MATLAB PROGRAMS FOR MATRIX EXPONENTIAL FUNCTION DERIVATIVE EVALUATION

Lubomír Brančík

Institute of Radio Electronics, Faculty of Electrical Engineering and Communication
Brno University of Technology

Abstract

The paper deals with six approaches how to determine a derivative of the matrix exponential function in the Matlab language environment. Namely, a Taylor series expansion, an augmented matrix utilization, an eigenvalues decomposition, a Laplace transform approach, a convolution integral evaluation and a Padé approximation method are discussed in the paper. Some of the above methods are connected with a scaling and squaring process to improve the stability. Besides, possible Matlab listings are shown at each method as well.

1 Introduction & Motivation Comment

Six possible methods how to compute a first derivative of the matrix exponential function are discussed while using the Matlab language environment. A usefulness for such the computation appears in many branches of the electrical engineering simulation. For example, in the field of the circuit theory, it arises in the process of a sensitivity determination at the electrical circuits containing multiconductor transmission lines (MTL) [1, 2]. The properties of an (*n*+1)-conductor TL in the *s*–domain can be expressed using a chain matrix

$$\mathbf{\Phi}(x,s) = e^{\mathbf{M}(s)x} \ , \tag{1}$$

where x denotes a geometric coordinate along the MTL and $\mathbf{M}(s)$ is the $2n$-order square matrix

$$\mathbf{M}(s) = \begin{bmatrix} \mathbf{0} & -\mathbf{Z}_0(s) \\ -\mathbf{Y}_0(s) & \mathbf{0} \end{bmatrix} , \tag{2}$$

with

$$\mathbf{Z}_0(s) = \mathbf{R}_0 + s\mathbf{L}_0 \quad , \quad \mathbf{Y}_0(s) = \mathbf{G}_0 + s\mathbf{C}_0 \tag{3}$$

as the n-order series impedance and shunting admittance matrices, respectively, $\mathbf{R}_0$, $\mathbf{L}_0$, $\mathbf{G}_0$ and $\mathbf{C}_0$ as MTL per-unit-length matrices, and $\mathbf{0}$ as a zero matrix. The sensitivities can be got resulting from the first derivative of (1), with respect to a parameter $\gamma \in \mathbf{M}(s)$ [1]. Thus, γ is the element of one of the per-unit-length matrices stated above, while x is considered as a constant in this case. As will be shown later, for all the techniques the first derivative of the matrix at the exponent is required. In case of our application example, depending on γ and considering (2) and (3), it is given by Table 1 [1].

TABLE 1. DERIVATIVES $\partial\mathbf{M}(S)/\partial\gamma$ NEEDED FOR DETERMINING $\partial\mathbf{\Phi}(X,S)/\partial\gamma$

$\gamma \in \mathbf{R}_0$	$\gamma \in \mathbf{L}_0$	$\gamma \in \mathbf{G}_0$	$\gamma \in \mathbf{C}_0$
$\begin{bmatrix} \mathbf{0} & -\frac{\partial \mathbf{R}_0}{\partial \gamma} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$	$\begin{bmatrix} \mathbf{0} & -s\frac{\partial \mathbf{L}_0}{\partial \gamma} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$	$\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -\frac{\partial \mathbf{G}_0}{\partial \gamma} & \mathbf{0} \end{bmatrix}$	$\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -s\frac{\partial \mathbf{C}_0}{\partial \gamma} & \mathbf{0} \end{bmatrix}$

As is obvious finding the above derivatives is already straightforward. Some examples of sensitivities computation at the MTL systems can be found e.g. in [1-3].

To generalize our approach for any type of matrix, however, let us denote the matrix in view as $\mathbf{M} \equiv \mathbf{M}(\gamma)$, when its dependence on γ is evident. The matrix exponential function is then formulated as

$$\mathbf{\Phi}(\gamma,x) = e^{\mathbf{M}(\gamma)x} \ , \tag{4}$$

with x standing for any real parameter, and $\partial\mathbf{\Phi}(\gamma,x)/\partial\gamma$ is what we want to determine.

