3

Numerical Inverse Laplace Transforms for Electrical Engineering Simulation

Lubomír Brančík
Brno University of Technology
Czech Republic

1. Introduction

Numerical inverse Laplace transform (NILT) methods are widely used in various scientific areas, especially for a solution of respective differential equations. In field of an electrical engineering many various approaches have been considered so far, but mostly for a single variable (1D NILT), see at least (Brančík, 1999, 2007b; Cohen, 2007; Valsa & Brančík, 1998; Wu at al., 2001) from plenty of papers. Much less attention was paid to multidimensional variable (nD NILT) methods, see e.g. (Hwang at al., 1983; Singhal at al., 1975), useful rather for more complicated electromagnetic systems. The 2D NILT methods, see e.g. (Brančík, 2005, 2007a, 2007b; Hwang & Lu, 1999), can be applied for a transmission line analysis, or nD NILT methods, $n \geq 2$, for a nonlinear circuits analysis, if relevant Laplace transforms are developed through a Volterra series expansion, see e.g. (Brančík, 2010a, 2010b, Karmakar, 1980; Schetzen, 2006), to highlight at least a few applications. This paper is focused on the class of NILT methods based on complex Fourier series approximation, their error analysis, their effective algorithms development in a Matlab language, and after all, on their selected applications in field of electrical engineering to show practical usefulness of the algorithms.

2. Multidimensional numerical inverse Laplace transform

An n-dimensional Laplace transform of a real function $f(\boldsymbol{t})$, with $\boldsymbol{t} = (t_1,...,t_n)$ as a row vector of n real variables, is defined as (Hwang at al., 1983)

$$F(\boldsymbol{s}) = \int_0^\infty \cdots \int_{n\text{-}fold} \int_0^\infty f(\boldsymbol{t}) \exp(-\boldsymbol{s}\boldsymbol{t}^T) \prod_{i=1}^{n} dt_i \,, \tag{1}$$

where $\boldsymbol{s} = (s_1,...,s_n)$ and T means a transposition. Under an assumption $|f(\boldsymbol{t})| < M\exp(\boldsymbol{\alpha}\boldsymbol{t}^T)$, with M real positive and $\boldsymbol{\alpha} = (\alpha_1,...,\alpha_n)$ being a minimal abscissa of convergence, and the nD Laplace transform $F(\boldsymbol{s})$ defined on a region $\{\boldsymbol{s} \in C^n : \mathrm{Re}[\boldsymbol{s}] > \boldsymbol{\alpha}\}$, with $\mathbf{c} = (c_1,...,c_n)$ as an abscissa of convergence, and the inequality taken componentwise, the original function is given by an n-fold Bromwich integral

$$f(\boldsymbol{t}) = \frac{1}{(2\pi j)^n} \int_{c_1 - j\infty}^{c_1 + j\infty} \cdots \int_{c_n - j\infty}^{c_n + j\infty} F(\boldsymbol{s}) \exp(\boldsymbol{s}\boldsymbol{t}^T) \prod_{i=1}^{n} dt_i \,. \tag{2}$$

In the papers (Brančík, 2007a, 2007b, 2010b), it was shown for the 1D, 2D, and 3D cases, the rectangular method of a numerical integration leads to an approximate formula whose a relative error is adjustable, and corresponds to the complex Fourier series approximation of a respective dimension. The method has been generalized for an arbitrary dimension n in the recent work (Brančík, 2011).

2.1 Complex Fourier series approximation and limiting relative error

Substituting $s_i = c_i + j\omega_i$ into (2), and using a rectangular rule of the integration, namely $\omega_i = m_i\Omega_i$, and $\Omega_i = 2\pi/\tau_i$ as generalized frequency steps, with τ_i forming a region of the solution $\boldsymbol{t} \in [0,\tau_1) \times ... \times [0,\tau_n)$, an approximate formula is

$$\tilde{f}(\boldsymbol{t}) = \exp(\boldsymbol{c}\boldsymbol{t}^T)\left(\prod_{i=1}^{n}\tau_i^{-1}\right)\sum_{m_1=-\infty}^{\infty}\cdots\sum_{m_n=-\infty}^{\infty}F(\boldsymbol{s})\exp\left(j\sum_{i=1}^{n}m_i\Omega_i t_i\right), \tag{3}$$

with $s_i = c_i + jm_i\Omega_i,\ \forall i$. As is shown in (Brančík, 2011), a limiting relative error δ_M of (3) can be controlled by setting $\mathbf{c} = (c_1,...,c_n)$, defining paths of the integration in (2), namely

$$c_i = \alpha_i - \frac{1}{\tau_i}\ln\left(1 - \frac{1}{\sqrt[n]{1+\delta_M}}\right) \approx \alpha_i - \frac{1}{\tau_i}\ln\frac{\delta_M}{n}, \tag{4}$$

for $i = 1,\ldots,n$, and while keeping the equalities $\tau_1(c_1 - \alpha_1) = ... = \tau_n(c_n - \alpha_n)$. The simplification in (4) is enabled due to small values δ_M considered in practice. The last equation is used for setting up parameters of the nD NILT method relating them to a limiting relative error δ_M required for practical computations.

2.2 Practical computational methods

It should be highlighted that the formula (4) is valid, and a relative error supposed is really achievable by the nD NILT (3), if infinite numbers of terms are used in the series. In practice, it cannot sure be fulfilled, but a suitable technique for accelerating a convergence of infinite series is usable, as is e.g. a quotient-difference (q-d) algorithm (Rutishauser, 1957). Besides, as has been already successfully used for cases of $n \le 3$, the formula (3) can be rearranged to enable using FFT & IFFT algorithms for an effective computation.

2.2.1 Partial ILTs evaluation technique

The technique of practical evaluation of the n-fold infinite sum (3) follows the properties of the n-fold Bromwich integral (2), namely we can rearrange it into the form

$$f(t_1,t_2,\ldots,t_n) = \frac{1}{2\pi j}\int_{c_1-j\infty}^{c_1+j\infty}\left(\frac{1}{2\pi j}\int_{c_2-j\infty}^{c_2+j\infty}\left(\cdots\frac{1}{2\pi j}\int_{c_n-j\infty}^{c_n+j\infty}F(s_1,s_2,\ldots,s_n)e^{s_n t_n}ds_n\cdots\right)e^{s_2 t_2}ds_2\right)e^{s_1 t_1}ds_1, \tag{5}$$

or shortly

$$f(t_1,t_2,\ldots,t_n) = \mathbb{L}_1^{-1}\left[\mathbb{L}_2^{-1}\left[\cdots\mathbb{L}_n^{-1}\left[F(s_1,s_2,\ldots,s_n)\right]\cdots\right]\right]. \tag{6}$$

Although the order of the integration may be arbitrary on principle, here the above one will be used for an explanation. Similarly, (3) can be rewritten as

$$\tilde{f}(t_1,t_2,\ldots,t_n)=\frac{e^{c_1t_1}}{\tau_1}\sum_{m_1=-\infty}^{\infty}\left(\frac{e^{c_2t_2}}{\tau_2}\sum_{m_2=-\infty}^{\infty}\left(\cdots\frac{e^{c_nt_n}}{\tau_n}\sum_{m_n=-\infty}^{\infty}F(s_1,s_2,\ldots,s_n)e^{jm_n\Omega_nt_n}\cdots\right)e^{jm_2\Omega_2t_2}\right)e^{jm_1\Omega_1t_1}\ , \quad (7)$$

with $s_i = c_i + jm_i\Omega_i$. If we define $F_n \equiv F(s_1,...,s_{n-1},s_n)$ and $F_0 \equiv f(t_1,...,t_{n-1},t_n)$, then n consequential partial inversions are performed as

$$\begin{aligned}\mathbb{L}_n^{-1}\{F_n\}&=F_{n-1}(s_1,\ldots,s_{n-1},t_n)\,,\\ \mathbb{L}_{n-1}^{-1}\{F_{n-1}\}&=F_{n-2}(s_1,\ldots,t_{n-1},t_n)\,,\\ &\vdots\\ \mathbb{L}_1^{-1}\{F_1\}&=f(t_1,\ldots,t_{n-1},t_n)\,.\end{aligned} \quad (8)$$

As is obvious we need to use a procedure able to make the inversion of Laplace transforms dependent on another n-1 parameters, complex in general. Let us denote arguments in (8) by $\boldsymbol{p}_i = (p_1,...,p_{n-1},p_n)$. Then the ILT of the type

$$F_{i-1}(\boldsymbol{p}_{i-1})=\mathbb{L}_i^{-1}\{F_i(\boldsymbol{p}_i)\}=\frac{1}{2\pi j}\int_{c_i-j\infty}^{c_i+j\infty}F_i(\boldsymbol{p}_i)e^{s_it_i}ds_i \quad (9)$$

can be used n times, $i = n,n$-1...,1, to evaluate (8), with $p_n = (s_1,...,s_{n-1},s_n)$, $p_{n-1} = (s_1,...,s_{n-1},t_n)$,..., $p_1 = (s_1,...,t_{n-1},t_n)$, and $p_0 = (t_1,...,t_{n-1},t_n)$, while $p_j = s_j$ for $j \le i$, and $p_j = t_j$ otherwise. A further technique is based on demand to find the solution on a whole region of discrete points. Then, taking into account $t_{ik} = kT_i$ in (9), with T_i as the sampling periods in the original domain, we can write an approximate formula

$$\tilde{F}_{i-1}(\boldsymbol{p}_{i-1})=\frac{e^{c_ikT_i}}{\tau_i}\sum_{m=-\infty}^{\infty}\tilde{F}_i(\boldsymbol{p}_i)e^{j2\pi mkT_i/\tau_i}\ , \quad (10)$$

$i = n,n$-1...,1, and with $\Omega_i = 2\pi/\tau_i$ substituted. As follows from the error analysis (Brančík, 2011) a relative error is predictable on the region $O_{err} = [0,\tau_1) \times ... \times [0,\tau_n)$. For $k = 0,1,...,M_i$-1, $i = 1,...,n$, a maximum reachable region is $O_{max} = [0,(M_1\text{-}1)T_1] \times ... \times [0,(M_n\text{-}1)T_n]$. Thus, to meet the necessary condition $O_{max} \subset O_{err}$, we can set up fittingly $\tau_i = M_iT_i$, $i = 1,...,n$. In practice, a region of the calculation is chosen to be $O_{cal} = [0,t_{1cal}] \times ... \times [0,t_{ncal}]$, with $t_{ical} = (M_i/2\text{–}1)T_i$, $i = 1,...,n$, to provide certain margins.

