МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2016
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2016
Proceedings of the Forty Fifth Spring Conference
of the Union of Bulgarian Mathematicians
Pleven, April 6–10, 2016

MATLAB SIMULATION OF HIGH ORDER MODELS IN STRUCTURE ENGINEERING*

Mihail Konstantinov, Galina Pelova, Petko Petkov, Vesela Pasheva

In this tutorial paper we deal with certain high order mathematical models in structure engineering. We propose a MATLAB[a] based approach to simulate these models using in-built matrix functions such as the the program **expm** for reliable computation of the matrix exponential. The computation of the matrix sine and cosine functions is also considered together with their applications. This approach has been used by the authors in teaching mathematics and mechanics at the University of Architecture, Civil Engineering and Geodesy (Sofia, Bulgaria), Technical University of Sofia and the European Polytechnical University (Pernik, Bulgaria).

Introduction. Many processes in structure engineering are governed by linear high-order time-invariant differential equations with given initial or boundary conditions. The classical approach to deal with initial or boundary value problems may not be applicable in this case. The direct implementation of integration schemes is also connected with numerical difficulties since they do not take into account the special structure of the problem. At the same time the formal solution is described by a matrix exponential function, integrals involving this function and a set of unknown parameters that are subject to further calculation. The computer computation of matrix functions (such as sine, cosine and exponent) may be a problem. Recently effective and reliable algorithms as well as computer codes for this purpose have been derived. An example is the program **expm** from MATLAB for computing the matrix exponential. This program may successfully be used for tutorial purposes when solving problems in mathematics and mechanics as well as for simulation of real models arising in structure engineering. Some aspects of this approach are described in the present paper.

Dynamical model. We consider high-order time-invariant dynamical systems governing elastic and viscoelastic mechanical structures of the type [2]

$$\mathcal{L}(\mathbf{y})(t) := \mathbf{y}''(t) + \mathbf{A}\mathbf{y}'(t) + \mathbf{B}\mathbf{y}(t) = \mathbf{f}(t) \tag{1}$$

where

$$\mathbf{y}(t) = [y_1(t); y_2(t); \ldots; y_n(t)] \in \mathbb{R}^n$$

is the state vector of the system at the moment t and $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ are given matrices. The function $\mathbf{f}\colon \mathbb{R} \to \mathbb{R}^n$ is continuous (belongs to the set $C(\mathbb{R}, \mathbb{R}^n)$). By $\mathcal{L}\colon C^2(\mathbb{R}, \mathbb{R}^{n\times p}) \to$

*2010 Mathematics Subject Classification: Primary 97M50.
Key words: MATLAB based modeling, structure engineering, matrix functions.
[a] MATLAB® is a trademark of MathWorks, Inc.

$C(\mathbb{R}, \mathbb{R}^{n\times p})$ we denote a differential operator acting on $(n\times p)$–matrix functions. In many applications $\mathbf{A} = \mathbf{0}$ and $\mathbf{B}$ is a symmetric positively definite matrix. The matrices $\mathbf{A}, \mathbf{B}$ may also depend on a vector-parameter $\lambda \in \mathbb{R}^m$, namely $\mathbf{A} = \mathbf{A}(\lambda)$, $\mathbf{B} = \mathbf{B}(\lambda)$.

The vector differential equation (1) is considered together with a set of initial

$$\mathbf{y}(0) = \mathbf{y}_0,\ \mathbf{y}'(0) = \mathbf{y}'_0 \tag{2}$$

or boundary

$$\mathbf{y}(0) = \mathbf{y}_0,\ \mathbf{y}(l) = \mathbf{y}_l \tag{3}$$

conditions, where $\mathbf{y}_0, \mathbf{y}'_0, \mathbf{y}_l \in \mathbb{R}^n$ are given vectors. The more general multi-point boundary condition

$$\sum_{k=0}^{r} (\mathbf{C}_k \mathbf{y}(t_k) + \mathbf{D}_k \mathbf{y}'(t_k)) = \mathbf{b},\ 0 = t_0 < t_1 < \cdots < t_r = l \tag{4}$$

may also be considered, where $\mathbf{C}_k, \mathbf{D}_k \in \mathbb{R}^{2n\times n}$ and $\mathbf{b} \in \mathbb{R}^{2n}$ are given arrays. The vector and matrix data in (2)–(4) may also depend on the parameter λ.