2 Taylor Series Expansion with Scaling & Squaring

As is well known, the matrix exponential function (4) can be expanded into the Taylor series as

$$\mathbf{\Phi}(\gamma,x)=\sum_{k=0}^{\infty}\frac{x^k}{k!}\mathbf{M}^k(\gamma)\ . \tag{5}$$

Then, we can write

$$\frac{\partial\mathbf{\Phi}(\gamma,x)}{\partial\gamma}=\sum_{k=1}^{\infty}\frac{x^k}{k!}\frac{\partial\mathbf{M}^k(\gamma)}{\partial\gamma}\ , \tag{6}$$

where a derivative of the k–th power of the matrix is stated as follows. If we consider e.g. an identity $\mathbf{M}^k(\gamma)=\mathbf{M}(\gamma)\mathbf{M}^{k-1}(\gamma)$ the first derivative is written as

$$\frac{\partial\mathbf{M}^k(\gamma)}{\partial\gamma}=\frac{\partial\mathbf{M}(\gamma)}{\partial\gamma}\mathbf{M}^{k-1}(\gamma)+\mathbf{M}(\gamma)\frac{\partial\mathbf{M}^{k-1}(\gamma)}{\partial\gamma}\ , \tag{7}$$

that can be processed recursively, starting with $k=2$.

In practice, the infinite sum (6) will only be evaluated for some finite number of terms. The convergence sufficiently rapid is ensured if the product $\mathbf{M}(\gamma)x$ is close enough to a null matrix. Therefore, the derivative is found for a transformed matrix exponential function $\mathbf{\Phi}(\gamma,x/r)$, for such the r in order to meet the Euclidian norm $\|\mathbf{M}(\gamma)x/r)\| <$ (0.15 to 0.20) (the values are recommended in [4] for 3 × 3 matrices). This temporal result is transformed back to get the original derivative. A procedure is based on so-called scaling and squaring [4]. First, considering a formula based on (4) as

$$\mathbf{\Phi}(\gamma,x)=\mathbf{\Phi}^r(\gamma,x/r)\ , \tag{8}$$

then if $r=2^M$, M integer, a restoring $\mathbf{\Phi}(\gamma,x)$ from $\mathbf{\Phi}(\gamma,x/r)$ can fast be performed by a squaring process

$$\mathbf{\Phi}(\gamma,x/2^{m-1})=\mathbf{\Phi}^2(\gamma,x/2^m)\ , \tag{9}$$

successively for $m=M, M\text{-}1,\ldots,1$. Then, a recursive formula to restore $\partial\mathbf{\Phi}(\gamma,x)/\partial\gamma$ from $\partial\mathbf{\Phi}(\gamma,x/r)/\partial\gamma$ follows (9) as

$$\frac{\partial\mathbf{\Phi}(\gamma,x/2^{m-1})}{\partial\gamma}=\frac{\partial\mathbf{\Phi}(\gamma,x/2^m)}{\partial\gamma}\mathbf{\Phi}(\gamma,x/2^m)+\mathbf{\Phi}(\gamma,x/2^m)\frac{\partial\mathbf{\Phi}(\gamma,x/2^m)}{\partial\gamma}\ , \tag{10}$$

processing it M-times, successively for $m=M, M\text{-}1,\ldots,1$. This recursive formula contains (9) as its inner part. During evaluating (6) the successive increments are used to obtain temporal errors enabling to stop the summation process. A possible Matlab function dexpmt listing is shown below.

```
%*************************************************************************************************
function [F,dF]=dexpmt(M,dM)                                          % by Lubomír Brančík, 2008
% Scale M by power of 2 so that its norm is < 1/2
[f,e]=log2(norm(M,'inf'));
r=max(0,e+1);
M=M/2^r; dM=dM/2^r;
% Taylor series expansion of exp(M) and diff[exp(M)]
FM=eye(size(M)); dFM=dM;
F=FM+M; dF=dM;
l=1; n=1;
incF=1;
while incF>1e-16
    n=n+1; l=l*n;
    FM=FM*M;
    dFM=dM*FM+M*dFM;
    dF=dF+dFM/l;
    F=F+FM*M/l;
    incF=sum(sum(abs(FM)+abs(dFM)));
end
% Undo scaling by repeated squaring
for k=1:r
    dF=dF*F+F*dF;
    F=F*F;
end
%*************************************************************************************************
```

Calling the function as [F,dF]=dexpmt(M,dM); leads to the evaluation of both matrix exponential function (stored in F) and its derivative (stored in dF), with M and its derivative dM as arguments.