2.2.2 FFT, IFFT, and quotient-difference algorithms utilization

As is shown in (Brančík, 2007a, 2010c), the discretized formula (10) can be evaluated by the FFT and IFFT algorithms, in conjunction with the quotient-difference (q-d) algorithm for accelerating convergence of the residual infinite series, see following procedures.

To explain it in more detail, let us consider an r-th cycle in gaining the original function via (9), i.e. $F_{r-1}(\boldsymbol{p}_{r-1})=\mathbb{L}_r^{-1}\{F_r(\boldsymbol{p}_r)\}$. For its discretized version (10) we have

$$\tilde{F}_{r-1}(s_1,\ldots,kT_r,\ldots,t_n)=\frac{e^{c_rkT_r}}{\tau_r}\sum_{m=-\infty}^{\infty}\tilde{F}_r(s_1,\ldots,c_r+jm\frac{2\pi}{\tau_r},\ldots,t_n)e^{j2\pi mkT_r/\tau_r}\ . \quad (11)$$

The above stated formula can be decomposed and expressed also as

$$\tilde{F}_{r-1}(s_1,\ldots,kT_r,\ldots,t_n)=\frac{e^{c_r kT_r}}{\tau_r}\left[\sum_{m=0}^{M_r-1}\tilde{F}_r^{(-m)}z_{-k}^m+\sum_{m=0}^{\infty}\tilde{G}_r^{(-m)}z_{-k}^m+\sum_{m=0}^{M_r-1}\tilde{F}_r^{(m)}z_k^m+\sum_{m=0}^{\infty}\tilde{G}_r^{(m)}z_k^m-\tilde{F}_r^{(0)}\right], \quad (12)$$

where individual terms are defined as

$$\begin{aligned} M_r &= 2^{K_r},\ K_r \text{ integer},\\ \tilde{F}_r^{(\pm m)} &= \tilde{F}_r(s_1,\ldots,c_r \pm jm2\pi/\tau_r,\ldots,t_n),\\ \tilde{G}_r^{(\pm m)} &= \tilde{F}_r^{(\pm M_r \pm m)},\\ z_{\pm k} &= \exp(\pm j2\pi kT_r/\tau_r), \end{aligned} \quad (13)$$

when $z_{\pm k}^{M_r}=e^{\pm j2\pi k}=1,\ \forall k$, has been considered, and $\tau_r = M_r T_r$.
As is evident the first and the third finite sum of (12) can be evaluated via the FFT and IFFT algorithms, respectively, while $2P$+1 terms from the infinite sums are used as the input data in the quotient-difference algorithm (Macdonald, 1964; McCabe, 1983; Rutishauser, 1957). We can replace the above infinite power series by a continued fraction as

$$\sum_{m=0}^{\infty}\tilde{G}_r^{(\pm m)}z_{\pm k}^m \approx d_0/(1+d_1z_{\pm k}/(1+\cdots+d_{2P}z_{\pm k})),\ \forall k, \quad (14)$$

which gives much more accurate result than the original sum truncated on $2P$+1 terms only. The q-d algorithm process can be explained based on a lozenge diagram shown in Fig. 1.

$$\begin{array}{ccccccc} e_0^{(0)} & & & & & & \\ & q_1^{(0)} & & & & & \\ e_0^{(1)} & & e_1^{(0)} & & & & \\ & q_1^{(1)} & & q_2^{(0)} & & & \\ e_0^{(2)} & & e_1^{(1)} & & e_2^{(0)} & & \\ & q_1^{(2)} & & q_2^{(1)} & & & \\ e_0^{(3)} & & e_1^{(2)} & & & & \\ & q_1^{(3)} & & & & & \\ e_0^{(4)} & & & & & & \end{array}$$

Fig. 1. Quotient-difference algorithm lozenge diagram

The first two columns are formed as

$$\begin{aligned} e_0^{(i)} &= 0, \qquad i=0,\ldots,2P,\\ q_1^{(i)} &= \tilde{G}_r^{\pm(i+1)}/\tilde{G}_r^{\pm i}, \quad i=0,\ldots,2P-1, \end{aligned} \quad (15)$$

while the successive columns are given by the rules

$$\begin{aligned} e_j^{(i)} &= q_j^{(i+1)}-q_j^{(i)}+e_{j-1}^{(i+1)}, \quad i=0,\cdots,2P-2j, \quad \text{for} \quad j=1,\ldots,P,\\ q_j^{(i)} &= q_{j-1}^{(i+1)}e_{j-1}^{(i+1)}/e_{j-1}^{(i)}, \quad i=0,\cdots,2P-2j-1, \quad \text{for} \quad j=2,\ldots,P. \end{aligned} \quad (16)$$

Then, the coefficients d_m, $m = 0,...,2P$, in (14) are given by

$$d_0 = \tilde{G}_r^{(0)} \ , \ d_{2j-1} = -q_j^{(0)} \ , \ d_{2j} = -e_j^{(0)} \ , \quad j = 1,\dots,P \ . \tag{17}$$

For practical computations, however, the recursive formulae stated below are more effective to be used (DeHoog et al., 1982). They are of the forms

$$A_m(z_{\pm k}) = A_{m-1}(z_{\pm k}) + d_m z_{\pm k} A_{m-2}(z_{\pm k}) \ , \ B_m(z_{\pm k}) = B_{m-1}(z_{\pm k}) + d_m z_{\pm k} B_{m-2}(z_{\pm k}) \ , \tag{18}$$

for $m = 1,...,2P$, $\forall k$, with the initial values $A_{-1} = 0$, $B_{-1} = 1$, $A_0 = d_0$, and $B_0 = 1$. Then, instead of the continued fraction (14), we can write

$$\sum_{m=0}^{\infty} \tilde{G}_r^{(\pm m)} z_{\pm k}^m \approx A_{2P}(z_{\pm k}) / B_{2P}(z_{\pm k}) \ , \ \forall k \ . \tag{19}$$

The q-d algorithm is a very efficient tool just for a power series convergence acceleration, here enabling (7) to achieve a relative error near its theoretical value defined by (4), see the following examples.

2.3 Matlab listings and experimental errors evaluation

In this part experimental verifications of the nD NILT theory above will first be presented, for one to three dimensional cases, i.e. $n \le 3$. For such dimensions the Matlab functions have been developed and errors stated on a basis of some sample images with known originals. The Matlab listings of basic versions of the NILT functions are provided, together with examples of their right calling. Another Matlab listings will be discussed in more detail later, in the chapter with practical applications.

2.3.1 One-dimensional NILT

In case of the 1D inverse LT, a well-known Bromwich integral results from (2), namely

$$f(t) = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s) e^{st} dt \ , \tag{20}$$

where indexes 1 were omitted. By using the theory above a path of the numerical integration is stated according to (4), leading to

$$c = \alpha - \frac{1}{\tau} \ln\left(1 - \frac{1}{1+\delta_M}\right) = \alpha + \frac{1}{\tau} \ln\left(1 + \frac{1}{\delta_M}\right) \approx \alpha - \frac{1}{\tau} \ln \delta_M \ . \tag{21}$$

In contrast to most other approaches, the 1D NILT method described here enables to treat complex images resulting in complex originals as no real or imaginary parts are extracted during an evaluation process. It can be useful in some special applications, not only in the electrical engineering. We can śhow it on a simple transform pair

$$F(s) = \frac{1}{s - j\omega} = \frac{s}{s^2 + \omega^2} + j \frac{\omega}{s^2 + \omega^2} \quad \mapsto \quad f(t) = e^{j\omega t} = \cos \omega t + j \sin \omega t \ . \tag{22}$$

Of course, when preprocessing the transform to arrange it to a Cartesian form, as is shown on the right sides in (22), the result could by get by inverting the real and imaginary parts separately, by using an arbitrary NILT method. Here, however, no symbolic manipulations are needed in advance, and $F(s)$ enters the NILT function in its basic form as a whole.
A Matlab language listing is shown in Tab. 1, where the relative error needed is marked by `Er` and is subject to a change if necessary, similarly as the minimal abscissa of convergence (exponential order) α, `alfa`, numbers of points for the resultant solution, `M`, and for the q-d algorithm, `P`. If only real transforms $F(s)$ are considered the bottom line in the listing can be inactivated. The NILT function is called from a command line as follows: `niltc('F',tm);` where `F` is a name of another function in which the $F(s)$ is defined, and `tm` marks an upper limit of the original variable t. In our case, and for $\omega = 2\pi$, this function can have a form

```
function f=expc(s)
f=1./(s-2*pi*j);
```

```
% ------ 1D NILT for complex arguments - basic version -------
% ----------- based on FFT/IFFT/q-d, by L. Brančík -----------
function [ft,t]=niltc(F,tm);
alfa=0; M=256; P=3; Er=1e-10;                        % adjustable
N=2*M; qd=2*P+1;
t=linspace(0,tm,M); NT=2*tm*N/(N-2); omega=2*pi/NT;
c=alfa+log(1+1/Er)/NT; s=c-i*omega*(0:N+qd-1);
Fs(1,:)=feval(F,s); Fs(2,:)=feval(F,conj(s));
ft(1,:)=fft(Fs(1,1:N)); ft(2,:)=N*ifft(Fs(2,1:N));
ft=ft(:,1:M); D=zeros(2,qd); E=D;
Q=Fs(:,N+2:N+qd)./Fs(:,N+1:N+qd-1);
D(:,1)=Fs(:,N+1); D(:,2)=-Q(:,1);
for r=2:2:qd-1
    w=qd-r;
    E(:,1:w)=Q(:,2:w+1)-Q(:,1:w)+E(:,2:w+1); D(:,r+1)=-E(:,1);
    if r>2
        Q(:,1:w-1)=Q(:,2:w).*E(:,2:w)./E(:,1:w-1);
        D(:,r)=-Q(:,1);
    end
end
A2=zeros(2,M); B2=ones(2,M); A1=repmat(D(:,1),[1,M]); B1=B2;
z1=exp(-i*omega*t); z=[z1;conj(z1)];
for n=2:qd
    Dn=repmat(D(:,n),[1,M]);
    A=A1+Dn.*z.*A2; B=B1+Dn.*z.*B2; A2=A1; B2=B1; A1=A; B1=B;
end
ft=ft+A./B; ft=sum(ft)-Fs(2,1); ft=exp(c*t)/NT.*ft;
ft(1)=2*ft(1);
figure; plot(t,real(ft));
figure; plot(t,imag(ft));                             % optional
```

Table 1. Matlab listing of 1D NILT method accepting complex arguments

As is obvious, the Laplace transform must be defined to enable Matlab array processing, i.e. element-by-element array operators have to be used. Thus, the calling our function can look like `niltc('expc',4);` if the function is saved under the same name, `expc`, or it is placed inside the M-file with own NILT function (Tab. 1), following always its body. Alternatively, the calling can look like `[ft,t]=niltc('F',tm);` if respective variables in the brackets are to be saved in the memory after the function finishes.

