,z40 + z24\,(z34 + z40))}$$

Replace node 0:

```
zzz12 = par[zz12, zz10 zz20 (1/zz10 + 1/zz20 + 1/zz30)] // Simplify
```

```
General::spell1 : Possible spelling error: new
   symbol name "zzz12" is similar to existing symbol "zz12".
```

```
(z12 (z20 z30 (z24 z34 z40 + z14 (z34 z40 + z24 (z34 + z40))) +
      z10 (z24 z34 (z30 z40 + z20 (z30 + z40)) +
         z14 (z30 (z34 z40 + z24 (z34 + z40)) +
            z20 (z34 (z30 + z40) + z24 (z30 + z34 + z40))))) /
 (z30 (z20 (z24 z34 z40 + z14 (z34 z40 + z24 (z34 + z40))) +
      z12 ((z20 + z24) z34 z40 +
         z14 (z20 z34 + z34 z40 + z24 (z34 + z40)))) +
   z10 (z34 (z24 (z30 z40 + z20 (z30 + z40)) +
         z12 (z30 (z24 + z40) + z20 (z30 + z40))) +
      z14 (z30 (z34 z40 + z24 (z34 + z40)) +
         z20 (z34 (z30 + z40) + z24 (z30 + z34 + z40)))))
```

```
zzz13 = par[zz13, zz10 zz30 (1/zz10 + 1/zz20 + 1/zz30)] // Simplify
```

```
General::spell1 : Possible spelling error: new
   symbol name "zzz13" is similar to existing symbol "zz13".
```

```
(z13 (z20 z30 (z24 z34 z40 + z14 (z34 z40 + z24 (z34 + z40))) +
      z10 (z24 z34 (z30 z40 + z20 (z30 + z40)) +
         z14 (z30 (z34 z40 + z24 (z34 + z40)) +
            z20 (z34 (z30 + z40) + z24 (z30 + z34 + z40))))) /
 (z20 (z30 (z24 z34 z40 + z14 (z34 z40 + z24 (z34 + z40))) + z13
       (z24 (z30 + z34) z40 + z14 (z34 z40 + z24 (z30 + z34 + z40)))) +
   z10 (z24 z34 (z30 z40 + z20 (z30 + z40)) +
      z13 z24 (z30 z40 + z20 (z30 + z34 + z40)) +
      z14 (z30 (z34 z40 + z24 (z34 + z40)) +
         z20 (z34 (z30 + z40) + z24 (z30 + z34 + z40)))))
```

```
zzz23 = par[zz23, zz20 zz30 (1/zz10 + 1/zz20 + 1/zz30)] // Simplify
```

```
General::spell1 : Possible spelling error: new
   symbol name "zzz23" is similar to existing symbol "zz23".
```

```
(z23 (z20 z30 (z24 z34 z40 + z14 (z34 z40 + z24 (z34 + z40))) +
      z10 (z24 z34 (z30 z40 + z20 (z30 + z40)) +
         z14 (z30 (z34 z40 + z24 (z34 + z40)) +
            z20 (z34 (z30 + z40) + z24 (z30 + z34 + z40))))) /
 (z20 z30 (z24 z34 z40 + z14 ((z23 + z34) z40 + z24 (z34 + z40))) +
   z10 (z24 z34 ((z23 + z30) z40 + z20 (z30 + z40)) +
      z14 (z30 (z34 z40 + z24 (z34 + z40)) + z23 ((z30 + z34) z40 +
            z24 (z30 + z34 + z40)) + z20 (z34 (z30 + z40) +
            z23 (z30 + z34 + z40) + z24 (z30 + z34 + z40)))))
```

Replace node 3:

```
zzzz12 = par[zzz12, zzz13 zzz23 (1/zzz13 + 1/zzz23)] // Simplify
```

```
— General::spell1 : Possible spelling error: new
    symbol name "zzzz12" is similar to existing symbol "zzz12".
```

```
(z12 (z20 (z23 z30 (z24 z34 z40 + z14 (z34 z40 + z24 (z34 + z40))) +
        z13 (z24 (z30 z34 + z23 (z30 + z34)) z40 +
           z14 (z30 (z34 z40 + z24 (z34 + z40)) +
              z23 ((z30 + z34) z40 + z24 (z30 + z34 + z40))))) +
     z10 (z23 (z24 z34 (z30 z40 + z20 (z30 + z40)) +
           z14 (z30 (z34 z40 + z24 (z34 + z40)) +
              z20 (z34 (z30 + z40) + z24 (z30 + z34 + z40)))) +
        z13 (z24 ((z30 z34 + z23 (z30 + z34)) z40 +
              z20 (z34 (z30 + z40) + z23 (z30 + z34 + z40))) +
           z14 (z30 (z34 z40 + z24 (z34 + z40)) + z23 ((z30 + z34) z40 +
                 z24 (z30 + z34 + z40)) + z20 (z34 (z30 + z40) +
                 z23 (z30 + z34 + z40) + z24 (z30 + z34 + z40)))))))) /
 (z20 (z23 z30 (z24 z34 z40 + z14 (z34 z40 + z24 (z34 + z40))) +
      z13 (z24 (z30 z34 + z23 (z30 + z34)) z40 +
         z14 (z30 (z34 z40 + z24 (z34 + z40)) +
            z23 ((z30 + z34) z40 + z24 (z30 + z34 + z40))))) +
   z12 (z30 ((z23 z24 + z20 (z23 + z24)) z34 z40 +
         z14 (z23 (z34 z40 + z24 (z34 + z40)) +
            z20 (z34 z40 + z23 (z34 + z40) + z24 (z34 + z40)))) +
      z13 ((z24 (z30 z34 + z23 (z30 + z34)) + z20
            (z30 z34 + z23 (z30 + z34) + z24 (z30 + z34))) z40 +
         z14 (z30 (z34 z40 + z24 (z34 + z40)) + z23 ((z30 + z34)
               z40 + z24 (z30 + z34 + z40)) + z20 (z34 (z30 + z40) +
              z23 (z30 + z34 + z40) + z24 (z30 + z34 + z40))))) +
   z10 (z23 (z24 z34 (z30 z40 + z20 (z30 + z40)) +
         z14 (z30 (z34 z40 + z24 (z34 + z40)) +
            z20 (z34 (z30 + z40) + z24 (z30 + z34 + z40)))) +
      z13 (z24 ((z30 z34 + z23 (z30 + z34)) z40 +
            z20 (z34 (z30 + z40) + z23 (z30 + z34 + z40))) +
         z14 (z30 (z34 z40 + z24 (z34 + z40)) + z23 ((z30 + z34)
               z40 + z24 (z30 + z34 + z40)) + z20 (z34 (z30 + z40) +
              z23 (z30 + z34 + z40) + z24 (z30 + z34 + z40))))) +
      z12 (z34 (z24 z30 z40 + z20 (z23 + z24) (z30 + z40) +
            z23 (z30 z40 + z24 (z30 + z40))) +
         z13 (z30 (z34 z40 + z24 (z34 + z40)) + z23 ((z30 + z34)
               z40 + z24 (z30 + z34 + z40)) + z20 (z34 (z30 + z40) +
              z23 (z30 + z34 + z40) + z24 (z30 + z34 + z40))) +
         z14 (z30 (z34 z40 + z24 (z34 + z40)) + z23 ((z30 + z34)
               z40 + z24 (z30 + z34 + z40)) + z20 (z34 (z30 + z40) +
              z23 (z30 + z34 + z40) + z24 (z30 + z34 + z40))))))
```