3 Augmented Matrix Utilization

On principle the matrix exponential function (4) figures in the solution of the first-order matrix ordinary differential equation

$$\frac{d\mathbf{W}(\gamma,x)}{dx}=\mathbf{M}(\gamma)\mathbf{W}(\gamma,x)\ , \tag{11}$$

namely

$$\mathbf{W}(\gamma,x)=\mathbf{\Phi}(\gamma,x)\mathbf{W}(\gamma,0)\,, \tag{12}$$

with $\mathbf{W}(\gamma,0)$ as an initial condition. A further technique is based on the concept in [4]. When putting together (12) with its first derivative with respect to γ of the form

$$\frac{\partial\mathbf{W}(\gamma,x)}{\partial\gamma}=\frac{\partial\mathbf{\Phi}(\gamma,x)}{\partial\gamma}\mathbf{W}(\gamma,0)+\mathbf{\Phi}(\gamma,x)\frac{\partial\mathbf{W}(\gamma,0)}{\partial\gamma}\ , \tag{13}$$

then only one, an augmented matrix equation, can be written as

$$\begin{bmatrix}\mathbf{W}(\gamma,x)\\ \dfrac{\partial\mathbf{W}(\gamma,x)}{\partial\gamma}\end{bmatrix}=\begin{bmatrix}\mathbf{\Phi}(\gamma,x) & \mathbf{0}\\ \dfrac{\partial\mathbf{\Phi}(\gamma,x)}{\partial\gamma} & \mathbf{\Phi}(\gamma,x)\end{bmatrix}\cdot\begin{bmatrix}\mathbf{W}(\gamma,0)\\ \dfrac{\partial\mathbf{W}(\gamma,0)}{\partial\gamma}\end{bmatrix}\ . \tag{14}$$

Similarly, combining (11) with its first derivative with respect to γ of the form

$$\frac{d}{dx}\left(\frac{\partial\mathbf{W}(\gamma,x)}{\partial\gamma}\right)=\frac{\partial\mathbf{M}(\gamma)}{\partial\gamma}\mathbf{W}(\gamma,x)+\mathbf{M}(\gamma)\frac{\partial\mathbf{W}(\gamma,x)}{\partial\gamma}\ , \tag{15}$$

we get only one, an augmented matrix differential equation, in the form

$$\frac{d}{dx}\begin{bmatrix}\mathbf{W}(\gamma,x)\\ \dfrac{\partial\mathbf{W}(\gamma,x)}{\partial\gamma}\end{bmatrix}=\begin{bmatrix}\mathbf{M}(\gamma) & \mathbf{0}\\ \dfrac{\partial\mathbf{M}(\gamma)}{\partial\gamma} & \mathbf{M}(\gamma)\end{bmatrix}\cdot\begin{bmatrix}\mathbf{W}(\gamma,x)\\ \dfrac{\partial\mathbf{W}(\gamma,x)}{\partial\gamma}\end{bmatrix}. \tag{16}$$

The solution of the last differential equation can be written as

$$\begin{bmatrix}\mathbf{W}(\gamma,x)\\ \dfrac{\partial\mathbf{W}(\gamma,x)}{\partial\gamma}\end{bmatrix}=\exp\left(\begin{bmatrix}\mathbf{M}(\gamma) & \mathbf{0}\\ \dfrac{\partial\mathbf{M}(\gamma)}{\partial\gamma} & \mathbf{M}(\gamma)\end{bmatrix}x\right)\cdot\begin{bmatrix}\mathbf{W}(\gamma,0)\\ \dfrac{\partial\mathbf{W}(\gamma,0)}{\partial\gamma}\end{bmatrix}=\begin{bmatrix}\mathbf{\Psi}_{11}(\gamma,x) & \mathbf{\Psi}_{12}(\gamma,x)\\ \mathbf{\Psi}_{21}(\gamma,x) & \mathbf{\Psi}_{22}(\gamma,x)\end{bmatrix}\cdot\begin{bmatrix}\mathbf{W}(\gamma,0)\\ \dfrac{\partial\mathbf{W}(\gamma,0)}{\partial\gamma}\end{bmatrix}\ , \tag{17}$$

with the same initial conditions as are in (14). Now denoting $\mathbf{\Psi}(\gamma,x)=[\mathbf{\Psi}_{ij}(\gamma,x)]$, $i,j=1,2$, as a transient matrix, see the right side of (17), and comparing it with the system matrix in (14), $\partial\mathbf{\Phi}(\gamma,x)/\partial\gamma$ is equal to the homothetic submatrix $\mathbf{\Psi}_{21}(\gamma,x)$. The derivative $\partial\mathbf{M}(\gamma)/\partial\gamma$ can be stated easily. Besides, by using a Matlab language, the expm built-in function based on Padé approximation and scaling and squaring can be utilized with advantages to determine the transient matrix $\mathbf{\Psi}(\gamma,x)$. A possible Matlab function dexpma listing is shown below, calling the function corresponds to the preceding method.

```
%*****************************************************************************************
function [F,dF]=dexpma(M,dM)                                    % by Lubomír Brančík, 2008
N=length(M);
% Creating augmented matrix
AM=[M,zeros(N);dM,M];
% Matrix exponential function of augmented matrix - usage of Matlab expm
PSI=expm(AM);
% Extracting F and dF from PSI
F=PSI(1:N,1:N);
dF=PSI(N+1:2*N,1:N);
%*****************************************************************************************
```