Graphical results and corresponding errors are shown in Fig. 2. Because the originals are bounded by values ±1, and $\alpha = 0$, we can see the errors satisfy (21) very well ($\delta_M = 10^{-10}$ was considered), excluding only beginning of the interval.

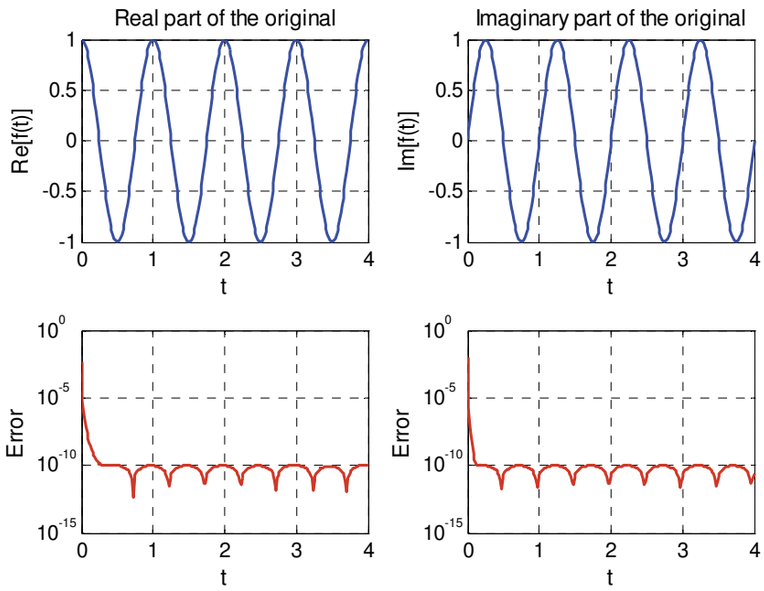

Fig. 2. Numerical inversion leading to complex original $f(t) = \exp(j\omega t)$

Another test functions are considered in Tab. 2, with numerical results shown in Fig. 3. As is again obvious from Fig. 3 the relative errors satisfy theoretical expectations, with an exception of vicinities of discontinuities.

i	1	2	3	4	5	6
$F_i(s)$	$\frac{1}{s+1}$	$\frac{1}{(s+1)^2}$	$\frac{2\pi}{s^2+4\pi^2}$	$\frac{1}{\sqrt{s^2+1}}$	$\frac{e^{-\sqrt{s}}}{s}$	$\frac{e^{-s}}{s}$
$f_i(t)$	e^{-t}	te^{-t}	$\sin(2\pi t)$	$J_0(t)$	$\operatorname{erfc}\left(\frac{1}{2\sqrt{t}}\right)$	$\underline{1}(t-1)$

Table 2. Test Laplace transforms for errors evaluation

2.3.2 Two-dimensional NILT

In case of the 2D inverse LT, a two-fold Bromwich integral results from (2), namely

$$f(t_1,t_2) = -\frac{1}{4\pi^2}\int_{c_1-j\infty}^{c_1+j\infty}\int_{c_2-j\infty}^{c_2+j\infty} F(s_1,s_2)e^{s_1t_1+s_2t_2}\,ds_1 ds_2\,, \tag{23}$$

and by using the theory above the paths of numerical integrations are stated based on (4) as

$$c_i = \alpha_i - \frac{1}{\tau_i}\ln\left(1-\frac{1}{\sqrt{1+\delta_M}}\right) \approx \alpha_i - \frac{1}{\tau_i}\ln\frac{\delta_M}{2}, \quad i = 1,2\,. \tag{24}$$

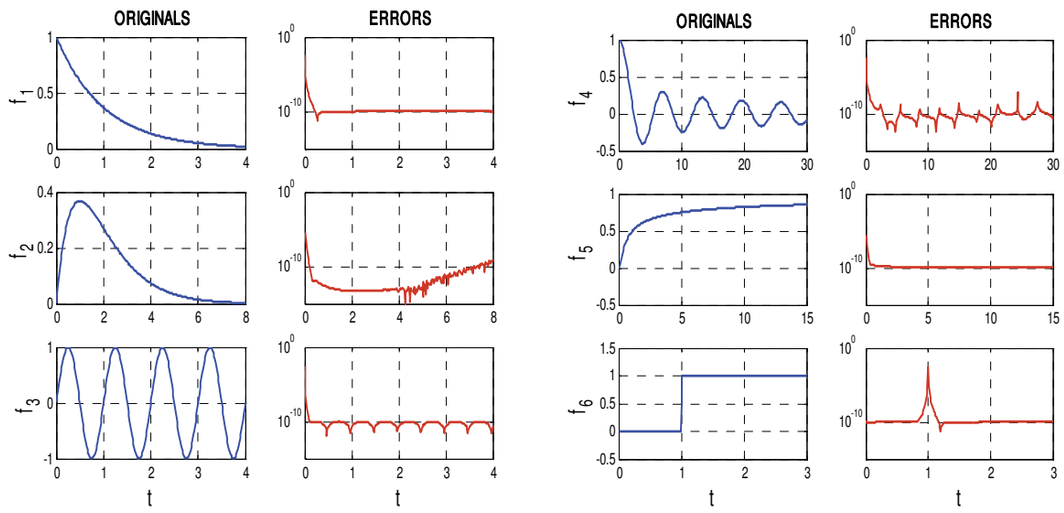

Fig. 3. Computed originals and errors for test Laplace transforms in Tab. 2

A Matlab language listing is shown in Tab. 3, with all the parameters denoted by similar way as in the previous case.
Nevertheless, the Laplace transform variables and the original variables have changed in this listing as $s_1 \to p$, $s_2 \to q$, and $t_1 \to x$, $t_2 \to y$, respectively, which simplified a writing. The parameters are then indexed in compliance with these new notations. Besides, numbers of points used to plot three-dimensional graphical results are set by `xpl` and `ypl`.
For the same reasons as at the 1D NILT, the 2D NILT method discussed here enables to treat complex images of two variables resulting in complex originals. We will śhow it on a simple transform pair

$$F(s_1,s_2)=\frac{1}{(s_1-j\omega_1)(s_2-j\omega_2)} \quad \mapsto \quad f(t_1,t_2)=e^{j(\omega_1 t_1+\omega_2 t_2)}. \tag{25}$$

After rearranging the above equation, we can also write

$$F(s_1,s_2)=\frac{s_1s_2-\omega_1\omega_2}{\left(s_1^2+\omega_1^2\right)\left(s_2^2+\omega_2^2\right)}+j\frac{\omega_2 s_1+\omega_1 s_2}{\left(s_1^2+\omega_1^2\right)\left(s_2^2+\omega_2^2\right)} \quad \mapsto \quad f(t_1,t_2)=\cos(\omega_1 t_1+\omega_2 t_2)+j\sin(\omega_1 t_1+\omega_2 t_2). \tag{26}$$

The 2D NILT function is called from a command line as follows: `nilt2c('F',xm,ym);` where `F` is a name of another function in which the $F(p,q)$ is defined, and `xm` and `ym` mark upper limits of the original variables x and y. In our case, and for $\omega_1 = \omega_2 = 2\pi$, this function can have a form

```
function f=exp2c(p,q)
f=1./(p-2*pi*j)./(q-2*pi*j);
```

and its calling can look like `nilt2c('exp2c',3,3);` with graphical results in Fig. 4. As the originals are bounded by values ±1, and $\alpha_1 = \alpha_2 = 0$, we can see the errors satisfy (21) very well ($\delta_M = 10^{-8}$ was considered), excluding beginnings of the 2D region.

```
% ----- 2D NILT based on partial inversions, by L. Brančík -----
function fxy=nilt2c(F,xm,ym);
alfax=0; alfay=0; Mx=256; My=256; P=3; Er=1e-8;     % adjustable
xpl=64; ypl=64;                                     % adjustable
Nx=2*Mx; Ny=2*My; qd=2*P+1; Ke=log(1-1/sqrt(1+Er));
nx=2*xm*Nx/(Nx-2); ny=2*ym*Ny/(Ny-2);
omegax=2*pi/nx; omegay=2*pi/ny; sigx=alfax-Ke/nx;
sigy=alfay-Ke/ny; qd1=qd-1; Nxw=Nx+qd1; Nyw=Ny+qd1;
Asigx=sigx-i*omegax*(0:Nxw); Asigy=sigy-i*omegay*(0:Nyw);
Asigx2=cat(2,Asigx,conj(Asigx));
rx=[1:Mx/xpl:Mx,Mx]; ry=[1:My/ypl:My,My];
x=linspace(0,xm,Mx); y=linspace(0,ym,My); x=x(rx); y=y(ry);
[q,p]=meshgrid(Asigy,Asigx2); Fpq(:,:,1)=feval(F,p,q);
[q,p]=meshgrid(conj(Asigy),Asigx2); Fpq(:,:,2)=feval(F,p,q);
Fpyp=Pnilt(Fpq,Ny,ry,qd,y,ny,omegay,sigy); % Pnilt to get F(p,y)
Fpy(:,:,1)=Fpyp(1:Nxw+1,:).';
Fpy(:,:,2)=Fpyp(Nxw+2:2*Nxw+2,:).';
fxy=Pnilt(Fpy,Nx,rx,qd,x,nx,omegax,sigx);  % Pnilt to get f(x,y)
figure; mesh(x,y,real(fxy));
figure; mesh(x,y,imag(fxy));                          % optional
% ------ PARTIAL NILT based on FFT/IFFT/Q-D, by L.Brančík ------
function fx=Pnilt(Fq,N,grid,qd,xy,nxy,omega,c);
fx(:,:,1)=fft(Fq(:,:,1),N,2); fx(:,:,2)=N*ifft(Fq(:,:,2),N,2);
fx=fx(:,grid,:); delv=size(Fq,1); delxy=length(xy);
d=zeros(delv,qd,2); e=d; q=Fq(:,N+2:N+qd,:)./Fq(:,N+1:N+qd-1,:);
d(:,1,:)=Fq(:,N+1,:); d(:,2,:)=-q(:,1,:);
for r=2:2:qd-1
    w=qd-r; e(:,1:w,:)=q(:,2:w+1,:)-q(:,1:w,:)+e(:,2:w+1,:);
    d(:,r+1,:)=-e(:,1,:);
    if r>2
        q(:,1:w-1,:)=q(:,2:w,:).*e(:,2:w,:)./e(:,1:w-1,:);
        d(:,r,:)=-q(:,1,:);
    end
end
A2=zeros(delv,delxy,2); B2=ones(delv,delxy,2);
A1=repmat(d(:,1,:),[1,delxy,1]); B1=B2;
z1(1,:,1)=exp(-i*omega*xy); z1(1,:,2)=conj(z1(1,:,1));
z=repmat(z1,[delv,1]);
for n=2:qd
    Dn=repmat(d(:,n,:),[1,delxy,1]);
    A=A1+Dn.*z.*A2; B=B1+Dn.*z.*B2; A2=A1; B2=B1; A1=A; B1=B;
end
fx=fx+A./B; fx=sum(fx,3)-repmat(Fq(:,1),[1,delxy]);
fx=repmat(exp(c*xy)/nxy,[delv,1]).*fx; fx(:,1)=2*fx(:,1);
```

Table 3. Matlab listing of 2D NILT based on partial inversions

Another simple example shows a shifted 2D unit step, with different shifts along the axis. A corresponding transform pair is

$$F(s_1,s_2)=\frac{\exp(-2s_1-s_2)}{s_1 s_2} \quad \mapsto \quad f(t_1,t_2)=\underline{1}(t_1-2,t_2-1)\,. \tag{27}$$

In this case, a displaying imaginary part gives a zero plane, and the respective line in the 2D NILT function can be inactivated. The graphical results are depicted in Fig. 5, including an absolute error. The respective Matlab function can be of a form

```
function f=step2(p,q)
f=exp(-2*p-q)./p./q;
```

and called as `nilt2c('step2',4,4);`, with the results theoretically expected.