As a first task the students have to show that the initial conditions (2) and the boundary conditions (3) are particular cases of the general boundary condition (4). This is done as follows. Setting $r = 0$ and

$$\mathbf{C}_0 = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix},\ \mathbf{D}_0 = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix},\ \mathbf{b} = \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}'_0 \end{bmatrix}$$

we obtain (2). For $r = 1$ and

$$\mathbf{C}_0 = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix},\ \mathbf{C}_1 = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix},\ \mathbf{D}_0 = \mathbf{D}_1 = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix},\ \mathbf{b} = \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_l \end{bmatrix}$$

the conditions (4) take the form (3).

A variant of the above problem arises when the system is homogeneous ($\mathbf{f} = \mathbf{0}$, $\mathbf{b} = \mathbf{0}$) and we are interested in values of λ (i.e. *generalized eigenvalues*) for which a nontrivial solution of (1), (4) exists. This is the so called *generalized eigenvalue problem*, see e.g. [4]. An example of a particular system solved in MATLAB environment is given in [5].

Further on the students are advised to write their own programs in MATLAB environment following the next presentation.

Formal solution. It is instructive to recall that the general solution of equation (1) has the form

$$\mathbf{y}(t) = \mathbf{Y}_1(t)\mathbf{c}_1 + \mathbf{Y}_2(t)\mathbf{c}_2 + \mathbf{y}_f(t)$$

where the matrix functions $\mathbf{Y}_k$ satisfy the homogeneous equation

$$\mathbf{Y}''(t) + \mathbf{A}\mathbf{Y}'(t) + \mathbf{B}\mathbf{Y}(t) = \mathbf{0}$$

and $\mathbf{y}_f$ is a particular solution corresponding to the right-hand side $\mathbf{f}$, namely

$$\mathbf{y}''_f(t) + \mathbf{A}\mathbf{y}'_f(t) + \mathbf{B}\mathbf{y}_f(t) = \mathbf{f}(t)$$

The particular solution can be computed explicitly for various classes of functions like

$$\mathbf{f}(t) = \sum_{k=0}^{m} f_k(t)\mathbf{a}_k$$

where the scalar functions f_k contain powers, exponents, sines, cosines and other elementary functions of t, and $\mathbf{a}_k$ are given vectors. In such cases the particular solution is of

the form of the function $\mathbf{f}$ with unknown vector coefficients which can be found by the method of undetermined coefficients. This is done in a close cooperation with the students (or, at least, with their active part).

Note that the use of this method is possible only under some restrictions on the matrices $\mathbf{A}$, $\mathbf{B}$ and the function $\mathbf{f}$ and this requires an additional analysis. Let for example $\mathbf{f}(t) = \mathbf{a}_0 + t\mathbf{a}_1$ and the matrix $\mathbf{B}$ be invertible. Then $\mathbf{y}_f(t) = \mathbf{b}_0 + t\mathbf{b}_1$, where

$$\mathbf{b}_1 = \mathbf{B}\backslash\mathbf{a}_1,\ \mathbf{b}_0 = \mathbf{B}\backslash(\mathbf{a}_0 - \mathbf{A} * \mathbf{b}_1)$$

The notation $\mathbf{b}_1 = \mathbf{B}\backslash\mathbf{a}_1$ uses the MATLAB operator for left matrix division and means that the vector $\mathbf{b}_1$ satisfies equation $\mathbf{B} * \mathbf{b}_1 = \mathbf{a}_1$ and is computed by the corresponding MATLAB algorithm (Gauss elimination with partial pivoting) thus avoiding matrix inversion and matrix–vector multiplication.

Nonhomogeneous case. To solve (1), (4) in this case we set

$$\mathbf{z}(t) = \begin{bmatrix} \mathbf{y}(t) \\ \mathbf{y}'(t) \end{bmatrix} \in \mathbb{R}^{2n},\ \mathbf{M} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{B} & -\mathbf{A} \end{bmatrix} \in \mathbb{R}^{2n\times 2n},\ \mathbf{g}(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}(t) \end{bmatrix} \in \mathbb{R}^{2n}$$

and $\mathbf{E}_k = [\mathbf{C}_k, \mathbf{D}_k] \in \mathbb{R}^{2n\times 2n}$. The problem is then written as

$$\mathbf{z}'(t) = \mathbf{M}\mathbf{z}(t) + \mathbf{g}(t),\ \sum_{k=0}^{r} \mathbf{E}_k\mathbf{z}(t_k) = \mathbf{b}$$

Using the matrix exponential

$$\exp(\mathbf{X}) = \sum_{k=0}^{\infty} \frac{\mathbf{X}^k}{k!},\ \mathbf{X} \in \mathbb{R}^{n\times n}$$

the solution is obtained in the form

$$\mathbf{z}(t) = \exp(t\mathbf{M})\mathbf{z}_0 + \mathbf{u}(t),\ \mathbf{u}(t) = \int_0^t \exp((t-s)\mathbf{M})\mathbf{g}(s)\mathrm{d}s \tag{5}$$

where $\mathbf{z}_0 \in \mathbb{R}^{2n}$ is the solution of the linear algebraic vector equation

$$\mathbf{F}\mathbf{z}_0 = \mathbf{c},\ \mathbf{F} = \sum_{k=0}^{r} \mathbf{E}_k \exp(t_k\mathbf{M}),\ \mathbf{c} = \mathbf{b} - \sum_{k=0}^{r} \mathbf{E}_k\mathbf{u}(t_k) \tag{6}$$

In the generic case the matrix $\mathbf{F}$ is invertible and the solution for $\mathbf{z}_0$ is unique. Finally the state vector is restored as $\mathbf{y}(t) = [\mathbf{I}, \mathbf{0}]\mathbf{z}(t)$.

The computation of the solution from (5) and (6) in MATLAB is done using the command **expm** for the matrix exponential and the program **quadl** for numerical integration. The solution of the vector algebraic equation in (6) is obtained by the MATLAB operator for left matrix division as `» z0 = F\c`.

Property of fundamental matrix. If we partition the fundamental matrix as

$$\exp(t\mathbf{M}) = \begin{bmatrix} \Phi_{11}(t) & \Phi_{12}(t) \\ \Phi_{21}(t) & \Phi_{22}(t) \end{bmatrix},\ \Phi_{pq}(t) \in \mathbb{R}^{n\times n} \tag{7}$$

where $\Phi_{11}(0) = \Phi_{22}(0) = \mathbf{I}$ and $\Phi_{12}(0) = \Phi_{21}(0) = \mathbf{0}$, then

$$\mathcal{L}(\Phi_{pq}) = \mathbf{0} \text{ for all } p, q = 1, 2$$

This important property of the blocks of partitioned fundamental matrices of autonomous equations is not very popular and the students are advised to check it theoretically working 'by hand' and computationally using the MATLAB function **expm** (for $p = 1$

and $q = 1, 2$ the property is trivial and is valid for non-autonomous equations as well).

Using the partition (7) the solution of the initial value problem (1), (2) is represented as

$$\mathbf{y}(t) = \Phi_{11}(t)\mathbf{y}_0 + \Phi_{12}(t)\mathbf{y}'_0 \tag{8}$$

In turn, the solution of the boundary value problem (1), (3) is again in the form (8), where the vector $\mathbf{y}'_0$ is now not given but is the solution (if it ever exists) of the linear algebraic equation

$$\Phi_{12}(l)\mathbf{y}'_0 = \mathbf{y}_l - \Phi_{11}(l)\mathbf{y}_0$$

A generalization of the above facts for vector differential equations of higher order such as

$$\mathbf{y}'''(t) + \mathbf{A}\mathbf{y}''(t) + \mathbf{B}\mathbf{y}'(t) + \mathbf{C}\mathbf{y}(t) = \mathbf{0}$$

etc. is straightforward and is proposed as an exercise to the students.

Homogeneous case. Here $\mathbf{f} = \mathbf{0}$ and $\mathbf{b} = \mathbf{0}$. The problem (1), (4) has trivial solution $\mathbf{z} = \mathbf{0}$. When the data matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}_k$ and $\mathbf{D}_k$ depend on the vector-parameter λ, we have $\mathbf{F} = \mathbf{F}(\lambda)$ and the problem is to find the *eigenvalues* λ. The boundary value problem has a nontrivial solution

$$\mathbf{z}(t) = \exp(t\mathbf{M})\mathbf{z}_0,\ \mathbf{z}_0 \neq \mathbf{0}$$

when the matrix $\mathbf{F}(\lambda)$ is singular. In this case $\mathbf{z}_0$ is any nonzero vector from the kernel of the matrix $\mathbf{F}(\lambda)$. Although computing determinants is not a pleasant operation in finite arithmetics, we can find the values for λ by the equation

$$f(\lambda) = 0,\ f(\lambda) = \det(\mathbf{F}(\lambda))$$

This equation defines a $(m-1)$-dimensional variety in the space $\mathbb{R}^m$ of the parameter λ.

The cases $m = 1$, $m = 2$ and $m = 3$ are especially instructive. For $m = 1$ the argument λ of the function f is scalar and we may construct the graph of this function numerically. The zeros of f are found by the MATLAB commands **solve**, or **fzero**.

In general the equation $f(\lambda) = 0$ may have more than one root. However, in this case the command **solve** most probably will produce only one root. To find all roots in the interesting interval for λ we plot the graph of the function $\lambda \mapsto |f(\lambda)|$. The inverted peaks of the graph correspond to the zeros of equation $f(\lambda) = 0$. Having information about the approximate values of all roots in the given interval we can implement the program **fzero** for any one of them (this program requires an initial approximation to the root). Of course, theoretically the number of eigenvalues may be infinite.

For $m = 2$ we have a curve in the plane (λ_1, λ_2) which may be constructed by the command **ezplot** from MATLAB. For $m = 3$ we have a surface in the space $(\lambda_1, \lambda_2, \lambda_3)$ which is constructed by the MATLAB program **ezsurf**.

Classical approach example. Let $n = 2$, $m = 1$ and consider the equation

$$\mathbf{y}''(t) + \mathbf{A}(\lambda)\mathbf{y}'(t) + \mathbf{B}(\lambda)\mathbf{y}(t) = \mathbf{0} \tag{9}$$

with boundary conditions

$$\mathbf{y}(0) = \mathbf{0},\ \mathbf{y}(l) = \mathbf{0} \tag{10}$$

The problem of existence of nontrivial solutions may be reduced to a scalar 4-th order differential equation with zero homogeneous boundary conditions. There is an analytical

solution recommended in classical textbooks on Theory of Elasticity. It is based on the type of the characteristic 4-th degree algebraic characteristic equation of the form

$$s^4 + a_1(\lambda)s^3 + a_2(\lambda)s^2 + a_3(\lambda)s + a_4(\lambda) = 0$$

with four roots $s_j = s_j(\lambda)$, $j = 1, 2, 3, 4$. There are 8 (!) different subcases for the roots and respectively 8 forms for the solution. In each subcase we have to solve a transcendental equation for the scalar λ. The complete solution of this problem requires a very large amount of hand calculations which, as a rule, are contaminated with errors and the thus result obtained is useless.

Even more involved is the case of two scalar parameters, i.e. $\lambda = [\lambda_1; \lambda_2] \in \mathbb{R}^2$. Here we have 8 classes of transcendental curves $g_k(\lambda_1, \lambda_2) = 0$, $k = 1, 2, \ldots, 8$. Until recently the solution of such a problem was often the main part of a PhD thesis in Structure Engineering.

It is clear that such an *antediluvian approach* is not to be used in a modern teaching course in Structure Engineering. Moreover, it is not applicable at all for $n \geq 3$ since the characteristic equation here is of degree ≥ 6 when no closed form solution for the roots ever exists.

The use of matrix exponential approach allows to solve the problem easily not only for $n = 2$ but also for large values of n.

Scaling and Squaring Algorithm (SSA). The SSA for computing the matrix exponential is well studied [1, 3, 6] and implemented as the MATLAB function **expm**. Suppose that the computation of the matrix function $\mathbf{F}\colon \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ involves the evaluation of matrix powers $\mathbf{M}, \mathbf{M}^2, \ldots, \mathbf{M}^k$ in floating-point binary arithmetics. If $\mathbf{M}$ has large elements this may lead to large errors in the computed result.

In many cases, however, this may be avoided by preliminary scaling the matrix argument $\mathbf{M}$, e.g. $\mathbf{M}_1 = \mathbf{M}/s$, where the scaling factor $s = 2^p$, $p \in \mathbb{N}$, is a power of the base 2 of the arithmetic. Usually we look for p such that $\|\mathbf{M}_1\| \leq 1/2$, i.e. p is the smallest integer such that $p \geq 1 + \log_2(\|\mathbf{M}\|)$. Thus the computation of $\mathbf{F}(\mathbf{M}_1)$ is reliable but restoring the value of $\mathbf{F}(\mathbf{M})$ from $\mathbf{F}(\mathbf{M}_1)$ may be a problem. For the matrix exponential, however, the restoration is straightforward due to the property $\exp(y) = (\exp(y/s))^s$ of the exponential function. In particular we

$$\exp(\mathbf{M}) = (\exp(\mathbf{M}/s))^s \simeq (P(\mathbf{M}/s))^s\,,\ s = 2^p$$

where $P(x)$ is a symmetric Padé approximation of $\exp(x)$,

$$\begin{aligned} P(x) &= \frac{p(x)}{p(-x)},\ p(x) = 1 + \sum_{j=1}^{k} a_j x^j \\ a_1 &= \frac{1}{2},\ a_2 = \frac{1}{9},\ a_3 = \frac{1}{72},\ a_4 = \frac{1}{1008},\ a_5 = \frac{1}{30240}, \ldots \end{aligned}$$

or a truncation of the Taylor series of $\exp(x)$,

$$P(x) \simeq \sum_{k=0}^{N} \frac{x^k}{k!}$$

Under certain conditions Padé and Taylor formulas have similar accuracy.

Computing matrix sine and cosine functions. The SSA cannot be applied directly to the computation of the matrix sine and cosine functions either by Padé or Taylor approximations, e.g.

$$\sin(\Omega) \simeq \sum_{k=0}^{N} \frac{(-1)^k \Omega^{2k+1}}{(2k+1)!},\ \cos(\Omega) \simeq \sum_{k=0}^{N} \frac{(-1)^k \Omega^{2k}}{(2k)!},\ \Omega \in \mathbb{R}^{n\times n}$$

In many applications the matrix $\Omega = \Omega^\top$ is nonnegatively definite. Together with the matrix $\cos(t\Omega)$ which is always defined, we also need the matrix

$$\mathbf{S}(t,\Omega) = \Omega^{-1}\sin(t\Omega)$$

if Ω is invertible, and

$$\mathbf{S}(t,\Omega) = \sum_{k=0}^{\infty} \frac{(-1)^k t^{2k+1} \Omega^{2k}}{(2k+1)!}$$

in the general case.

The importance of these matrices in Theory of Elasticity of mechanical systems comes from the wide spread model

$$\mathbf{y}''(t) + \Omega^2 \mathbf{y}(t) = \mathbf{0};\ \mathbf{y}(0) = \mathbf{y}_0,\ \mathbf{y}'(0) = \mathbf{y}_0 \tag{11}$$

Here the solution is

$$\mathbf{y}(t) = \cos(t\Omega)\mathbf{y}_0 + \mathbf{S}(t,\Omega)\mathbf{y}_0' \tag{12}$$

or, if the matrix Ω is invertible,

$$\mathbf{y}(t) = \cos(t\Omega)\mathbf{y}_0 + \Omega^{-1}\sin(t\Omega)\mathbf{y}_0'$$

Consider next the homogeneous boundary value problem

$$\mathbf{y}''(t) + \Omega^2 \mathbf{y}(t) = \mathbf{0};\ \mathbf{y}(0) = \mathbf{y}(l) = \mathbf{0} \tag{13}$$

which has trivial solution $\mathbf{y} = \mathbf{0}$. The general solution of this problem is

$$\mathbf{y}(t) = \mathbf{S}(t,\Omega)\mathbf{a}$$

where $\mathbf{S}(l,\Omega)\mathbf{a} = \mathbf{0}$, i.e. the vector $\mathbf{a} \in \mathbb{R}^n$ is from the kernel of the matrix $\mathbf{S}(l,\Omega)$ which has dimension $n - \mathrm{rank}(\mathbf{S}(l,\Omega))$.

If $\Omega = \Omega(\lambda)$, where $\lambda \in \mathbb{R}^m$ is a vector-parameter then the *spectrum* of the problem is an $(m-1)$-dimensional variety in $\mathbb{R}^m$ defined by

$$\Lambda = \{\lambda \in \mathbb{R}^m :\ \det(\mathbf{S}(l,\Omega(\lambda))) = 0\}$$

When $\det(\Omega(\lambda)) \neq 0$ the equation for Λ is reduced to $\det(\sin(l\Omega(\lambda))) = 0$.

An effective way to compute the sine and cosine matrix functions for $(n \times n)$ matrices is via the matrix exponential of a $(2n \times 2n)$ matrix exponential function. Indeed, the initial value problem (11) may be written as

$$\mathbf{z}'(t) = \mathbf{N}\mathbf{z}(t),\ \mathbf{z}(t) = \begin{bmatrix} \mathbf{y}(t) \\ \mathbf{y}'(t) \end{bmatrix},\ \mathbf{N} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\Omega^2 & \mathbf{0} \end{bmatrix} \tag{14}$$

The fundamental matrix for (14) is

$$\exp(t\mathbf{N}) = \begin{bmatrix} \Psi_{11}(t) & \Psi_{12}(t) \\ \Psi_{21}(t) & \Psi_{22}(t) \end{bmatrix} = \begin{bmatrix} \cos(t\Omega) & \mathbf{S}(t,\Omega) \\ -\Omega\sin(t\Omega) & \cos(t\Omega) \end{bmatrix}$$

or, if the matrix Ω is invertible,

$$\exp(t\mathbf{N}) = \begin{bmatrix} \cos(t\Omega) & \Omega^{-1}\sin(t\Omega) \\ -\Omega\sin(t\Omega) & \cos(t\Omega) \end{bmatrix}$$

Thus *the interesting matrices* $\cos(t\Omega)$ *and* $\mathbf{S}(t,\Omega)$ *are simply the* $n \times n$ *blocks of the matrix exponential* $\exp(t\mathbf{N})$ *in positions* $(1,1)$ *and* $(1,2)$*, respectively.*

Note that the case $\Omega = \mathbf{0}$ simply leads to $\mathbf{N} = [\mathbf{0}, \mathbf{I}; \mathbf{0}, \mathbf{0}]$ and

$$\exp(t\mathbf{N}) = \begin{bmatrix} \mathbf{I} & t\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

A further development is as follows. Consider the equation

$$\mathbf{y}''(t) + \mathbf{B}\mathbf{y}(t) = \mathbf{0}$$

where the matrix $\mathbf{B}$ may not be symmetric and/or nonnegatively definite. If the matrix $\mathbf{B}$ has a square root Ω (i.e. there is a matrix Ω such that $\Omega^2 = \mathbf{B}$), then we can still apply the above approach obtaining the solution in the form (12). An interesting simple case is $n = 2$ and

$$\mathbf{B} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},\ \Omega = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$$

Here the principal root $\Omega = \mathbf{B}^{1/2}$ may be calculated easily "by hand" as well as by the MATLAB command **sqrtm**(B) for computing the matrix square root. Obviously the matrix $-\Omega$ is also a square root of $\mathbf{B}$.

If the matrix $\mathbf{B}$ has no square root (this is possible when $\mathbf{B}$ is singular) we cannot use the sine and cosine matrix functions immediately. Of course, the matrix exponential for the augmented system may again be used to compute the solution.

Examples with reference solution. A very effective teaching technique is to consider examples with known solutions. Our first example is $n = 2$ and $\mathbf{B} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ (the students must show that $\mathbf{B}$ has no square root). The students are asked to find the solution

$$y_1(t) = y_{10} + ty'_{10} - \frac{t^2}{2}y_{20} - \frac{t^3}{6}y'_{20},\ y_2(t) = y_{20} + ty'_{20}$$

of the corresponding initial value problem both analytically "by hand" and numerically by MATLAB (in the latter case the solution is obtained for a set of values for t).

The second example is the scalar equation $y''+\lambda y = 0$, $\lambda > 0$ with boundary conditions $y(0) = 0$, $y(l) = 0$. The closed form nontrivial solutions are $y(t) = a\sin(t\sqrt{\lambda})$, where $a \neq 0$ is arbitrary and $\lambda = \lambda_k = k^2\pi^2/l^2$, $k \in \mathbb{N}$. In addition to these analytical solutions the students are asked to find numerically the first several eigenvalues $\lambda_1, \lambda_2, \ldots$ using the matrix exponential $\exp(l\mathbf{M})$ for $\mathbf{M} = \begin{bmatrix} 0 & 1 \\ -\lambda & 0 \end{bmatrix}$, the graph of the function $\lambda \mapsto |f(\lambda)|$ described above and the command **fzero**.

Graphical implementation. In all cases it is very instructive to construct 2D and 3D projections of phase trajectories of the corresponding dynamical system using the graphical interface of MATLAB. This is done by the commands **plot** and **ezplot** for 2D graphs as well as **plot3** and **ezplot3** for 3D graphs.

Conclusion. In this tutorial paper we have considered the implementation of the matrix exponential function and the corresponding computer code **expm** from MATLAB for simulation of linear high-order time-invariant models in structure engineering. The approach proposed is much more effective than the classical approach based on the closed-form solution of low-order models. The results may also be used in the disciplines "Mathematical Analysis II" and "Applied Mathematics" taught in the technical universities.

REFERENCES

[1] A. Al-Mohy, N. Higham. A scaling and squaring algorithm for the matrix exponential. *SIAM J. Matrix Anal. Appl.*, **31** (2009), 970–989.

[2] E. Gaylord, Ch. Gaylord, J. Stallmeyer. Structural Engineering Handbook. McGraw-Hill, N.Y., 1997.

[3] N. Higham. The scaling and squaring method for the matrix exponential revisted. *SIAM J. Matrix Anal. Appl.*, **26** (2005), 1179–1193.

[4] M. Konstantinov, G. Pelova. Autonomous generalized Sturm-Liuville problems: Numerical solution by MATLAB. Proc. First Int. Conf. "Education, Innovations, Science", Pernik 2011, 147–150.

[5] D. Lolov. Application of the matrix exponential for analyzing stability problems of mechanical systems with MATLAB (in Bulgarian) *Annual Univ. Arch. Civil Eng. Geodesy* **43, 44** (2004–2009), fasc. II, 189–196.

[6] P. Petkov, N. Christov, M. Konstantinov. Computational Methods for Linear Control Systems. Prentice Hall, NJ, 1991.

Mihail Konstantinov
e-mail: mmk_fte@uacg.bg
Galina Pelova
e-mail: gpelova@abv.bg
University of Architecture,
Civil Engineering and Geodesy
1, Hr. Smirnenski Blvd
1046 Sofia, Bulgaria

Petko Petkov
e-mail: php@tu-sofia.bg
Vesela Pasheva
e-mail: vvp@tu-sofia.bg
Technical University of Sofia
8, Kl. Ohridski Blvd
1756 Sofia, Bulgaria

СИМУЛАЦИЯ В СРЕДА НА MATLAB НА МОДЕЛИ ОТ ВИСОК РЕД В СТРОИТЕЛНОТО ИНЖЕНЕРСТВО

Михаил Михайлов Константинов, Галина Божилова Пелова, Петко Христов Петков, Весела Василева Пашева

В тази образователна работа са разгледани някои математически модели от висок ред в строителното инженерство. Предложен е подход, основан на някои вградени функции в MATLAB, като например **expm** за надеждно пресмятане на матричната експонента. Разгледано е също пресмятането на функциите матричен синус и матричен косинус, както и някои техни приложения. Описаният подход е използван от авторите при преподаване на математика и механика в Университета по архитектура, строителство и геодезия в София, Техническия университет в София и Европейския политехнически университет в Перник.