4 Eigenvalues Decomposition

An another Matlab function can be utilized to get a matrix exponential function derivative, namely the eig built-in function for the matrix eigenvalues and eigenvectors calculation. A procedure is based on a concept in [5]. Let us assume $\mathbf{M}(\gamma)$ having distinct eigenvalues $q_j(\gamma)$, with associated

eigenvectors $\mathbf{u}_j(\gamma)$, $j = 1,2,...,N$. Now creating a diagonal matrix $\mathbf{Q}(\gamma) = \mathrm{diag}(q_1(\gamma),q_2(\gamma),...,q_N(\gamma))$ and a modal matrix $\mathbf{U}(\gamma)$, composed of $\mathbf{u}_j(\gamma)$ as its columns, a similarity transform leads to

$$\mathbf{M}(\gamma) = \mathbf{U}(\gamma)\mathbf{Q}(\gamma)\mathbf{U}^{-1}(\gamma) \ . \tag{18}$$

Then, the matrix exponential function (4) can be expressed as

$$\mathbf{\Phi}(\gamma,x) = \mathbf{U}(\gamma)e^{\mathbf{Q}(\gamma)x}\mathbf{U}^{-1}(\gamma) = \mathbf{U}(\gamma)\mathrm{diag}\left(e^{q_1(\gamma)x},e^{q_2(\gamma)x},\ldots,e^{q_N(\gamma)x}\right)\mathbf{U}^{-1}(\gamma)\,, \tag{19}$$

and its first derivative as [5]

$$\frac{\partial\mathbf{\Phi}(\gamma,x)}{\partial\gamma} = \mathbf{U}(\gamma)\mathbf{P}(\gamma,x)\mathbf{U}^{-1}(\gamma) \ . \tag{20}$$

In the last equation, $\mathbf{P}(\gamma,x)$ is a matrix built-up from the elements

$$p_{ij}(\gamma,x) = h_{ij}(\gamma)\frac{e^{q_i(\gamma)x} - e^{q_j(\gamma)x}}{q_i(\gamma) - q_j(\gamma)}\,, \quad i \neq j, \quad \text{and} \quad p_{ii}(\gamma,x) = h_{ii}(\gamma)xe^{q_i(\gamma)x}\,, \quad i = j\,, \tag{21}$$

where $h_{ij}(\gamma)$ are the (i,j)-th elements of the matrix

$$\mathbf{H}(\gamma) = \mathbf{U}^{-1}(\gamma)\frac{\partial\mathbf{M}(\gamma)}{\partial\gamma}\mathbf{U}(\gamma) \ . \tag{22}$$

The derivative $\partial\mathbf{M}(\gamma)/\partial\gamma$ can be stated easily. The calling Matlab eig function with the argument $\mathbf{M}(\gamma)$ will return both $\mathbf{Q}(\gamma)$ and $\mathbf{U}(\gamma)$ matrices. In case of repeated eigenvalues a more complex method based on a Jordan canonical decomposition should be applied instead, however, it happens rarely in practice [4, 5]. A possible Matlab function dexpme listing is shown below, its calling corresponds to the first method.

```
%*************************************************************************************************
function [F,dF]=dexpme(M,dM)                                        % by Lubomír Brančík, 2008
% Eigenvalues/eigenvectors decomposition - usage of Matlab eig
[U,Q]=eig(M);
eQr=exp(diag(Q));
H=U\dM*U;
P=diag(diag(H).*eQr);
for i=1:length(H)-1
    for j=i+1:length(H)
        RQ=(eQr(i)-eQr(j))/(Q(i,i)-Q(j,j));
        P(i,j)=H(i,j)*RQ; P(j,i)=H(j,i)*RQ;
    end
end
F=U*diag(eQr)/U;
dF=U*P/U;
%*************************************************************************************************
```