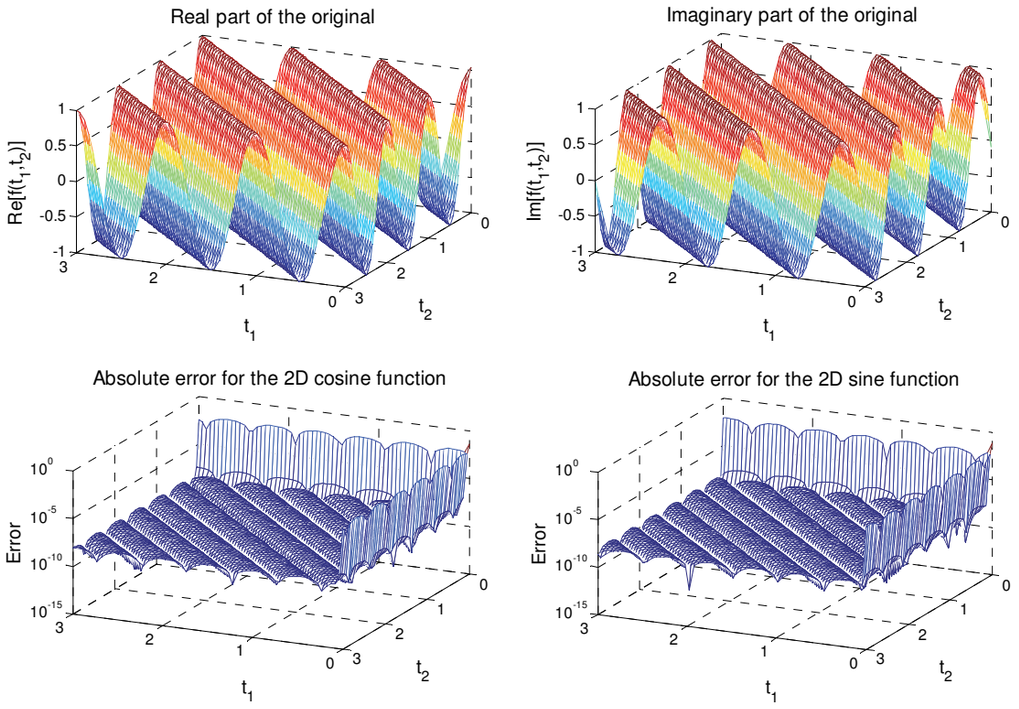

Fig. 4. Numerical inversion leading to complex original $f(t_1,t_2) = \exp(j\omega(t_1+t_2))$

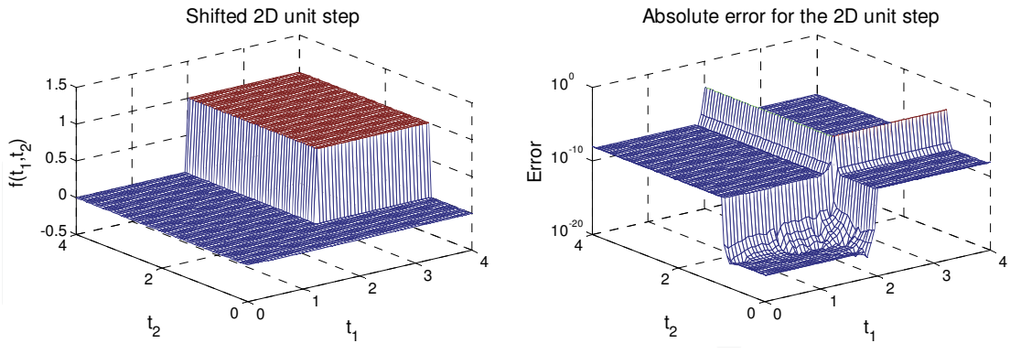

Fig. 5. Numerical inversion leading to shifted 2D unit step $f(t_1,t_2) = \underline{1}(t_1-2,t_2-1)$

2.3.3 Three-dimensional NILT

In case of the 3D inverse LT, a three-fold Bromwich integral results from (2), namely

$$f(t_1,t_2,t_3) = \frac{j}{8\pi^3} \int_{c_1-j\infty}^{c_1+j\infty} \int_{c_2-j\infty}^{c_2+j\infty} \int_{c_3-j\infty}^{c_3+j\infty} F(s_1,s_2,s_3) e^{s_1t_1+s_2t_2+s_3t_3} ds_1 ds_2 ds_3 \,, \tag{28}$$

and by using the theory above the paths of numerical integrations are stated based on (4) as

$$c_i = \alpha_i - \frac{1}{\tau_i}\ln\left(1 - \frac{1}{\sqrt[3]{1+\delta_M}}\right) \approx \alpha_i - \frac{1}{\tau_i}\ln\frac{\delta_M}{3}, \quad i = 1,2,3\,. \tag{29}$$

Here only experimental results will be shown to verify an accuracy of the method. A Matlab language listing looks similarly like for the 2D NILT case, but the partial NILT subfunction is called once more, and respective arrays dimensions are enlarged. Original functions corresponding to 3D Laplace transforms cannot be displayed graphically as a whole, of course. However, for one variable chosen as constant, it is posssible to display three respective two-dimensional cuts. It will be demonstrated on the example of 3D shifted unit step, with a Laplace transform pair

$$\frac{\exp(-s_1 - 2s_2 - 3s_3)}{s_1 s_2 s_3} \quad \mapsto \quad \underline{1}(t_1 - 1, t_2 - 2, t_2 - 3)\,, \tag{30}$$

with different values of shifts along respective coordinates so that correctness of results can easily be identified, see Fig. 6. Errors again correspond to theoretically expected ones.

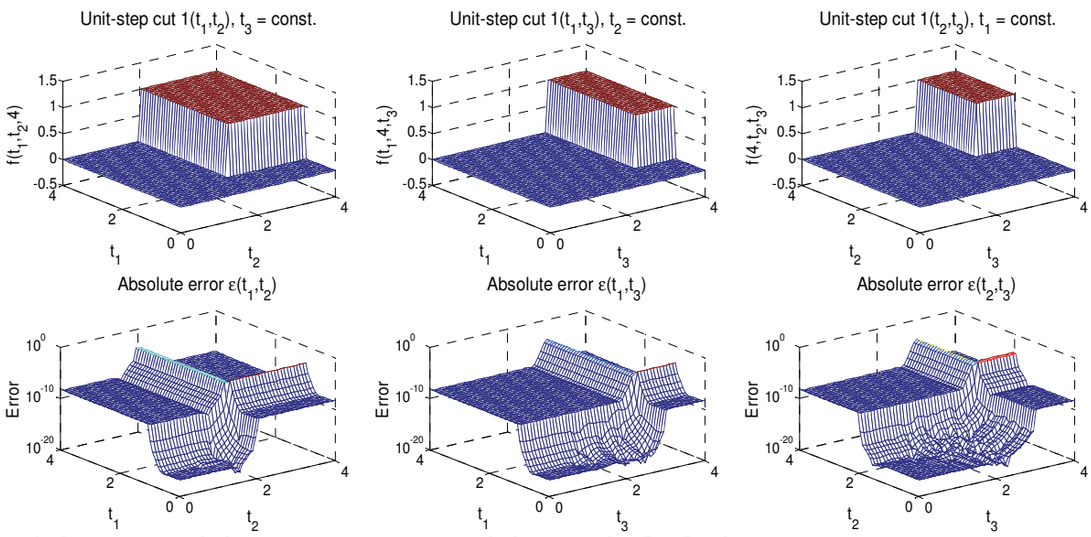

Fig. 6. Numerical inversion leading to shifted 3D unit step $f(t_1,t_2,t_3) = \underline{1}(t_1–1,t_2–2,t_3–3)$

3. Application of NILT algorithms to electrical engineering simulation

In this chapter some examples of the application of the NILT algorithms developed relating to problems of electrical engineering simulation are presented. First, the 1D NILT method is applied for the solution of transient phenomena in linear electrical circuits with both lumped and distributed parameters. This well-known approach is usable wherever linear ordinary differential equations (ODE) are transformed into algebraic ones so that an inverse Laplace transform can be considered. Then the 2D NILT method is utilized to solve transient phenomena on transmission lines (TL) after relevant telegraphic equations (a type of partial differential equations (PDE)) are transformed into algebraic ones by a 2D Laplace transform. In this way voltage and/or current distributions along the TL wires can be determined in a single calculation step. Finally, the utilization of the 1D to 3D NILTs to weakly nonlinear

electrical circuits solution is discussed. In this case the relevant nonlinear ODEs describing the circuit are expanded into Volterra series which respective NILTs are applied on.

3.1 One-dimensional NILT algorithm application

3.1.1 Preliminary example based on lumped parameter circuit

A simple example demonstrating the application of the basic 1D NILT algorithm in Tab. 1 is shown in Fig. 7. This really initiatory linear electrical circuit was chosen with an intention to be also considered later, in chapter 3.3.1, as a nonlinear circuit, with G_2 being a nonlinear element. In this way one will be able to compare results and make some conclusions.