After entering the capacity values for the impedances, we get $\hat{C_{12}} = C_1'$ from (3) for the first differential line!

```
z10 = 1 / 129.28; z12 = 1 / 0.242; z13 = 1 / 0.0035; z14 = 1 / 0.0001;
z20 = 1 / 129.04; z23 = 1 / 0.242; z24 = 1 / 0.0035; z30 = 1 / 129.04;
z34 = 1 / 0.242; z40 = 1 / 129.28; 1 / zzzz12
```

```
64.8842
```

To get $\hat{C_{23}}$ between nodes 2 and 3 the numbers 1 and 3 in the capacitors indices swap places (cf. Fig. 8):

```
z10 = 1 / 129.04; z12 = 1 / 0.242; z13 = 1 / 0.0035; z14 = 1 / 0.242;
z20 = 1 / 129.04; z23 = 1 / 0.242; z24 = 1 / 0.0035; z30 = 1 / 129.28;
z34 = 1 / 0.0001; z40 = 1 / 129.28; 1 / zzzz12
```

```
64.8845
```

Both the capacities $\hat{C_{24}}$ between nodes 2 and 4 (swap numbers 1 and 4) and $\hat{C_{13}}$ between nodes 1 and 3 (swap numbers 2 and 3) are identical :

```
z10 = 1 / 129.28; z12 = 1 / 0.0035; z13 = 1 / 0.242; z14 = 1 / 0.0001;
z20 = 1 / 129.04; z23 = 1 / 0.242; z24 = 1 / 0.242; z30 = 1 / 129.04;
z34 = 1 / 0.0035; z40 = 1 / 129.28; 1 / zzzz12
```

```
64.7649
```

The capacity $\hat{C_{14}}$ between nodes 1 and 4 is (swap numbers 2 and 4):

```
z10 = 1 / 129.28; z12 = 1 / 0.0001; z13 = 1 / 0.0035; z14 = 1 / 0.242;
z20 = 1 / 129.28; z23 = 1 / 0.242; z24 = 1 / 0.0035; z30 = 1 / 129.04;
z34 = 1 / 0.242; z40 = 1 / 129.04; 1 / zzzz12
```

```
64.7626
```

Last is C_2', the capacity of the second differential line, where 1 is swapped for 4 and 2 for 3 (identical to C_1') :

```
z10 = 1 / 129.28; z12 = 1 / 0.242; z13 = 1 / 0.0035; z14 = 1 / 0.0001;
z20 = 1 / 129.04; z23 = 1 / 0.242; z24 = 1 / 0.0035; z30 = 1 / 129.04;
z34 = 1 / 0.242; z40 = 1 / 129.28; 1 / zzzz12
```

```
64.8842
```

The remaining effective coupling *that determines the crosstalk*, evaluates then with (45) to $C_{12}' = 0.1173$ [pF/m].

■ *Mathematica* Code (Coefficients of inductance of the pigtail)

RAM requirements: Front End 20 MB, Kernel 30 MB

Set v1 - v4 such that the resulting current in the loop integral below has the correct value (-1 or 0 or +1).