5 Laplace Transform Approach

This technique is based on the fact that the derivative $\partial\mathbf{\Phi}(\gamma,x)/\partial\gamma$ is easier to find after a Laplace transform of the matrix exponential function with respect to x is performed. It means its q–domain image is first found, then differentiated and finally inverted into the x–domain again. A commutativity property of the integration and derivative operations leads to the equality

$$\frac{\partial\mathbf{\Phi}(\gamma,x)}{\partial\gamma} = \mathbb{L}_x^{-1}\left\{\frac{\partial}{\partial\gamma}\mathbb{L}_x\left\{\mathbf{\Phi}(\gamma,x)\right\}\right\}\,, \tag{23}$$

where the Laplace transform can analytically be stated as

$$\mathbb{L}_x\left\{\mathbf{\Phi}(\gamma,x)\right\} = \int_0^\infty e^{\mathbf{M}(\gamma)x}e^{-qx}dx = \int_0^\infty e^{-(q\mathbf{I}-\mathbf{M}(\gamma))x}dx = \left(q\mathbf{I}\text{-}\mathbf{M}(\gamma)\right)^{-1}\,, \tag{24}$$

with $\mathbf{I}$ as an identity matrix. Based on (23) and (24), and doing some arrangements, the result is

$$\frac{\partial\mathbf{\Phi}(\gamma,x)}{\partial\gamma} = \mathbb{L}_x^{-1}\left\{\left(q\mathbf{I}\text{-}\mathbf{M}(\gamma)\right)^{-1}\frac{\partial\mathbf{M}(\gamma)}{\partial\gamma}\left(q\mathbf{I}\text{-}\mathbf{M}(\gamma)\right)^{-1}\right\} \ . \tag{25}$$

The derivative $\partial\mathbf{M}(\gamma)/\partial\gamma$ can be stated easily. The inverse Laplace transform has to be done numerically in general. To guarantee the sufficiently fast convergence the scaling and squaring are applied in according to (8) – (10). The matrix exponent part is scaled to obtain its Euklidian norm $\|\mathbf{M}(\gamma)x/r)\| < 0.5$. In such a case the inverse Laplace transform is solved for $x = 1$. It is possible to use various types of NILT techniques, see e.g. [6, 7]. Here, a NILT based on approximation of exp(qx) in the ILT Bromwich integral by a specially created infinite series, in conjunction with Euler transform, is applied [7]. A possible Matlab function dexpml listing is shown below, its calling follows the first method. Due to the NILT used, however, this function is applicable for real matrices **M** only.

```
%*************************************************************************************************
function [F,dF]=dexpml(M,dM)                                          % by Lubomír Brančík, 2008
N=length(M);
% Scale M by power of 2 so that its norm is < 1/2
[f,e]=log2(norm(M,'inf'));
r=max(0,e+1);
M=M/2^r; dM=dM/2^r;
% Setting up parameters of the NILT method
nsum=200;                                             % number of terms in a basic sum
ndif=10;                                              % number of terms for Euler transform
a=8;                                                  % error boundary proportional to exp(-4*a)
nd=1:ndif;
eul=fliplr(cumsum(factorial(ndif)./factorial(ndif+1-nd)./factorial(nd-1)));
kn=(-1).^(1:nsum+ndif);
F=inv(a*eye(N)-M)/2;
dF=2*F*dM*F;
for n=1:nsum
    qr=a+j*n*pi;
    LFr=inv(qr*eye(N)-M);
    qi=a+j*(n-0.5)*pi;
    LFi=inv(qi*eye(N)-M);
    F=F+kn(n)*(real(LFr)+imag(LFi));
    dF=dF+kn(n)*(real(LFr*dM*LFr)+imag(LFi*dM*LFi));
end
Fe=zeros(N);
dFe=zeros(N);
for n=nsum+1:nsum+ndif
    qr=a+j*n*pi;
    LFr=inv(qr*eye(N)-M);
    qi=a+j*(n-0.5)*pi;
    LFi=inv(qi*eye(N)-M);
    Fe=Fe+kn(n)*(real(LFr)+imag(LFi))*eul(n-nsum);
    dFe=dFe+kn(n)*(real(LFr*dM*LFr)+imag(LFi*dM*LFi))*eul(n-nsum);
end
F=exp(a)/2*(F+Fe/2^ndif);
dF=exp(a)/2*(dF+dFe/2^ndif);
% Undo scaling by repeated squaring
for k=1:r
    dF=dF*F+F*dF;
    F=F*F;
end
%*************************************************************************************************
```