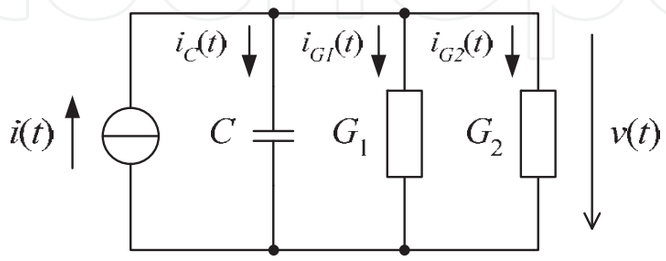

Fig. 7. Linear reactive electrical circuit of the 1st order

Denoting $G = G_1 + G_2$, the 1st-order linear ODE has a form

$$C\frac{dv(t)}{dt} + Gv(t) = i(t)\,, \tag{31}$$

with a Laplace-domain solution

$$V(s) = \frac{I(s) + Cv(0)}{G + sC}\,, \tag{32}$$

with an initial condition $v(0)$. Even if the above circuit is very simple a finding time-domain solution could be rather work-intensive if the circuit is excited from some non-trivial input current waveform. A few basic examples are given in Tab. 4, specially the first one results in a transient characteristic of the circuit.

k	1	2	3	4
$i_k(t)$	$I_0\underline{1}(t)$	$I_0 e^{-5t}\underline{1}(t)$	$I_0 \sin(2\pi t)\underline{1}(t)$	$I_0 \cos(2\pi t)\underline{1}(t)$
$I_k(s)$	$\frac{I_0}{s}$	$\frac{I_0}{s+5}$	$\frac{2\pi I_0}{s^2+4\pi^2}$	$\frac{sI_0}{s^2+4\pi^2}$

Table 4. Exciting current source waveforms and their Laplace transforms

For the above examples, of course, time-domain analytical solutions can be found e.g. based on a Heaviside formula. The 1D NILT function graphical results, under a condition $v(0) = 0$, and considering values C = 1mF, $G_1 = G_2$ = 10mS, and I_0 = 1mA, are shown in Fig. 8.
The above waveforms can be got by either successive application of a basic version of the 1D NILT method according to Tab. 1, or a generalized 1D NILT function, its vector version, can be used to process all the computations in parallel. This function is shown in Tab. 5.

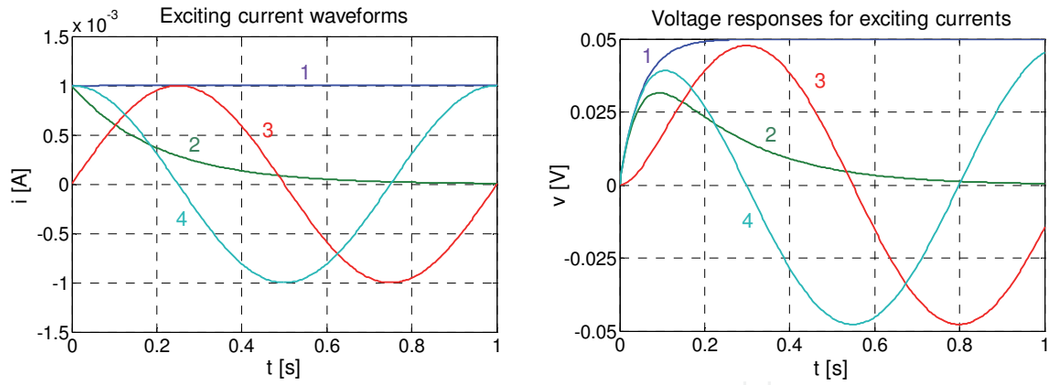

Fig. 8. Numerically computed exciting current and voltage responses waveforms

```
% ------ 1D NILT for complex arguments - vector version ------- %
% ----------- based on FFT/IFFT/q-d, by L. Brančík ------------ %
function [ft,t]=niltcv(F,tm,depict);
alfa=0; M=256; P=3; Er=1e-10;                          % adjustable
N=2*M; qd=2*P+1; t=linspace(0,tm,M); NT=2*tm*N/(N-2);
omega=2*pi/NT;
c=alfa+log(1+1/Er)/NT; s=c-i*omega*(0:N+qd-1);
Fs(:,:,1)=feval(F,s); Fs(:,:,2)=feval(F,conj(s)); lv=size(Fs,1);
ft(:,:,1)=fft(Fs(:,:,1),N,2); ft(:,:,2)=N*ifft(Fs(:,:,2),N,2);
ft=ft(:,1:M,:);
D=zeros(lv,qd,2); E=D; Q=Fs(:,N+2:N+qd,:)./Fs(:,N+1:N+qd-1,:);
D(:,1,:)=Fs(:,N+1,:); D(:,2,:)=-Q(:,1,:);
for r=2:2:qd-1
    w=qd-r;
    E(:,1:w,:)=Q(:,2:w+1,:)-Q(:,1:w,:)+E(:,2:w+1,:);
    D(:,r+1,:)=-E(:,1,:);
    if r>2
    Q(:,1:w-1,:)=Q(:,2:w,:).*E(:,2:w,:)./E(:,1:w-1,:);
    D(:,r,:)=-Q(:,1,:);
    end
end
A2=zeros(lv,M,2); B2=ones(lv,M,2); A1=repmat(D(:,1,:),[1,M,1]);
B1=B2; z1=repmat(exp(-i*omega*t),[lv,1]); z=cat(3,z1,conj(z1));
for n=2:qd
    Dn=repmat(D(:,n,:),[1,M,1]);
    A=A1+Dn.*z.*A2; B=B1+Dn.*z.*B2; A2=A1; B2=B1; A1=A; B1=B;
end
ft=ft+A./B; ft=sum(ft,3)-repmat(Fs(:,1,2),[1,M,1]);
ft=repmat(exp(c*t)/NT,[lv,1]).*ft; ft(:,1)=2*ft(:,1);
switch depict
    case 'p1', plott1(t,ft); case 'p2', plott2(t,ft);
    case 'p3', plott3(t,ft); otherwise display('Invalid Plot');
end
```

Table 5. Matlab listing of vector version of 1D NILT method

Here one more parameter `depict` is used to define a method of plotting individual items from a set of originals. The 1D NILT function is called as `niltcv('F',tm,'depict');` where `'depict'` is a text string `'p1'`, `'p2'`, or `'p3'`, see Tab. 6 for more details.

```
% --- Plotting functions called by 1D NILT, vector version ----
%---------- Multiple plotting into single figure ------------
function plott1(t,ft)
figure; plot(t,real(ft)); grid on;
figure; plot(t,imag(ft)); grid on;                    % optional
% ------------- Plotting into separate figures ----------------
function plott2(t,ft)
for k=1:size(ft,1)
    figure; plot(t,real(ft(k,:))); grid on;
    figure; plot(t,imag(ft(k,:))); grid on;          % optional
end
% ------------------ Plotting into 3D graphs ------------------
function plott3(t,ft)
global x;                                  % x must be global in F
    m=length(t); tgr=[1:m/64:m,m];         % 65 time points chosen
    figure; mesh(t(tgr),x,real(ft(:,tgr)));
    figure; mesh(t(tgr),x,imag(ft(:,tgr)));           % optional
```

Table 6. Matlab listing of plotting functions for vector version of 1D NILT method

To get e.g. the right part of Fig. 8, that is the voltage responses of the circuit in Fig. 7, the calling the 1D NILT function looks like `niltcv('V4',1,'p1');` where `V4` denotes a name of the function defining individual responses as follows:

```
function f=V4(s)
I0=1e-3; C=1e-3; G=2e-2;
f(1,:)=I0./s./(G+s*C);
f(2,:)=I0./(s+5)./(G+s*C);
f(3,:)=2*pi*I0./(s.^2+4*pi^2)./(G+s*C);
f(4,:)=s.*I0./(s.^2+4*pi^2)./(G+s*C);
```

In this case the lines causing the imaginary parts plotting can be inactivated. The remaining plotting functions will be explained in the next chapter.

3.1.2 Application for transmission line simulation

Here, the 1D NILT algorithms will be used to simulate voltage and/or current distributions along transmission lines (TL), as shown on a Laplace-domain TL model in Fig. 9. As is well known, this model results from the application of a Laplace transform, with respect to time, on a pair of partial differential equations (telegraphic) of the form

$$-\frac{\partial v(t,x)}{\partial x} = R_0 i(t,x) + L_0 \frac{\partial i(t,x)}{\partial t} \quad , \quad -\frac{\partial i(t,x)}{\partial x} = G_0 v(t,x) + C_0 \frac{\partial v(t,x)}{\partial t} \quad , \tag{33}$$

with R_0, L_0, G_0, and C_0 as per-unit-length (p.-u.-l.) parameters, being constant for uniform TLs, and with a length l.

When considering zero initial voltage and current distributions, $v(0,x) = 0$ and $i(0,x) = 0$, and incorporating boundary conditions, we get the Laplace-domain solution in the forms

$$V(s,x) = V_i(s)\frac{Z_c(s)}{Z_i(s)+Z_c(s)} \cdot \frac{e^{-\gamma(s)x} + \rho_2(s)e^{-\gamma(s)[2l-x]}}{1-\rho_1(s)\rho_2(s)e^{-2\gamma(s)l}}, \tag{34}$$

$$I(s,x) = V_i(s)\frac{1}{Z_i(s)+Z_c(s)} \cdot \frac{e^{-\gamma(s)x} - \rho_2(s)e^{-\gamma(s)[2l-x]}}{1-\rho_1(s)\rho_2(s)e^{-2\gamma(s)l}}, \tag{35}$$

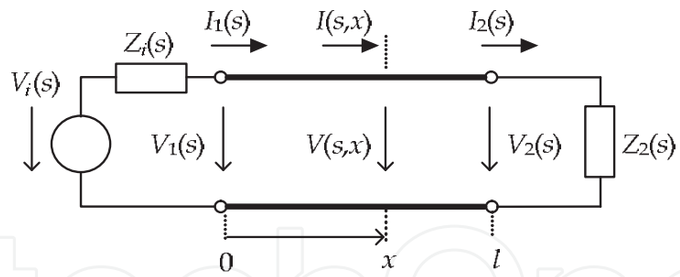

Fig. 9. Laplace-domain model of transmission line with linear terminations

where $Z_c(s)$ and $\gamma(s)$ are a characteristic impedance and a propagation constant, respectively,

$$Z_c(s) = \sqrt{\frac{R_0 + sL_0}{G_0 + sC_0}} \quad , \quad \gamma(s) = \sqrt{(R_0 + sL_0)(G_0 + sC_0)} \quad , \tag{36}$$

and $\rho_1(s)$ and $\rho_2(s)$ are reflection coefficients at the TL beginning and end, respectively,

$$\rho_1(s) = \frac{Z_i(s) - Z_c(s)}{Z_i(s) + Z_c(s)} \quad , \quad \rho_2(s) = \frac{Z_2(s) - Z_c(s)}{Z_2(s) + Z_c(s)} \, . \tag{37}$$