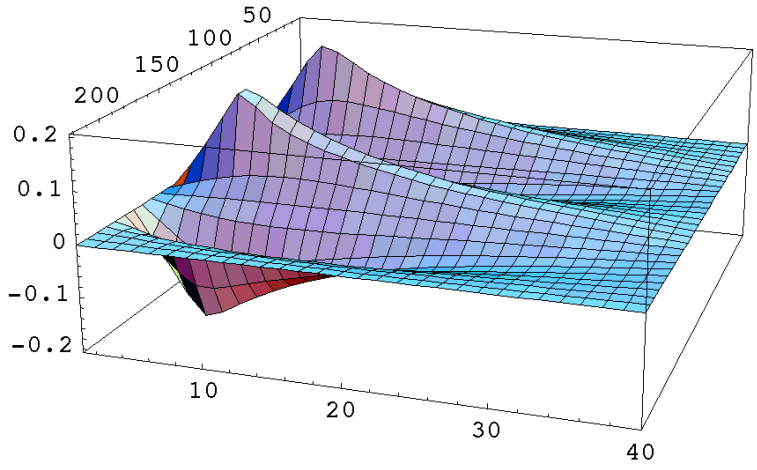
```
v1=0.196581;v2=-0.1947736;v3=0.1947728;v4=-0.196612;
nz=80; (* # of rows of the grid, divisible by 2 *)
ns=446; (* # of columns of the grid, divisible by 2 *)
{xr,yr}={91,12.5};(* center point for contour integral *)
u=.;m=.;m=Array[u,{nz,ns}];a=Array[u,{nz,ns}];
mb=Table[0,{nz},{ns}];
Do[Do[a[[i,j]]=u[[i,j]],{j,1,ns}],{i,1,nz}]
Do[Do[m[[i,j]]:=Evaluate[0.25(a[[i-1,j]]+a[[i,j-1]]+
a[[i+1,j]]+a[[i,j+1]])],
{j,2,ns-1}],{i,2,nz-1}]
Do[m[[1,j]]:=0.,{j,1,ns}]
Do[m[[nz,j]]:=0.,{j,1,ns}]
Do[m[[i,1]]:=0.,{i,1,nz}]
Do[m[[i,ns]]:=0.,{i,1,nz}]
xl1=80;xr1=102;yd1=11;yu1=14;
xl2=168;xr2=190;yd2=11;yu2=14;
xl3=256;xr3=278;yd3=11;yu3=14;
xl4=344;xr4=366;yd4=11;yu4=14;
Do[Do[{m[[i,j]]:=v1;mb[[i,j]]=1;},{j,xl1,xr1}],{i,yd1,yu1}]
Do[Do[{m[[i,j]]:=v2;mb[[i,j]]=1;},{j,xl2,xr2}],{i,yd2,yu2}]
Do[Do[{m[[i,j]]:=v3;mb[[i,j]]=1;},{j,xl3,xr3}],{i,yd3,yu3}]
Do[Do[{m[[i,j]]:=v4;mb[[i,j]]=1;},{j,xl4,xr4}],{i,yd4,yu4}]
```

```
nz1=nz/2;ns1=ns/2;u1=.;m1=.;
m1=Array[u1,{nz1,ns1}];a1=Array[u1,{nz1,ns1}];
Do[Do[a1[[i,j]]=u1[[i,j]],{j,1,ns1}],{i,1,nz1}]
Do[Do[
m1[[i,j]]:=Evaluate[0.25(a1[[i-1,j]]+a1[[i+1,j]]+a1[[i,j-1]]+
a1[[i,j+1]])],
{j,2,ns1-1}],{i,2,nz1-1}];
Do[m1[[1,j]]:=0.,{j,1,ns1}];
Do[m1[[nz1,j]]:=0.,{j,1,ns1}];
Do[m1[[i,1]]:=0.,{i,1,nz1}];
Do[m1[[i,ns1]]:=0.,{i,1,nz1}];
(* guess for coordinates:old/2;mb is used to identify
location of conductors *)
Do[Do[If[mb[[2i,2j]]==1,m1[[i,j]]=m[[2i,2j]]],{i,nz1}],
{j,ns1}]
u1=Table[0.,{i,1,nz1},{j,1,ns1}];
Timing[Do[{u1=m1;
If[Mod[i,500]==0,
Print[i," residual error:",Max[Flatten[Abs[u1-m1]]]]]},
{i,1,2000}]]
```

— *Part::partd : Part specification u1⟦1, 1⟧ is longer than depth of object.*

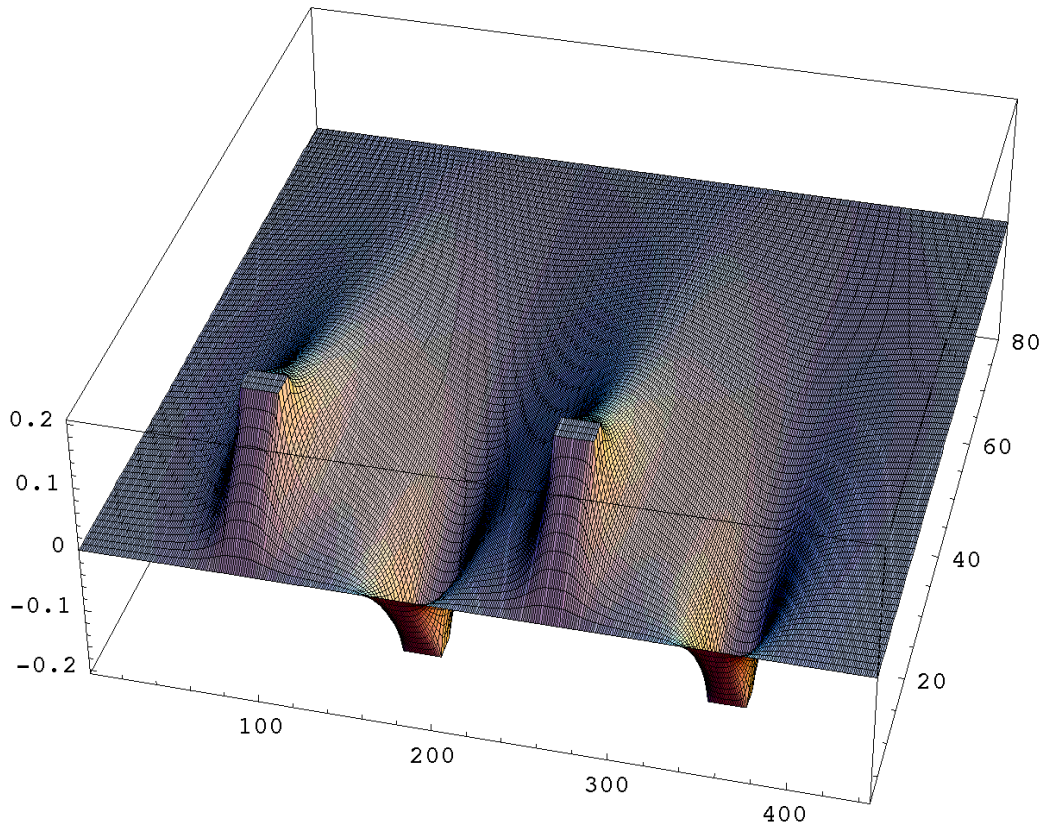
— *General::stop :*
Further output of Part::partd will be suppressed during this calculation.

```
p = ListInterpolation[Transpose[u1], InterpolationOrder → 1];
Plot3D[p[x, y], {x, 1, ns1}, {y, 1, nz1}, PlotRange → All]
```