6 Convolution Integral Evaluation

This method is a consequence of the Laplace transform approach in the last section. Namely, an argument of the inverse Laplace transform in (25) has the form of a product of matrices, therefore it can formally be transformed into the original domain through a convolution integral. Taking beside (24) into account the result is (see also [5])

$$\frac{\partial\mathbf{\Phi}(\gamma,x)}{\partial\gamma} = \int_0^x \mathbf{\Phi}(\gamma,x-u)\frac{\partial\mathbf{M}(\gamma)}{\partial\gamma}\mathbf{\Phi}(\gamma,u)du \ , \tag{26}$$

where u stands for an integration variable. The derivative $\partial\mathbf{M}(\gamma)/\partial\gamma$ can be stated easily. The above integral could then be evaluated numerically, with an accuracy dependent on the method applied. In [5], however, an efficient technique is developed whose accuracy depends on evaluation of respective matrix exponential function $\mathbf{\Phi}(\gamma,x)$ and on inversion of a specially built-up matrix. Namely, denoting $\mathbf{M}(\gamma) = [m_{ij}(\gamma)]$, $i,j = 1,2,\ldots, N$, the last equation can be expressed as

$$\frac{\partial\mathbf{\Phi}(\gamma,x)}{\partial\gamma} = \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\partial m_{ij}(\gamma)}{\partial\gamma}\int_0^x \mathbf{\Phi}(\gamma,x-u)\mathbf{B}_{ij}\mathbf{\Phi}(\gamma,u)du \ , \tag{27}$$

with $\mathbf{B}_{ij}$ as a square matrix with the (i,j)-th element 1, while 0 elsewhere. Denoting the integral in (27) as $\mathbf{S}_{ij}(\gamma,x)$, it is valid

$$\mathbf{S}_{ij}(\gamma,x)=\left[\sum_{k=1}^{N}(-1)^{N-k}\,kR_{N-k}^{N}(\gamma)\mathbf{M}^{k-1}(\gamma)\right]^{-1}\times \tag{28}$$

$$\left[\left(\sum_{k=1}^{N}(-1)^{N-k}\frac{\partial R_{N-k+1}^{N}(\gamma)}{\partial m_{ij}(\gamma)}\mathbf{M}^{k-1}(\gamma)\right)x\mathbf{\Phi}(\gamma,x)-\sum_{u=1}^{N-1}\left(\sum_{k=u+1}^{N}(-1)^{N-k}(k-u)R_{N-k}^{N}(\gamma)\mathbf{M}^{k-u-1}(\gamma)\right)\mathbf{C}_{ij}(\gamma,x)\mathbf{M}^{u-1}(\gamma)\right],$$

where $\mathbf{C}_{ij}(\gamma,x) = \mathbf{B}_{ij}\mathbf{\Phi}(\gamma,x) - \mathbf{\Phi}(\gamma,x)\mathbf{B}_{ij}$ is a commutator, and $R_k^N(\gamma)$ coefficients and their derivatives w.r. to $m_{ij}(\gamma)$ can be found via processing the characteristic polynomial of $\mathbf{M}(\gamma)$ (Matlab function poly can be utilized), or as the sums of certain subdeterminants of $\mathbf{M}(\gamma)$, see [5] for details. The inverse matrix in (28) must exist to be able to apply the method. This condition is fulfilled if the eigenvalues of $\mathbf{M}(\gamma)$ are distinct [5]. The $\mathbf{\Phi}(\gamma,x)$ can be evaluated by Matlab built-in functions, either by expm directly or by eig through formula (19). A possible Matlab function dexpmc listing is shown below, its calling corresponds to the first method.

```
%*************************************************************************************************
function [F,dF]=dexpmc(M,dM)                                          % by Lubomír Brančík, 2008
F=expm(M);
N=length(M);
Rs=poly(M);
IS=zeros(N);
for k=1:N
    IS=IS+k*Rs(N-k+1)*M^(k-1);
end
SS=zeros(N);
for i=1:N
    for j=1:N
        B=zeros(N);
        B(i,j)=1;
        C=B*F-F*B;
        Su=zeros(N);
        for u=1:N-1
            Sk=zeros(N);
            for k=u+1:N
                Sk=Sk+(k-u)*Rs(N-k+1)*M^(k-u-1);
            end
            Su=Su+Sk*C*M^(u-1);
        end
        Sk=zeros(N);
        for k=1:N
            K=N-k+1;
            if i==j
                if K==1
                    dR=1;
                else
                    DC=nchoosek([1:i-1,i+1:N],K-1);
                    dR=0;
                    for r=1:size(DC,1)
                        dR=dR+det(M(DC(r,:),DC(r,:)));
                    end
                end
            else
                if K==1
                    dR=0;
                elseif K==2
                    dR=-M(j,i);
                else
                    DC=nchoosek([1:min(i,j)-1,min(i,j)+1:max(i,j)-1,max(i,j)+1:N],K-2);
                    dR=0;
                    for r=1:size(DC,1)
                        dR=dR-det(M([j,DC(r,:)],[i,DC(r,:)]));
                    end
                end
            end
            Sk=Sk+(-1)^(N-k)*dR*M^(k-1);
        end
        SS=SS+dM(i,j)*(Sk*F-Su);
    end
end
dF=IS\SS;
%*************************************************************************************************
```