In a general case of lossy TLs, the time-domain solutions cannot be found by an analytical method, thus the only way is to use some numerical technique.
As an example, let us consider the TL of a length l = 1m, with p.-u.-l. parameters R_0 = 1mΩ, L_0 = 600nH, G_0 = 2mS, and C_0 = 80pF, terminated by resistive elements Z_i = 10Ω, Z_2 = 1kΩ, and excited by the voltage source waveform $v_i(t) = \sin^2(\pi t/2 \cdot 10^{-9})$, $0 \le t \le 2 \cdot 10^{-9}$, and $v_i(t) = 0$, otherwise, with the Laplace transform

$$V_i(s) = \frac{2\pi^2 \left[1 - \exp(-2 \cdot 10^{-9} s)\right]}{s\left[(2 \cdot 10^{-9} s)^2 + 4\pi^2\right]} \, . \tag{38}$$

The Fig. 10 shows time dependences at the beginning, the centre, and the end of the TL, while the 1D NILT is called as `niltcv('Vs',4e-8,'p1');` where the function `Vs` is defined as

```
function f=Vs(s)
l=1; x=[0,l/2,l];
Ro=1e-3; Lo=600e-9; Go=2e-3; Co=80e-12;
Zi=10; Z2=1e3;
Vi=2*pi^2*(1-exp(-2e-9*s))./s./((2e-9*s).^2+4*pi^2);
Z=Ro+s*Lo; Y=Go+s*Co; Zc=sqrt(Z./Y); gam=sqrt(Z.*Y);
ro1=(Zi-Zc)./(Zi+Zc); ro2=(Z2-Zc)./(Z2+Zc);
Ks=Vi./(Zi+Zc)./(1-ro1.*ro2.*exp(-2*gam*l));
for k=1:length(x)
    f(k,:)=Ks.*Zc.*(exp(-gam*x(k))+ro2.*exp(-gam*(2*l-x(k))));
end
```

Similarly, current waveforms can be computed by the above function slightly modified according to (35). Both waveforms are depicted in Fig. 10.
Finally, it will be shown, how to obtain three-dimensional graphical results representing voltage and current distributions along the TL. Besides a possibitity to use the `for` cycle, as

shown in the function `Vs` above, another method based on 3D arrays will be applied, see the function `Vsx` below:

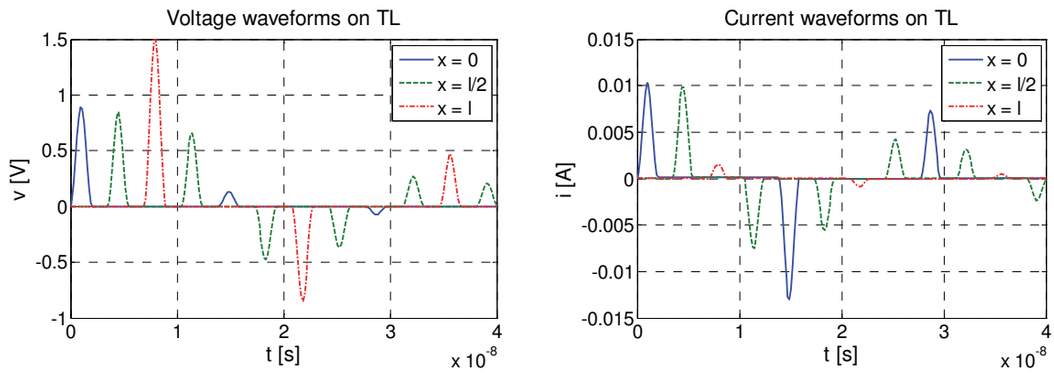

Fig. 10. Numerically computed TL voltage and current waveforms

```
function f=Vsx(s)
global x;
l=1;
Ro=1e-3; Lo=600e-9; Go=2e-3; Co=80e-12;
Zi=10; Z2=1e3;
x=linspace(0,l,65);       % 65 points along TL chosen
[S,X]=meshgrid(s,x);
Vi=2*pi^2*(1-exp(-2e-9*S))./S./((2e-9*S).^2+4*pi^2);
Z=Ro+S*Lo; Y=Go+S*Co; Zc=sqrt(Z./Y); gam=sqrt(Z.*Y);
ro1=(Zi-Zc)./(Zi+Zc); ro2=(Z2-Zc)./(Z2+Zc);
Ks=Vi./(Zi+Zc)./(1-ro1.*ro2.*exp(-2*gam*l));
f=Ks.*Zc.*(exp(-gam.*X)+ro2.*exp(-gam.*(2*l-X)));
```

In this case, the 1D NILT algorithm in Tab. 5 is called as `niltcv('Vsx',2e-8,'p3');` that is the `plott3` function is used for the plotting, see Tab. 6, and a time limit is half of that in Fig. 10 to get well-observable results. Again, the current distributions can be gained via the above function slightly modified according to (35). Both 3D graphs are depicted in Fig. 11.

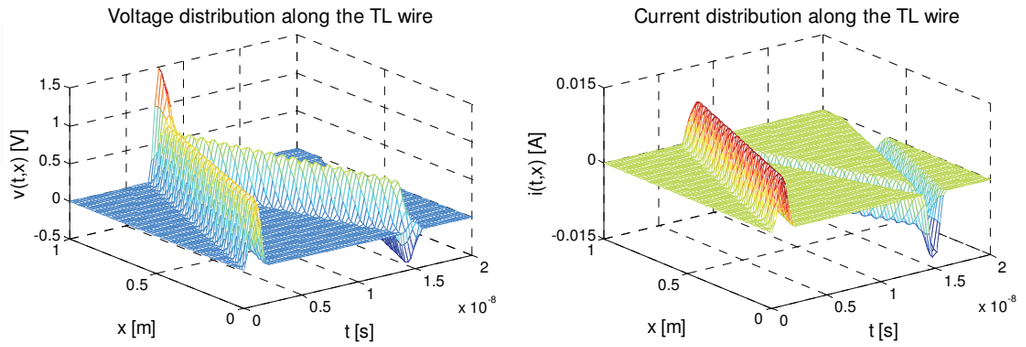

Fig. 11. Numerically computed TL voltage and current distributions

3.2 Two-dimensional NILT algorithm application

A two-dimensional Laplace transform can generally be used for the solution of linear partial differencial equations with two variables. The advantage is that we get completely algebraic

equations leading to much easier solution in the Laplace domain. A final step in the solution is then the utilization of the 2D NILT algorithm to get results in the original domain. Such a possibility will be shown on the example of telegraphic equations describing transmission lines, and results will be compared with the 1D NILT approach.

3.2.1 Application for transmission line simulation

Herein, rather less conventional approach for the simulation of voltage and/or current distributions along the TLs will be discussed. As is obvious from the telegraphic equations (33) they can be transformed not only with respect to the time t, which was matter of the previous paragraph, but also with respect to the geometrical coordinate x to get completely algebraic equations. After performing such the Laplace transforms, incorporating boundary conditions given by the terminating circuits, and considering again zero initial voltage and current distributions, $v(0,x) = 0$ and $i(0,x) = 0$, we get (Valsa & Brančík, 1998b)

$$V(s,q) = \frac{qV_1(s) - \gamma(s)Z_c(s)I_1(s)}{q^2 - \gamma^2(s)}, \tag{39}$$

$$I(s,q) = \frac{qI_1(s) - \dfrac{\gamma(s)}{Z_c(s)}V_1(s)}{q^2 - \gamma^2(s)}, \tag{40}$$

where $V_1(s) = V(s,0)$ and $I_1(s) = I(s,0)$ are given by (34) and (35), respectively, see also Fig. 9. Thus the 2D NILT function according to Tab. 3 can be called as `nilt2c('Vsq',2e-8,1);` leading to the same graphical results as are shown in Fig. 11. The function `Vsq` can be of the form as stated below. The current distribution is obtained via the same function slightly modified according to (40).

```
function f=Vsq(s,q)
l=1; Zi=10; Z2=1e3;
Ro=1e-3; Lo=600e-9; Go=2e-3; Co=80e-12;
Vi=2*pi^2*(1-exp(-2e-9*s))./s./((2e-9*s).^2+4*pi^2);
Z=Ro+s*Lo; Y=Go+s*Co; Zc=sqrt(Z./Y); gam=sqrt(Z.*Y);
ro1=(Zi-Zc)./(Zi+Zc); ro2=(Z2-Zc)./(Z2+Zc);
Ks=Vi./(Zi+Zc)./(1-ro1.*ro2.*exp(-2*gam*l));
V1=Ks.*Zc.*(1+ro2.*exp(-2*gam*l));
I1=Ks.*(1-ro2.*exp(-2*gam*l));
f=(q.*V1-Zc.*gam.*I1)./(q.^2-gam.^2);
```

One can notice an interesting thing, namely getting both voltage and current graphs by a single computation step. It is enabled by putting together the voltage and current transforms forming respectively real and imaginary parts of an artificial complex transform, and letting active the program command for the plotting the imaginary part of the original function, see Tab. 3. In our example, if the bottom line in the `Vsq` function is changed to

```
f=((q.*V1-Zc.*gam.*I1)+j*(q.*I1-gam./Zc.*V1))./(q.^2-gam.^2);
```

then both graphs in Fig. 11 are obtained simultaneously. The same possibility exists for the 1D NILT functions discussed earlier. There is no obvious physical meaning of such articifial complex transfoms, it is only a formal tool for inverting two transforms in parallel instead.

3.3 Multidimensional LT in nonlinear electrical circuits simulation

As is known some classes of nonlinear systems can be described through a Volterra series expansion, accurately enough from the practical point of view, when a response $v(t)$ to a stimulus $i(t)$ has a form (Schetzen, 2006)

$$v(t) = \sum_{n=1}^{\infty} v_n(t) \,, \tag{41}$$

where the terms of the infinite sum are

$$v_n(t) = \int_{-\infty}^{\infty} \cdots \int_{n-fold} \int_{-\infty}^{\infty} h_n(\tau_1, \tau_2, \ldots, \tau_n) \prod_{k=1}^{n} i(t-\tau_k) d\tau_k \,, \tag{42}$$

with $h_n(\tau_1, \tau_2, \ldots, \tau_n)$ as an n-th order Volterra kernel, called as a nonlinear impulse response as well. The Fig. 12 shows these equations in their graphical form.
By introducing new variables, $t_1 = t_2 = \ldots = t_n = t$, and by using the n-dimensional Laplace transform (1), the n-fold convolution integral (42) leads to a Laplace domain response

$$V_n(s_1, s_2, \cdots, s_n) = H_n(s_1, s_2, \cdots, s_n) \prod_{k=1}^{n} I(s_k) \,, \tag{43}$$

with $H_n(s_1, s_2, \ldots, s_n)$ as a nonlinear transfer function.