Control plot of interpolated vector potential.

```
u=Table[p[j/2,i/2],{i,1,nz},{j,1,ns}];Timing[Do[{u=m;
If[Mod[i,100]==0,{err=Max[Abs[Flatten[u-m]]];Print[i,"
residual error: ",err];If[err<10^-6,Break[]]}]},{i,1,7000}]]
ListPlot3D[u,ImageSize->400,ViewPoint->{1,-4.000,3}];
```



Vector potential of the pigtail, a current of strength 1 is flowing back and forth in both circuits.

```
g = Table[{u[[i, j + 1]] - u[[i, j]], u[[i + 1, j]] - u[[i, j]]},
   {i, 1, nz - 1}, {j, 1, ns - 1}];
lrow = Table[{u[[nz, j + 1]] - u[[nz, j]],
    u[[nz, j]] - u[[nz - 1, j]]}, {j, 1, ns - 1}];
g = Append[g, lrow];
lcol = Append[Table[{u[[i, ns]] - u[[i, ns - 1]],
     u[[i + 1, ns]] - u[[i, ns]]}, {i, 1, nz - 1}],
   {u[[nz, ns]] - u[[nz, ns - 1]], u[[nz, ns]] - u[[nz - 1, ns]]}];
g = Transpose[Append[Transpose[g], lcol]];
```

```
μ0 = 4. π * 10^-7;
```

The shape of the conductors now suggests an elliptic contour to calculate the current. The elliptic path of the line integral has to run completely outside of the current. Adjust the half axes a and b accordingly; for high accuracy it is necessary to control *all* 4 currents precisely :

```
dphi = π / 100;
{xr, yr} = {91, 12.5};
(*center point for contour integral 1 *)
i = Table[ Sum[-dphi (
     g[[Round[yr + b * Sin[phi]], Round[xr + a * Cos[phi]], 2]] *
       a * Sin[phi] + g[[Round[yr + b * Sin[phi]],
         Round[xr + a * Cos[phi]], 1]] * b * Cos[phi]),
   {phi, 0, 2 π, dphi}], {a, 13, 20}, {b, 5, 10}]
iMean1 = Apply[Plus, Flatten[i]] / Length[Flatten[i]]
```

```
{{1.00863, 1.00161, 1.00454, 1.00039, 0.998661, 0.999783},
 {1.00281, 1.00465, 1.00212, 1.00047, 0.999056, 0.99995},
 {0.998864, 1.00062, 1.00015, 0.99813, 0.997783, 0.998637},
 {1.00116, 1.00143, 1.00126, 0.999359, 1.00027, 1.00031},
 {1.00394, 1.00344, 1.00205, 0.999937, 1.0006, 0.999843},
 {1.00242, 1.00103, 0.998707, 0.997314, 0.998403, 0.99755},
 {0.996741, 1.00119, 0.999394, 0.998253, 0.99902, 0.998298},
 {0.995873, 0.999232, 0.997195, 0.995986, 0.996848, 0.996145}}
```

```
1.
```

```
dphi = π / 100;
{xr, yr} = {179, 12.5};
(*center point for contour integral 2 *)
i = Table[ Sum[-dphi (
     g[[Round[yr + b * Sin[phi]], Round[xr + a * Cos[phi]], 2]] *
       a * Sin[phi] + g[[Round[yr + b * Sin[phi]],
         Round[xr + a * Cos[phi]], 1]] * b * Cos[phi]),
   {phi, 0, 2 π, dphi}], {a, 13, 20}, {b, 5, 10}]
iMean2 = Apply[Plus, Flatten[i]] / Length[Flatten[i]]
```

```
{{-1.00867, -1.00161, -1.00454, -1.00035, -0.998615, -0.999734},
 {-1.00283, -1.00465, -1.00213, -1.00046, -0.999033, -0.999927},
 {-0.998885, -1.00063, -1.00015,
  -0.998121, -0.997766, -0.998623},
 {-1.00116, -1.00144, -1.00126, -0.999349, -1.00025, -1.0003},
 {-1.00395, -1.00345, -1.00206, -0.999929, -1.00059, -0.999831},
 {-1.00243, -1.00104, -0.998716, -0.997309, -0.998397,
  -0.997544}, {-0.996769, -1.00121, -0.999405,
  -0.998249, -0.999015, -0.998294}, {-0.995898,
  -0.999252, -0.997209, -0.995986, -0.996848, -0.996147}}
```

```
-1.
```

```
dphi = π / 100;
{xr, yr} = {267, 12.5};
(*center point for contour integral 3 *)
i = Table[ Sum[-dphi (
     g[[Round[yr + b * Sin[phi]], Round[xr + a * Cos[phi]], 2]] *
       a * Sin[phi] + g[[Round[yr + b * Sin[phi]],
         Round[xr + a * Cos[phi]], 1]] * b * Cos[phi]),
   {phi, 0, 2 π, dphi}], {a, 13, 20}, {b, 5, 10}]
iMean3 = Apply[Plus, Flatten[i]] / Length[Flatten[i]]
```

```
{{1.00867, 1.00161, 1.00454, 1.00035, 0.998616, 0.999736},
 {1.00283, 1.00465, 1.00213, 1.00046, 0.999034, 0.999928},
 {0.998884, 1.00062, 1.00015, 0.998121, 0.997767, 0.998624},
 {1.00116, 1.00144, 1.00126, 0.99935, 1.00025, 1.0003},
 {1.00395, 1.00345, 1.00206, 0.999929, 1.00059, 0.999832},
 {1.00243, 1.00104, 0.998716, 0.99731, 0.998398, 0.997545},
 {0.996768, 1.00121, 0.999405, 0.99825, 0.999015, 0.998295},
 {0.995897, 0.999252, 0.997209, 0.995987, 0.996849, 0.996148}}
```

```
1.
```

```
dphi = π / 100;
{xr, yr} = {355, 12.5};
(*center point for contour integral 4 *)
i = Table[ Sum[-dphi (
     g[[Round[yr + b * Sin[phi]], Round[xr + a * Cos[phi]], 2]] *
       a * Sin[phi] + g[[Round[yr + b * Sin[phi]],
         Round[xr + a * Cos[phi]], 1]] * b * Cos[phi]),
   {phi, 0, 2 π, dphi}], {a, 13, 20}, {b, 5, 10}]
iMean4 = Apply[Plus, Flatten[i]] / Length[Flatten[i]]
```

```
{{-1.0087, -1.00165, -1.00456, -1.00037, -0.998629, -0.999733},
 {-1.00286, -1.00467, -1.00214, -1.00047, -0.99904, -0.99992},
 {-0.99892, -1.00065, -1.00016, -0.998135, -0.997771, -0.998613},
 {-1.0012, -1.00147, -1.00128, -0.999367, -1.00025, -1.00029},
 {-1.00398, -1.00346, -1.00207, -0.999936, -1.00058, -0.999813},
 {-1.00245, -1.00105, -0.998717, -0.997308, -0.99838, -0.997513},
 {-0.996784, -1.00121, -0.999398, -0.998239,
  -0.998989, -0.998252}, {-0.995908, -0.999245,
  -0.997191, -0.995964, -0.996808, -0.996088}}


-1.
```

If all currents have their required values, then the inductance is:

```
ind = μ0 * Sum[g[[i, j]].g[[i, j]], {i, 1, nz}, {j, 1, ns}]
```

9.82397×10^{-7}

Results for the external inductances from the three pairs of currents {1,0},{0,1},and {1,1}:
L_1' = 0.493248 [μH/m] (v1 = 0.196541, v2 = - 0.196598, v3 = - 0.0018, v4 = - 0.00004)
L_2' = 0.493260 [μH/m] (v1 = 0.000039, v2 = 0.001824, v3 = 0.196572, v4 = - 0.196572)
LL_{12}' = 0.982397 [μH/m] (v1 = 0.196581, v2 = - 0.1947736, v3 = 0.1947728, v4 = - 0.196612)

The internal inductance amounts to $\frac{\mu_0}{8\pi} = 5 * 10^{-8}$ [H/m] per conductor (**Ref. 7**, p.320) and the complete result is:
L_1' = 0.593248 [μH/m], L_2' = 0.593260 [μH/m], LL_{12}' = 1.182397 [μH/m]

With (37) there is then the mutual inductance *that is responsible for the crosstalk*:

L_{12}' = - 3.34 [nH/m].

The negative sign can be understood from the following argument: conductor 1 generates with current +1 a certain amount of flux between conductors 3 and 4. The flux originating from conductor 2 with current in opposite direction is closer to the region between conductor 3 and 4 and has therefore (because it depends logarithmically on the distance) in this place a bigger impact, so that the addition of these two fluxes will give a negative number.

The obtained values for the capacities and inductances will now be used in the following code to calculate the propagating pulse.

■ *Mathematica* Code (Animation of Double Pulse including Crosstalk)

RAM requirements: Front End 60 MB, Kernel 20 MB

```
npl = 300;(* number of plots *)
dt = 1. * 10^-12; (* 1 psec *)
dx = 0.004; (* 4 mm *)
(*dx/dt≫speed of propagation for stability,
 trade-off with computational speed*)
m = 250; (* array dimension → space distance m*dx = 1 m *)
nmax = 50000; (* time considered: nmax * dt  *)
pl = Round[nmax / npl];
(* snapshot after every pl timesteps *)
k = 2; (* number of circuits *)
(* define input pulse *)
rise = Round[1. * 10^-9 / dt];
width = Round[10.5 * 10^-9 / dt];
fall = Round[1. * 10^-9 / dt];
emp = Round[12.5 * 10^-9 / dt];
len = rise + width + fall + emp + rise + width + fall;
uin = Table[1. * j / rise, {j, 1, rise}];
uin = Append[uin, Table[1., {j, 1, width}]];
uin = Append[uin, Table[1. - j / fall, {j, 1, fall}]];
uin = Append[uin, Table[0., {j, 1, emp}]];
uin = Append[uin, Table[1. * j / rise, {j, 1, rise}]];
uin = Append[uin, Table[1., {j, 1, width}]];
uin = Append[uin, Table[1. - j / fall, {j, 1, fall}]];
uin = Append[uin, Table[0., {j, 1, nmax - len}]];
uin = Flatten[uin];
```

Parameters per Unit Length:

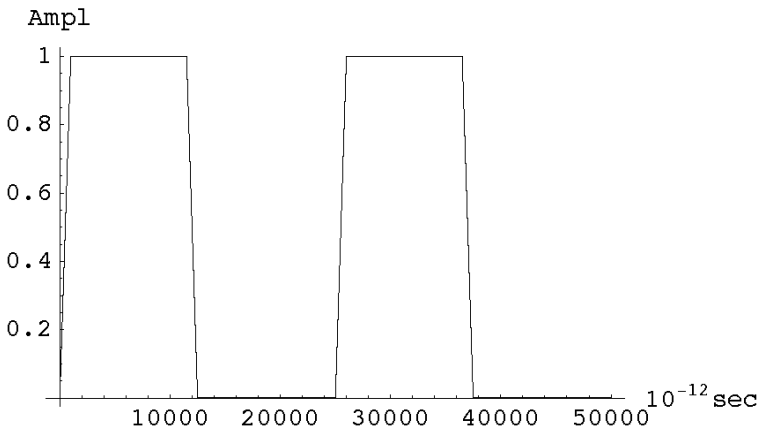
```
res = 8.;  (* resistance in Ω per meter *)
rm = ( res   0
       0   res );
 (* capacitance in F per meter  *)
cm = (  65.    -0.117
       -0.117   65.  ) * 10^-12; (* now incl crosstalk *)
ci = Inverse[cm];
 (* inductance in H per meter  *)
lm = (  0.59325  -0.00334
       -0.00334   0.59326 ) * 10^-6;(* now incl crosstalk *)
li = Inverse[lm];
zl = 220; (* Termination left  *)
zr = 220;(* Termination right  *)
```

Initialization and Setup of Coefficient Matrices:

```
id = IdentityMatrix[k];
u1 = u0 = Table[0., {m}, {k}]; i1 = i0 = Table[0., {m}, {k}];
m1 = (id - (dt/dx)^2 ci.li); m2 = 1/2 (dt/dx)^2 (ci.li); m3 = -1/2 (dt/dx) ci;
m4 = (id - dt li.rm).(id - (dt/dx)^2 li.ci);
m5 = 1/2 (dt/dx)^2 (id - dt li.rm).(li.ci);
m6 = -1/2 (dt/dx) (id - dt li.rm).li;
m8 = Inverse[id - (dt/dx)^2 ci.li].(id + (dt/dx)^2 ci.li);
m9 = 2 dt/dx Inverse[id - (dt/dx)^2 ci.li].ci;
m10 = id - dt li.rm;
m11 = 2 dt/dx (id - dt li.rm).Inverse[id - (dt/dx)^2 li.ci].li;
m12 =
  (id - dt li.rm).Inverse[id - (dt/dx)^2 li.ci].(id + (dt/dx)^2 li.ci);
(* output main and 2 adjacent wires *)
uout = {0, 0}; uout1 = {0, 0}; uout2 = {0, 0};
```

Static Initial Condition:

```
ListPlot[uin, PlotJoined → True,
  AxesLabel -> {"10^-12 sec", "Ampl"}];
```



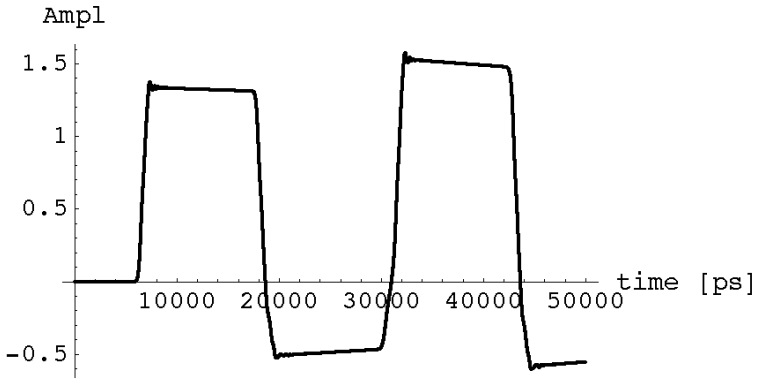
Input Pulse on the left side of the pigtail,two pulses of 12.5 nsec length and 12.5 nsec separation.

Calculate evolution in time and produce animation of the propagating pulses:

```
Timing[
 Do[
  {(* main loop, calculate line n+1 in u and i grid *)
   Do[{jp1 = j + 1; jm1 = j - 1;
     u1[[j]] = m1.u0[[j]] +
       m2.(u0[[jp1]] + u0[[jm1]]) + m3.(i0[[jp1]] - i0[[jm1]]);
     i1[[j]] = m4.i0[[j]] + m5.(i0[[jp1]] + i0[[jm1]]) +
       m6.(u0[[jp1]] - u0[[jm1]])}, {j, 2, m - 1}];
   (* edges *)
   u1[[m]] = u0[[m - 1]] +
     m8.(u0[[m]] - u1[[m - 1]]) - m9.(i0[[m]] - i1[[m - 1]]);
   i1[[m]] = m10.i0[[m - 1]] - m11.(u0[[m]] - u1[[m - 1]]) +
     m12.(i0[[m]] - i1[[m - 1]]);
   u1[[1]] = u0[[2]] + m8.(u0[[1]] - u1[[2]]) +
     m9.(i0[[1]] - i1[[2]]);
   i1[[1]] = m10.i0[[2]] + m11.(u0[[1]] - u1[[2]]) +
     m12.(i0[[1]] - i1[[2]]);
   (* static boundary conditions  open: set i=0,
    short: set u=0 *)
   u1[[m]] = zr i1[[m]]; (* termination right with z *)
   u1[[1]] = -zl i1[[1]]; (* termination left with -z *)
   u1[[1, 1]] = uin[[n]];
   (* build output *)
   uout = Append[uout, u1[[m, 1]]];
   (* seen on diff line 1 after 1 m *)
   uout1 = Append[uout1, u1[[m, 2]]];
   (* crosstalk on diff line 2 after 1 m *)
   uout2 = Append[uout2, Max[Transpose[u1][[2]]]];
   (* max ampl of crosstalk *)
   u0 = u1; i0 = i1; (* advance one step in time *)
   (* do a snapshot of the voltages *)
   If[Mod[n, pl] == 0,
    {(* plot voltage wire 1 *)
     p1 = ListPlot[Transpose[u1][[1]],
       PlotRange → {{0, m}, {-0.6, 1.7}},
       PlotJoined → True, DisplayFunction → Identity];
     (* plot current wire 1 *)
     p2 = ListPlot[Transpose[i1][[1]],
       PlotRange → {{0, m}, {-0.015, 0.015}},
       PlotJoined → True, DisplayFunction → Identity];
     (* plot voltage wire 2 *)
     p3 = ListPlot[Transpose[u1][[2]],
       PlotRange → {{0, m}, {-0.039, 0.039}},
       PlotJoined → True, DisplayFunction → Identity];
     Show[GraphicsArray[{{p1}, {p2}, {p3}}],
      DisplayFunction → $DisplayFunction];
     (* get rid of small numbers to
       accelerate calculation *)
     u0 = Chop[u0]; i0 = Chop[i0]; Print[n]}]}, {n, nmax}]
]
Print["Fertig !"]
```

This shows the voltage between line 1 and 2, sampled at the right end of the pigtail (1m), as it would appear on a scope – leading edge is to the left – :

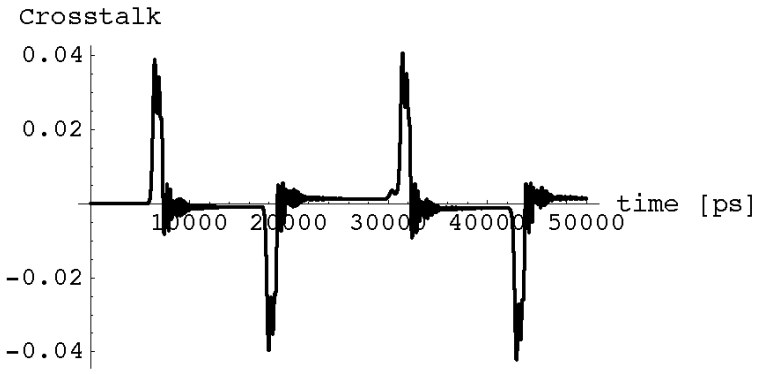
```
ListPlot[uout, AxesLabel -> {"time [ps]", "Ampl"}]
```




```
- Graphics -
```

Below is shown the corresponding crosstalk on lines 3 and 4, sampled at the right end of the pigtail, as it would appear on a scope:

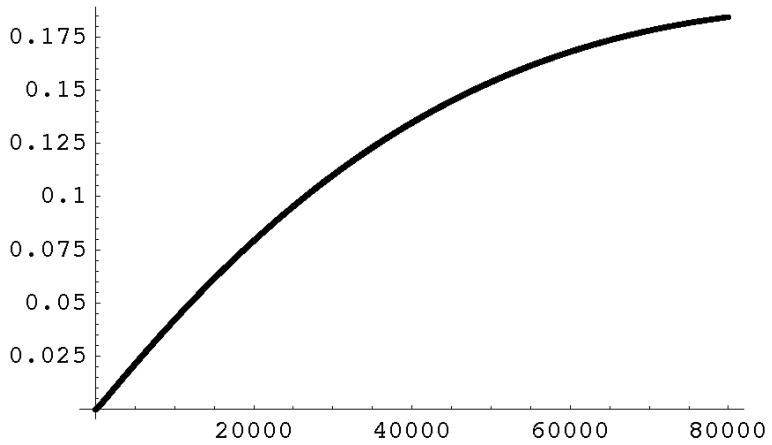
```
ListPlot[uout1, PlotRange → All,
 AxesLabel -> {"time [ps]", "Crosstalk"}]
```




```
- Graphics -
```

In both 'scope' pictures above the displayed pulses already contain a reflected part from the right end, where they are sampled. This is why – you see it clearly in the animation – the second pulse in the first picture is larger and has a lower baseline.

In the next plot we see the amplitude of the crosstalk signal as function of time during its motion from left to right – it is shown before it has reached the right end and is therefore not yet disturbed by reflection, because the simulated wires in this special run were more than 12 m long – as it rises from zero to its steady state value (horizontal unit is 1 ps):



Why might the crosstalk be increasing ? For an explanation look at Fig. 9 (**Ref. 11**, p. 206) :

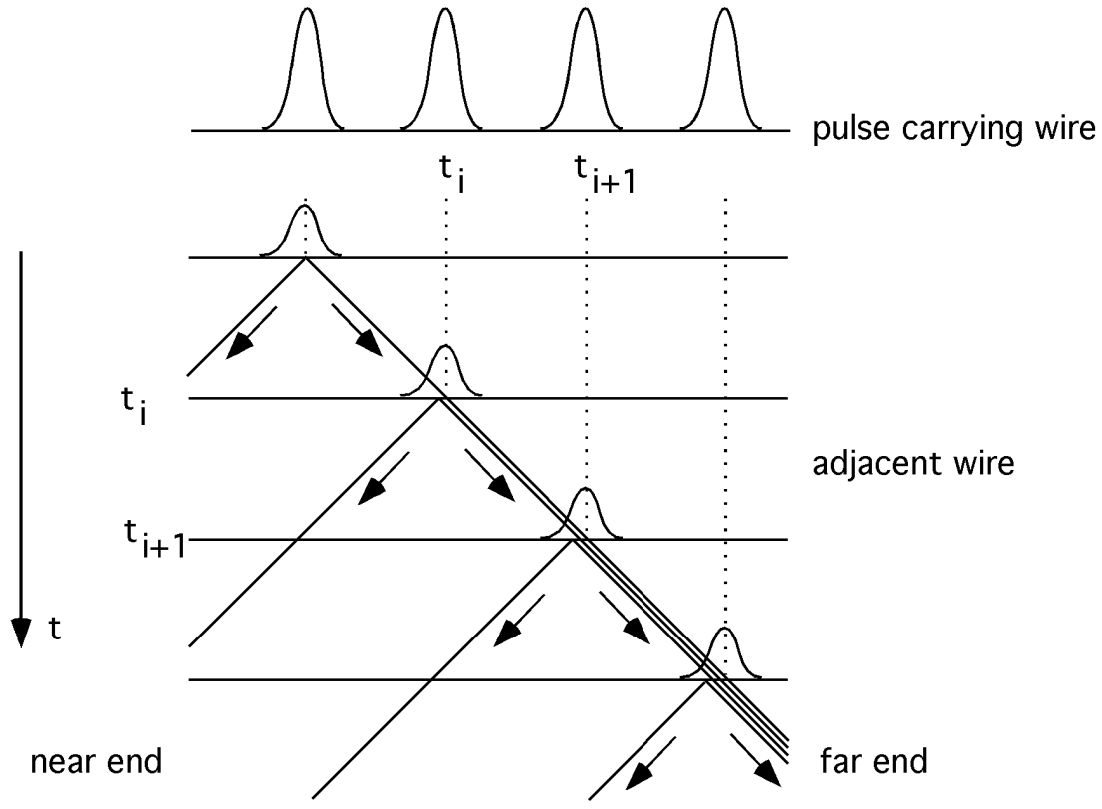
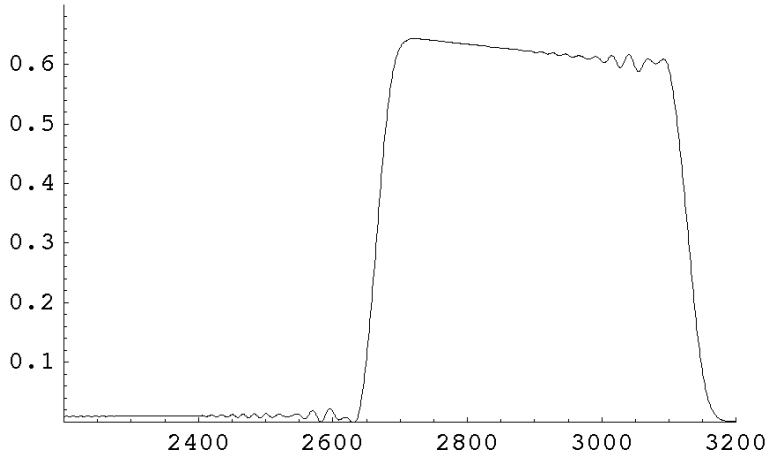


Figure 9

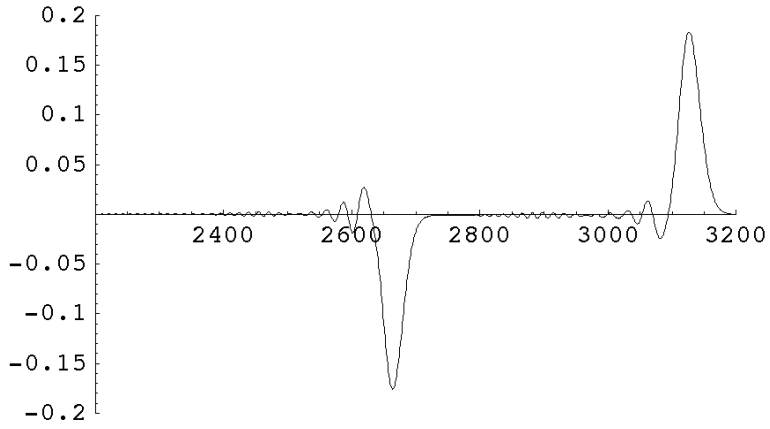
Figure 9. Accumulation of far end crosstalk

At each instant in time some portion of the main pulse is coupled to the adjacent wire, where it starts propagating to both ends. As the main pulse has one direction (here: to the right) and travels with the same speed as the crosstalk portions, everything that goes to the far end arrives at the same time, while the parts going to the left are distributed in time and see the near end only one after the other. This process continues until a likewise growing part of the crosstalk starts flowing back to the main wire and an equilibrium is reached.

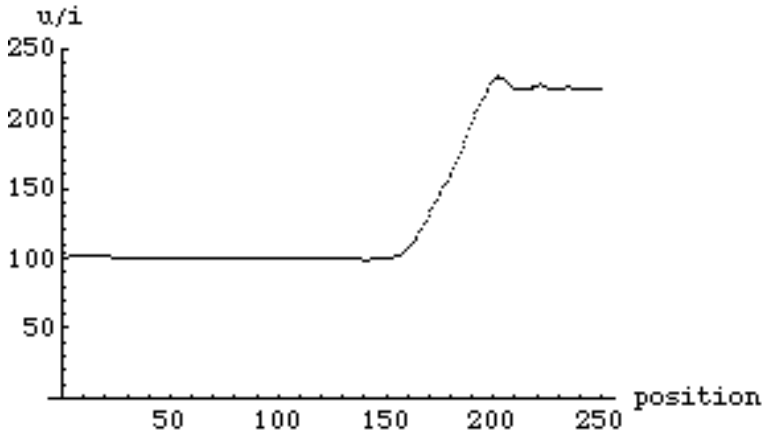
Next shown is the shape of the pulse after having travelled on line 1 for about 12m (horizontal unit is 4mm, leading edge is to the right):



and below the corresponding crosstalk on line 2, it has grown quite a bit :



One can do many more interesting experiments with this notebook. Let's look as a last thing for now at u/i on line 1 as function of position along the wire. The following plot displays voltage/current after the first reflection of the pulse at the right end. Before reflection we have $u_0 / i_0 = Z_0 = \sqrt{0.6/65}$ *1000 Ω = 96 Ω. The line is terminated with Z_x = 220 Ω. After reflection there is a superposition of the original pulse with the returning component: u = u_0 *(1+ (Z_x -Z_0)/ (Z_x +Z_0)), i = i_0 *(1- (Z_x -Z_0)/ (Z_x +Z_0)), so that u/i = Z_x .



- Graphics -

Conclusion

Tools have been presented that allow the detailed analysis of crosstalk on coupled linear conductors within the time domain. They also let you determine the capacities and inductances of linear conductors of arbitrary shape. All you have to do is to adapt the contour and the height of the potential boundaries and run the code example. The resulting values may be entered into the corresponding matrices of the telegraph equations to simulate and watch pulse propagation.

A simple formula was derived that lets you calculate the *exact potential of a line charge in a rectangular box* and hence the capacity of a wire inside a rectangular shield.

Crosstalk is a dynamic phenomenon. Although the coupling – capacitive and inductive – of the pigtail is surprisingly small, the amplitude of the crosstalk pulse grows considerably until it gets to an asymptotic value: In this case it reaches 2-3% after 1 m length and attains its equilibrium amplitude of well over 18% (even 30% if related to the damped pulse on line 1) after running about 13 m or 80 necs.

The ratio voltage/current is determined first by the impedance Z_0 , after the pulse has suffered reflection at the end of the conductor, the ratio changes to Z_x , the value of the terminating resistor.

References

(1) Reisch: Elektronische Bauelemente, Springer, 1998
(2) Unger: Elektromagnetische Wellen auf Leitungen, Hüthig, 4. Auflage, 1996
(3) Press et. al.: Numerical Recipes in Fortran, Cambridge University Press, Second Edition, 1992
(4) Ganzha,Vorozhtsov: Numerical Solutions for Partial Differential Equations, CRC Press, 1996
(5) Simonyi: Theoretische Elektrotechnik, Barth, 10. Auflage, 1993
(6) Jackson: Classical Electrodynamics, Wiley & Sons, Third Edition, 1998
(7) Küpfmüller, Kohn: Theoretische Elektrotechnik und Elektronik, Springer, 15. Auflage, 2000
(8) Wolfram: The *Mathematica* Book, Wolfram Media, Inc., Fourth Edition, 1999
(9) Wickham-Jones: *Mathematica* Graphics, Springer, 1994
(10) with help from D. Zagier, MPI für Mathematik, Bonn
(11) Johnson,Graham: High Speed Digital Design, Prentice Hall PTR, 1993