7 Padé Approximation Method with Scaling & Squaring

As was pointed out earlier the Matlab built-in function expm uses a Padé approximation method in conjunction with scaling and squaring techniques [8]. A similar method can be applied to dedermine the matrix exponential function derivative. Let us consider the Padé approximation of the matrix exponential function (4) as

$$\mathbf{\Phi}(\gamma,x) \doteq \mathbf{D}_{pq}^{-1}(\gamma,x)\mathbf{N}_{pq}(\gamma,x) \ , \tag{29}$$

where

$$\mathbf{N}_{pq}(\gamma,x) = \sum_{k=0}^{p} \frac{(p+q-k)!\,p!}{(p+q)!k!(p-k)!}\left(\mathbf{M}(\gamma)x\right)^k \ , \quad \mathbf{D}_{pq}(\gamma,x) = \sum_{k=0}^{q} \frac{(p+q-k)!q!}{(p+q)!k!(q-k)!}\left(\text{-}\mathbf{M}(\gamma)x\right)^k \tag{30}$$

are polynomials of the matrix argument. Now we can follow (29) to get the derivative w.r. to γ as

$$\frac{\partial\mathbf{\Phi}(\gamma,x)}{\partial\gamma} \doteq \frac{\partial\mathbf{D}_{pq}^{-1}(\gamma,x)}{\partial\gamma}\mathbf{N}_{pq}(\gamma,x) + \mathbf{D}_{pq}^{-1}(\gamma,x)\frac{\partial\mathbf{N}_{pq}(\gamma,x)}{\partial\gamma} \ . \tag{31}$$

The substitution for $\partial\mathbf{D}_{pq}^{-1}(\gamma,x)/\partial\gamma$ and further arrangements lead to a formula

$$\frac{\partial\mathbf{\Phi}(\gamma,x)}{\partial\gamma} \doteq \mathbf{D}_{pq}^{-1}(\gamma,x)\left(\frac{\partial\mathbf{N}_{pq}(\gamma,x)}{\partial\gamma} - \frac{\partial\mathbf{D}_{pq}(\gamma,x)}{\partial\gamma}\mathbf{\Phi}(\gamma,x)\right), \tag{32}$$

where

$$\frac{\partial\mathbf{N}_{pq}(\gamma,x)}{\partial\gamma} = \sum_{k=0}^{p} \frac{(p+q-k)!\,p!}{(p+q)!k!(p-k)!}\frac{\partial\mathbf{M}^k(\gamma)}{\partial\gamma}x^k \ , \quad \frac{\partial\mathbf{D}_{pq}(\gamma,x)}{\partial\gamma} = \sum_{k=0}^{q} \frac{(p+q-k)!q!}{(p+q)!k!(q-k)!}\frac{\partial\mathbf{M}^k(\gamma)}{\partial\gamma}(-x)^k \tag{33}$$

result directly from (30) and $\partial\mathbf{M}^k(\gamma)/\partial\gamma$ can be stated by the recursive formula (7). To guarantee the sufficiently fast convergence the scaling and squaring are applied in according to (8) – (10). Here the matrix exponent part is scaled to get its Euklidian norm $\|\mathbf{M}(\gamma)x/r\| < 0.5$. Besides, it is convenient to choose $p = q$ in the sums (30) and (33) which leads to a good numerical stability and a simplification. In practice, $p = q = 6$ is usually sufficient choice, as e.g. the Matlab expm built-in function uses. The generalized Matlab function dexpm has been developed based on the above theory. Calling the function corresponds to the first method.

```
%*************************************************************************************************
function [F,dF]=dexpm(M,dM)                                          % by Lubomír Brančík, 2008
% Scale M by power of 2 so that its norm is < 1/2
[f,e]=log2(norm(M,'inf'));
r=max(0,e+1);
M=M/2^r; dM=dM/2^r;
% Pade approximation of exp(M) and diff[exp(M)]
X=M; Y=dM;
c=1/2;
F=eye(size(M))+c*M; dF=c*dM;
D=eye(size(M))-c*M; dD=-c*dM;
q=6;
p=1;
for k=2:q
   c=c*(q-k+1)/(k*(2*q-k+1));
   Y=dM*X+M*Y;
   X=M*X;
   cX=c*X; cY=c*Y;
   F=F+cX; dF=dF+cY;
   if p
     D=D+cX; dD=dD+cY;
   else
     D=D-cX; dD=dD-cY;
   end
   p=~p;
end
F=D\F;
dF=D\(dF-dD*F);
% Undo scaling by repeated squaring
for k=1:r
    dF=dF*F+F*dF;
    F=F*F;
end
%*************************************************************************************************
```

8 Concluding Comments

The paper has followed rather theoretical work [9] where the above stated methods have been collectively summarized for purposes of MTL systems simulation. Here, the techniques are stated more precisely, while describing them in more general notation. Moreover, possible Matlab listings of all these techniques programmed in the form of M-file functions are presented. All the methods give both matrix exponential function (**F**) and its first derivative (**dF**) as the results, while taking respective matrix (**M**) and its first derivative (**dM**) as the arguments. The matrices **M** can be complex in general, which is just the case $\mathbf{M} \equiv \mathbf{M}(s)$ needed in the MTL systems simulation. The only exception was the Matlab listing in the Laplace transform approach, but it can also be modified to enable it. One can see another generally usable LT approach e.g. in [10]. An error analysis based on the second-order chain matrices (2), with known analytical solution expressed through the hyperbolic functions, can be found in [9]. Table 2 shows the mean values of relative errors obtained.

TABLE 2. MEAN VALUES OF RELATIVE ERRORS

Taylor series expansion	Augmented matrix utilization	Eigenvalues decomposition
$\approx 10^{-13}$	$\approx 10^{-13}$	$\approx 10^{-15}$
Laplace transform approach	Convolution integral evaluation	Padé approximation method
$\approx 10^{-12}$	$\approx 10^{-15}$	$\approx 10^{-13}$

References

[1] L. Brančík. Novel techniques for sensitivity evaluation in multiconductor transmission line systems, *WSEAS Transactions on Communications*, 4 (2005), No. 5, 216–223.

[2] L. Brančík. Voltage/current waves sensitivity in hybrid circuits with nonuniform MTLs, in *Proceedings of the 10th IEEE Workshop on SPI'06*, Berlin (Germany), 2006, 177-180.

[3] L. Brančík. Techniques of matrix exponential function derivative for electrical engineering simulations, in *Proceedings of IEEE International Conference on Industrial Technology*. Mumbai (India): Indian Institute of Technology Bombay, 2006. p. 2608 - 2613.

[4] T. C. Fung. Computation of the matrix exponential and its derivatives by scaling and squaring, *International Journal for Numerical Methods in Engineering*, 59 (2004), 1273–1286.

[5] H. Tsai, K. S. Chan. A note on parameter differentiation of matrix exponentials, with applications to continuous-time modeling, *Technical report #311*, Univ. of Iowa, 2001, 1–22.

[6] L. Brančík. Chain matrix derivative via Laplace transform method with applications to distributed circuits simulation, in *Proceedings of IMAPS CS International Conference EDS'06*, Brno (Czech Republic), 2006, 58 – 63.

[7] J. Valsa, L. Brančík. Approximate formulae for numerical inversion of Laplace transforms. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*. 1998, vol. 11, no. 3, p. 153 - 166.

[8] C. Moler , C. V. Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, *SIAM Review*, (45) 2003, No. 1, 1–46.

[9] L. Brančík. Procedures for matrix exponential function derivative in Matlab. *Przeglad Elektrotechniczny - Konferencje*. 2007, vol. 5, no. 2, p. 7 - 10.

[10] L. Brančík. Modified technique of FFT-based numerical inversion of Laplace transforms with applications. *Przeglad Elektrotechniczny*. 2007. vol. 83, no. 11, p. 53 - 56.

Acknowledgments

This work was supported by the Czech Ministry of Education under the MSM 0021630503 Research Project MIKROSYN.

Doc. Ing. Lubomír Brančík, CSc.
Department of Radio Electronics, FEEC BUT in Brno,
Purkyňova 118, 612 00 Brno, Czech Republic,
E-mail: brancik@feec.vutbr.cz, Phone: +420541149128, Fax: +420541149244.