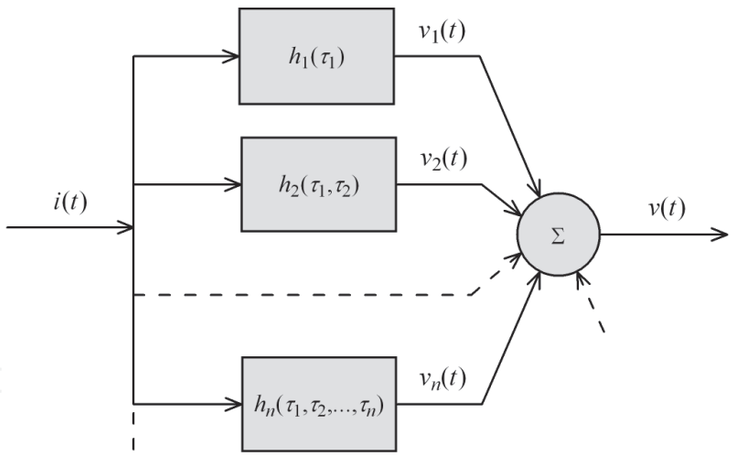

Fig. 12. Nonlinear system response via Volterra series expansion

A few methods are at disposal to find the transfer function for a given nonlinear system, like a harmonic input method, e.g. (Bussgang at al., 1974; Karmakar, 1980). Further procedure is usually based on the association of variables, (J. Debnath & N.C. Debnath, 1991; Reddy & Jagan, 1971), transforming $V_n(s_1, s_2, \ldots, s_n)$ into the function of a single variable $V_n(s)$, and enabling to use a one-dimensional ILT to get the required terms $v_n(t)$ in (41). In contrast to this procedure, it is also possible to determine the terms $v_n(t_1, t_2, \ldots, t_n)$ by the use of the n-dimensional ILT, considering $t_1 = t_2 = \ldots = t_n = t$ in the result as a final step. That is why the above NILT procedures can be adapted in this respect being able to serve as a tool for the nonlinear circuits transient simulation.

3.3.1 Utilization of 1D to 3D NILTs for nearly nonlinear circuits

The utilization of the NILT methods developed, up to three-dimensional case, will be shown on the solution of a nearly nonlinear circuit in Fig. 13. As can be observed this is just Fig. 7 modified to introduce a nonlinearity via G_2 conductance.

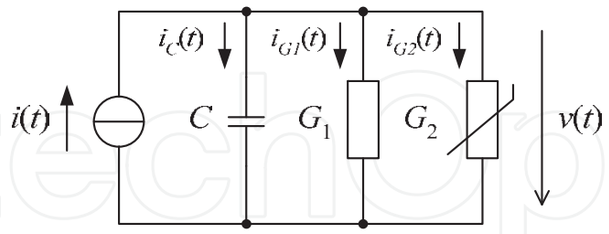

Fig. 13. Electrical circuit with nonlinear resistive element G_2

Assuming a square nonlinearity, a circuit equation is

$$C\frac{dv(t)}{dt} + G_1 v(t) + G_2 v^2(t) = i(t)\,. \tag{44}$$

By using the harmonic input method, and limiting the solution on the first three terms only, we get the nonlinear transfer functions for (43) as

$$H_1(s_1) = \left(s_1 C + G_1\right)^{-1}, \tag{45}$$

$$H_2(s_1, s_2) = -G_2 H_1(s_1) H_1(s_2) H_1(s_1 + s_2)\,, \tag{46}$$

$$H_3(s_1, s_2, s_3) = -\frac{G_2}{3}\left[H_1(s_1)H_2(s_2, s_3) + H_1(s_2)H_2(s_1, s_3) + H_1(s_3)H_2(s_1, s_2)\right] H_1(s_1 + s_2 + s_3)\,. \tag{47}$$

Let us use an exciting current and its Laplace transform as

$$i(t) = I_0 e^{-at}\underline{1}(t) \quad \mapsto \quad I(s) = \frac{I_0}{s + a}\,, \tag{48}$$

$a \geq 0$. The substitution (45) – (48) into (43) gives us respective Laplace-domain responses which will undergo the 1D, 2D and 3D NILT algorithms, respectively. We can write

$$\begin{aligned} v(t) = v_1(t) + v_2(t) + v_3(t) &= v_1(t_1)\big|_{t_1 = t} + v_2(t_1, t_2)\big|_{t_1 = t_2 = t} + v_3(t_1, t_2, t_3)\big|_{t_1 = t_2 = t_3 = t} = \\ &= \mathbb{L}_1^{-1}\left[V_1(s_1)\right]\Big|_{t_1 = t} + \mathbb{L}_2^{-1}\left[V_2(s_1, s_2)\right]\Big|_{t_1 = t_2 = t} + \mathbb{L}_3^{-1}\left[V_3(s_1, s_2, s_3)\right]\Big|_{t_1 = t_2 = t_3 = t} \end{aligned}, \tag{49}$$

with $\mathbb{L}_k^{-1}[.]$ as a k-dimensional ILT. Thereby, a time-consuming association of variables can be omitted, e.g. (Wambacq & Sansen, 2010). Individual terms $v_k(t)$ are depicted in Fig. 14, for values agreeing with the linear circuit version in Fig. 7. The current $i(t)$ is defined by $a = 0$ (a unit step), and $a = 5$ (an exponential impuls), compare the first two columns in Tab. 4.

The resultant voltage responses computed according to (49) are shown in Fig. 15, including relative errors, where also dependences on Volterra series orders are presented.

The relative errors above were computed via a Matlab ODE45 Runge-Kutta function applied directly to the nonlinear ODE (44). As expected, more Volterra terms lead to more accurate results, see also (Brančík, 2009) where up to 2nd-order terms were considered, and respective Matlab listings are presented.

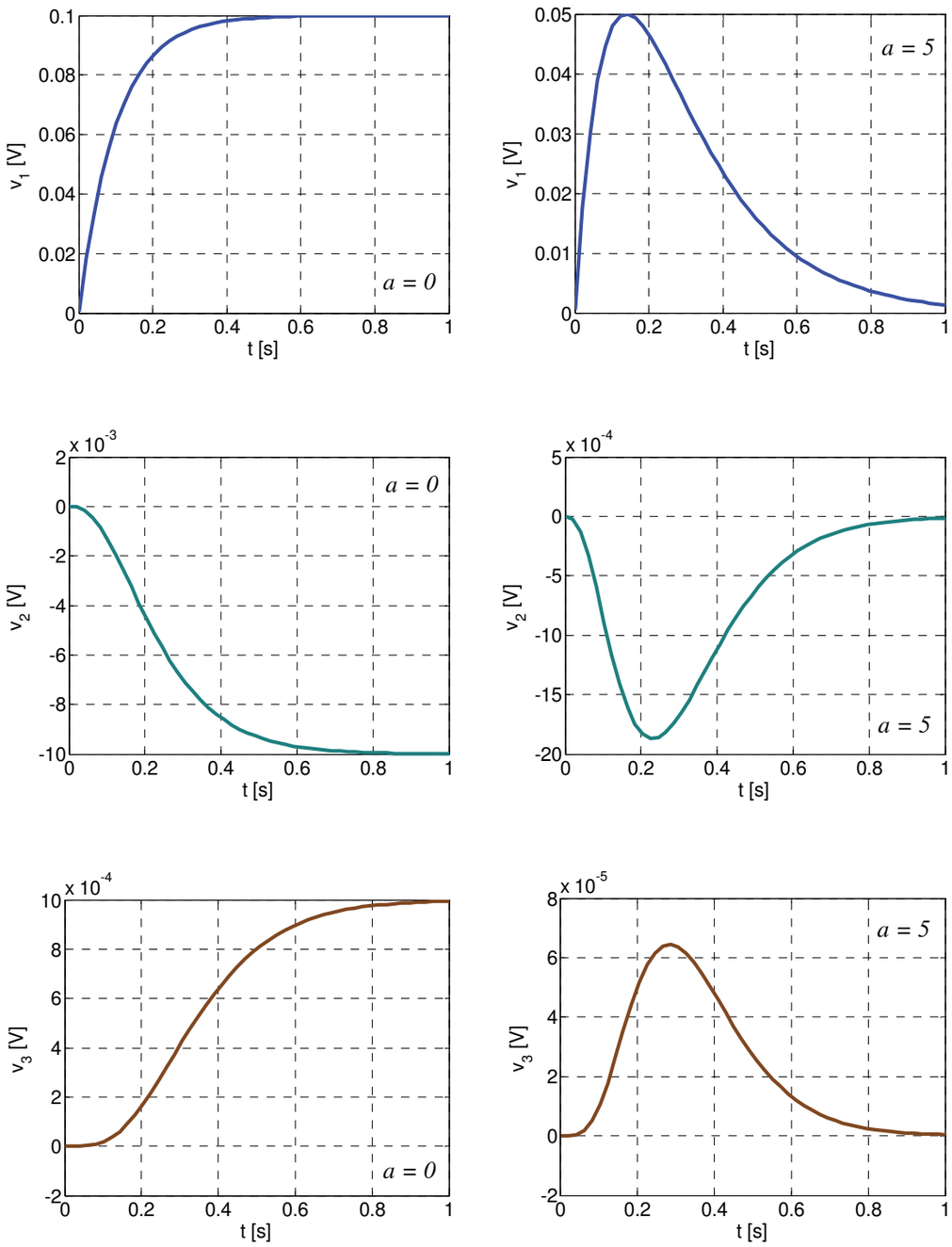

Fig. 14. Numerical inversions leading to voltage response Volterra series terms

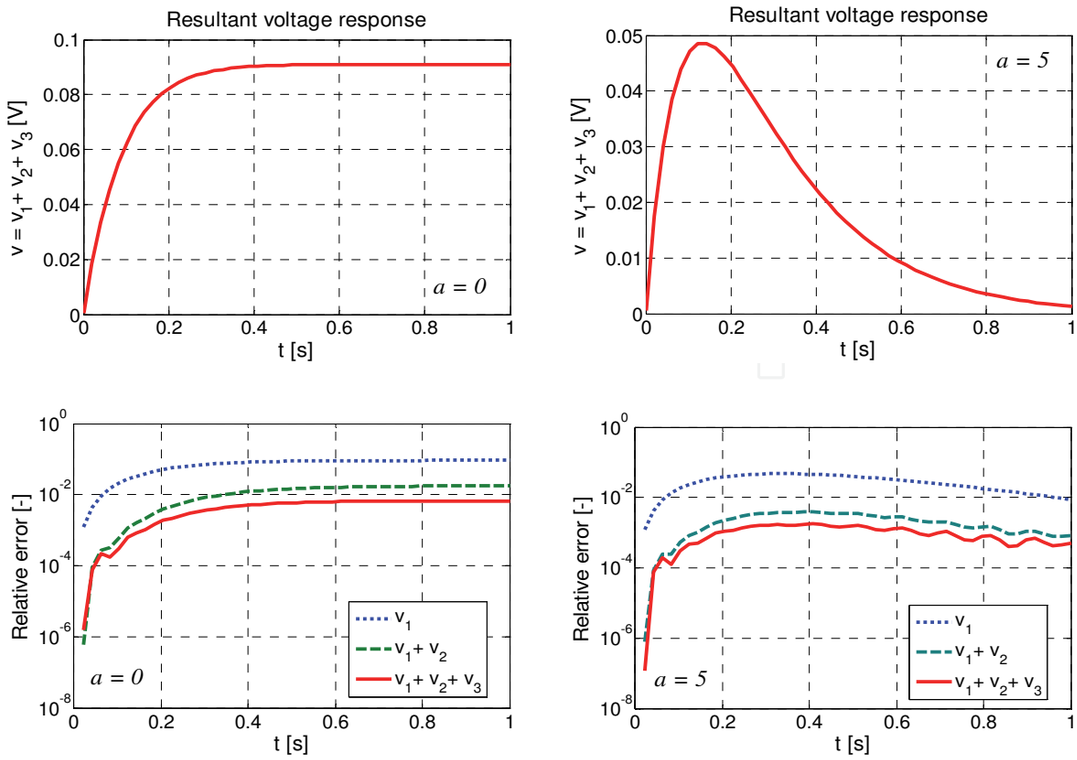

Fig. 15. Resultant voltage responses and relative errors

4. Conclusion

The paper dealt with a specific class of techniques for the numerical inversion of Laplace transforms, namely based on a complex Fourier series approximation, and connected with a quotient-difference algorithm to accelerate the convergence of infinite series arising in the approximate formulae. The 1D to 3D NILT techniques have been programmed in the Matlab language (R2007b), and most important ones provided as the Matlab function listings. To guide readers all the algorithms were explained on selected examples from field of electrical engineering, including right callings of the functions. In contrast to most others the NILT methods here developed are utilizable to numerically invert complex Laplace transforms, leading to complex originals, which can be useful for some special purposes. As has resulted from error analyses the accuracies range relative errors from 10^{-8} to 10^{-10} without difficulties which is acceptable for most of practical needs. Based on Matlab functions presented, one could further generalize a vector version of the 1D NILT function towards a matrix one, enabling e.g. to simulate multiconductor transmission line systems, as is shown in (Brančík, 1999), where, however, an alternative technique, so-called ε algorithm, has been applied to accelerate the convergence of infinite series. According to the author's knowledge, the paper presented ranks among few summary works describing multidimensional NILT techniques, covering Matlab listings beyond, based just on a complex Fourier series approximation, and in conjunction with the quotient-difference algorithm, which seems to be more numerically stable compared to the ε algorithm mentioned above.

5. Acknowledgment

Research described in this paper was supported by the Czech Ministry of Education under the MSM 0021630503 research project MIKROSYN, the European Community's Seventh Framework Programme under grant agreement no. 230126, and the project CZ.1.07/2.3.00/20.0007 WICOMT of the operational program Education for competitiveness.

6. References

Brančík, L. (1999). Programs for fast numerical inversion of Laplace transforms in Matlab language environment. *Proceedings of 7th Conference MATLAB'99,* pp. 27-39, ISBN 80-7080-354-1, Prague, Czech Republic, November 3, 1999

Brančík, L. (2005). Elaboration of FFT-based 2D-NILT methods in terms of accuracy and numerical stability. *Przeglad Elektrotechniczny,* Vol. 81, No. 2, (February 2005), pp. 84-89, ISSN 0033-2097

Brančík, L. (2007a). Numerical Inversion of two-dimensional Laplace transforms based on partial inversions. *Proceedings of 17th International Conference Radioelektronika 2007,* pp. 451-454, ISBN 1-4244-0821-0, Brno, Czech Republic, April 24-25, 2007

Brančík, L. (2007b). Modified technique of FFT-based numerical inversion of Laplace transforms with applications. *Przegląd Elektrotechniczny,* Vol. 83, No. 11, (November 2007), pp. 53-56, ISSN 0033-2097

Brančík, L. (2009). Numerical ILTs applied to weakly nonlinear systems described by second-order Volterra series. *ElectroScope*, [online], Special Issue on Conference EDS 2009, 4 pages, Available from http://electroscope.zcu.cz, ISSN 1802-4564

Brančík, L. (2010a). Utilization of NILTs in simulation of nonlinear systems described by Volterra series. *Przeglad Elektrotechniczny,* Vol. 86, No. 1, (January 2010), pp. 68-70, ISSN 0033-2097

Brančík, L. (2010b). Numerical inversion of 3D Laplace transforms for weakly nonlinear systems solution. *Proceedings of 20th International Conference Radioelektronika 2010*, pp. 221-224, ISBN 978-1-4244-6319-0, Brno, Czech Republic, April 19-21, 2010

Brančík, L. (2010c). Technique of 3D NILT based on complex Fourier series and quotient-difference algorithm. *Proceedings of 2010 IEEE International Conference on Electronics, Circuits, and Systems ICECS2010,* pp. 207-210, ISBN 978-1-4244-8156-9, Athens, Greece, December 12-15, 2010

Brančík, L. (2011). Error analysis at numerical inversion of multidimensional Laplace transforms based on complex Fourier series approximation. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences,* Vol. E94-A, No. 3, (March 2011), p. 999-1001, ISSN 0916- 8508

Bussgang, J.J.; Ehrman, L. & Graham, J.W. (1974). Analysis of nonlinear systems with multiple inputs. *Proceedings of the IEEE*, Vol. 62, No. 8, (August 1974), pp. 1088-1119, ISSN 0018-9219

Cohen, A.M. (2007). *Numerical methods for Laplace transform inversion.* Springer Science, ISBN 978-0-387-28261-9, New York, U.S.A

Debnath, J. & Debnath, N.C. (1991). Associated transforms for solution of nonlinear equations. *International Journal of Mathematics and Mathematical Sciences,* Vol. 14, No. 1, (January 1991), pp. 177-190, ISSN 0161-1712

DeHoog, F.R.; Knight, J.H. & Stokes, A.N. (1982). An improved method for numerical inversion of Laplace transforms. *SIAM Journal on Scientific and Statistical Computing,* Vol. 3, No. 3, (September 1982), pp. 357-366, ISSN 0196-5204

Hwang, C.; Guo, T.-Y. & Shih, Y.-P. (1983). Numerical inversion of multidimensional Laplace transforms via block-pulse functions. *IEE Proceedings D - Control Theory & Applications*, Vol. 130, No. 5, (September 1983), pp. 250-254, ISSN 0143-7054

Hwang, C. & Lu, M.-J. (1999). Numerical inversion of 2-D Laplace transforms by fast Hartley transform computations. *Journal of the Franklin Institute,* Vol. 336, No. 6, (August 1999), pp. 955-972, ISSN 0016-0032

Karmakar, S.B. (1980). Laplace transform solution of nonlinear differential equations. *Indian Journal of Pure & Applied Mathematics,* Vol. 11, No. 4, (April 1980), pp. 407-412, ISSN 0019-5588

Macdonald, J.R. (1964). Accelerated convergence, divergence, iteration, extrapolation, and curve fitting, *Journal of Applied Physics*, Vol. 35, No. 10, (February 1964), pp. 3034-3041, ISSN 0021-8979

McCabe, J.H. (1983). The quotient-difference algorithm and the Padé table: An alternative form and a general continued fraction. *Mathematics of Computation*, Vol. 41, No. 163, (July 1983), pp. 183-197, ISSN 0025-5718

Reddy, D.C. & Jagan, N.C. (1971). Multidimensional transforms: new technique for the association of variebles. *Electronics Letters*, Vol. 7, No. 10, (May 1971), pp. 278 – 279, ISSN 0013-5194

Rutishauser, H. (1957). *Der quotienten-differenzen-algorithmus.* Birkhäuser Verlag, Basel, Schweiz

Schetzen, M. (2006). *The Volterra and Wiener theories of nonlinear systems.* Krieger Publishing, ISBN 978-1-575-24283-5, Melbourne, Florida, U.S.A

Singhal, K.; Vlach, J. & Vlach, M. (1975). Numerical inversion of multidimensional Laplace transform. *Proceedings of the IEEE*, Vol. 63, No. 11, (November 1975), pp. 1627-1628, ISSN 0018-9219

Valsa, J. & Brančík, L. (1998a). Approximate formulae for numerical inversion of Laplace transforms. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields,* Vol. 11, No. 3, (May-June 1998), pp. 153-166, ISSN 0894-3370

Valsa, J. & Brančík, L. (1998b). Time-domain simulation of lossy transmission lines with arbitrary initial conditions. *Proceedings of Advances in Systems, Signals, Control and Computers*, Vol. III, pp. 305-307, ISBN 0-620-23136-X, Durban, South Africa, September 22-24, 1998

Wambacq, P. & Sansen, W.M.C. (2010). *Distortion analysis of analog integrated circuits.* Kluwer Academic Publishers, ISBN 978-1-4419-5044-4, Boston, U.S.A

Wu, J.L.; Chen, C.H. & Chen, C.F. (2001). Numerical inversion of Laplace transform using Haar wavelet operational matrices. *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications,* Vol. 48, No. 1, (January 2001), pp. 120-122, ISSN 1057-7122

MATLAB for Engineers - Applications in Control, Electrical Engineering, IT and Robotics

Edited by Dr. Karel Perutka

ISBN 978-953-307-914-1
Hard cover, 512 pages
Publisher InTech
Published online 13, October, 2011
Published in print edition October, 2011

The book presents several approaches in the key areas of practice for which the MATLAB software package was used. Topics covered include applications for: -Motors -Power systems -Robots -Vehicles The rapid development of technology impacts all areas. Authors of the book chapters, who are experts in their field, present interesting solutions of their work. The book will familiarize the readers with the solutions and enable the readers to enlarge them by their own research. It will be of great interest to control and electrical engineers and students in the fields of research the book covers.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Lubomîr Brančîk (2011). Numerical Inverse Laplace Transforms for Electrical Engineering Simulation, MATLAB for Engineers - Applications in Control, Electrical Engineering, IT and Robotics, Dr. Karel Perutka (Ed.), ISBN: 978-953-307-914-1, InTech, Available from: http://www.intechopen.com/books/matlab-for-engineers-applications-in-control-electrical-engineering-it-and-robotics/numerical-inverse-laplace-transforms-for-electrical-engineering-simulation

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen