An Operational Matrix-Based Algorithm for Simulating Linear and Fractional Differential Circuits

Yuanzhe Wang, Haotian Liu, Grantham K. H. Pang and Ngai Wong
Department of Electrical and Electronic Engineering
The University of Hong Kong, Pokfulam Road, Hong Kong
Email: {yzwang,htliu,gpang,nwong}@eee.hku.hk

***Abstract*—We present a new time-domain simulation algorithm (named OPM) based on operational matrices, which naturally handles system models cast in ordinary differential equations (ODEs), differential algebraic equations (DAEs), high-order differential equations and fractional differential equations (FDEs). When applied to simulating linear systems (represented by ODEs or DAEs), OPM has similar performance to advanced transient analysis methods such as trapezoidal or Gear's method in terms of complexity and accuracy. On the other hand, OPM naturally handles FDEs without much extra effort, which can not be efficiently solved using existing time-domain methods. High-order differential systems, being special cases of FDEs, can also be simulated using OPM. Moreover, adaptive time step can be utilized in OPM to provide a more flexible simulation with low CPU time. Numerical results then validate OPM's wide applicability and superiority.**

I. Introduction

Operational matrix approach, though not new in the literature, seems to have low awareness in the electronic design automation community [1]–[6]. In this paper, a new time-domain simulation method based on operational matrix (OPM) is proposed, which has a broad application scope. First, OPM handles linear systems depicted by ordinary differential equations (ODEs) and differential algebraic equations (DAEs). For instance, when the operational matrix is constructed using block-pulse functions (BPFs) as basis (which is the case in this paper for illustrative purpose), the OPM method for solving ODEs and DAEs features comparable complexity and accuracy to transient analysis schemes like trapezoidal or Gear's methods. Nonetheless, besides BPFs, there exist various other basis functions, such as the Walsh functions, the Laguerre functions, the Legendre functions, the Haar functions, etc. And OPM can readily switch to using other basis functions, each having its own merits. For example, the Walsh functions are a set of low- to high-frequency basis functions. Thereby if we are only interested in the overall trend of the response waveforms and do not care the details in a local time interval, Walsh function is a better choice.

On the other hand, OPM handles many special systems where traditional transient analysis methods fail, such as systems described by fractional differential equations (FDEs). Fractional differential systems arise from many applications areas, such as controller design and transmission line analysis [7], [8]. There exist fractional derivative operators in the format of $\frac{d^\alpha}{dt^\alpha}$ in the FDEs, where α is not a positive integer. For simulating such systems, the traditional transient analysis methods are extremely inefficient if not impossible. Previously, such fractional differential systems are usually simulated in the frequency domain using Fourier transform and inverse Fourier transform. However, it is difficult to control the approximation error. Besides, the CPU time of the frequency domain methods is high because they involve complex number arithmetic. In this paper, the OPM method for simulating the fractional differential systems is proposed, which is both compact in theory and efficient in practice. The fractional differential operational matrix is first constructed. Then the system can be simulated in exactly the same manner as ODEs and DAEs.

The main contributions of this paper are

(i) An OPM method for linear system simulation is proposed. It has roughly the same performance as advanced transient analysis methods (such as trapezoidal and Gear's methods) in terms of complexity and accuracy.
(ii) The very same OPM method can simulate fractional differential systems that can not be handled efficiently using traditional transient analysis.
(iii) The OPM method can naturally handle high-order differential systems, which are special cases of fractional differential systems.
(iv) Adaptive time step can be used in the scheme of OPM to provide a more flexible simulation with lower runtime

The remainder of the paper is organized as follows. Background is introduced in Section II. OPM for linear system simulation is proposed in Section III. In Section IV, the OPM method is extended to simulation of fractional and high-order differential systems. Numerical results are given in Section V. Section VI draws the conclusion and provides directions of

DATE'12, March 12, 2012, Dresden, Germany.

future work.

II. Background

A. Block-Pulse Functions

For a given time span $[0,\ T)$ and a time interval $h = T/m$, BPFs are defined as [4]

$$\phi_i(t) = \begin{cases} 1, & ih \le t < (i+1)h \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where m is an integer and $i = 0, \ldots, m-1$. Using the BPFs as basis, any (scalar) function $f(t)$ that is integrable in $[0,\ T)$ can be expanded as

$$f(t) \approx \sum_{i=0}^{m-1} f_i \phi_i(t) = f_{(m)}^T \phi_{(m)}(t), \tag{2}$$

where

$$\begin{aligned} f_{(m)} &= [f_0, f_1, \ldots, f_{m-1}]^T, \\ \phi_{(m)}(t) &= [\phi_0(t), \phi_1(t), \ldots, \phi_{m-1}(t)]^T, \\ f_i &= \frac{1}{h}\int_{ih}^{(i+1)h} f(t)dt. \end{aligned}$$

Equation (2) can be interpreted as a "discretization" of a continuous function $f(t)$. Roughly, $f_i = f(ih)$. From another perspective, if the coefficient vector $f_{(m)}$ is known, $f(t)$ can be constructed as $f(t) = f_{(m)}^T \phi_{(m)}(t)$.

B. Integral Operator

The integration of BPFs can be calculated using integral operator [4].

$$\int_0^t \phi_{(m)}(\tau)d\tau = H_{(m)}\phi_{(m)}(t), \quad t \in [0, T), \tag{3}$$

where

$$H_{(m)} = \begin{bmatrix} \frac{h}{2} & h & \cdots & h \\ & \frac{h}{2} & \cdots & h \\ & & \ddots & \\ & & & \frac{h}{2} \end{bmatrix}. \tag{4}$$

It is readily verified that

$$\begin{aligned} H_{(m)} &= h\left(\frac{1}{2}I + Q_m + \cdots + Q_m^{m-1}\right) \\ &= \frac{h}{2}\left(I + Q_m\right)\left(I - Q_m\right)^{-1} \end{aligned} \tag{5}$$

with

$$Q_m = \begin{bmatrix} 0_{(m-1)\times 1} & I_{m-1} \\ 0 & 0_{1\times(m-1)} \end{bmatrix} \tag{6}$$

being an index-m nilpotent matrix. The inverse of (4) is the differential operator

$$D_{(m)} = \frac{2}{h}\left(I - Q_m\right)\left(I + Q_m\right)^{-1}. \tag{7}$$

The derivative of a function $f(t)$ in its BPF representation ($f(t) = f_{(m)}^T \phi_{(m)}(t)$) can now be computed by

$$\frac{d}{dt}f(t) = \frac{d}{dt}[f_{(m)}^T \phi_{(m)}(t)] = f_{(m)}^T D_{(m)} \phi_{(m)}(t). \tag{8}$$

In other words, $\frac{d}{dt}f(t)$ is also a linear combination of BPFs with the coefficient vector being $D_{(m)}^T f_{(m)}$.

III. OPM for Linear Systems

A. Theory

A linear time-invariant system can be naturally depicted by a descriptor-format state-space model (a group of DAEs):

$$E\dot{x}(t) = Ax(t) + Bu(t), \tag{9}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^p$, $E, A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$. For ease of notation a zero initial condition is assumed, though not necessary.

Now we want to simulate (9) and obtain the response $x(t)$ in $[0, T)$. We divide the time span into m intervals with the length of each interval (time step) being T/m. To solve this problem using OPM, we assume the state vector $x(t)$ as a series of BPFs. Let

$$x(t) = X\phi_{(m)}(t), \tag{10}$$

where X is an n by m coefficient matrix to be determined. Similarly, the input function $u(t)$ can also be expressed as a BPF series

$$u(t) = U\phi_{(m)}(t). \tag{11}$$

As the input $u(t)$ is given, the p by m coefficient matrix U is known.

Using the differential operational matrix $D_{(m)}$, we can obtain the BPF representation of the rate vector $\dot{x}(t)$

$$\begin{aligned} \dot{x}(t) &= \frac{d}{dt}[X\phi_{(m)}(t)] = X\frac{d}{dt}\phi_{(m)}(t) \\ &\approx XD_{(m)}\phi_{(m)}(t). \end{aligned} \tag{12}$$

Substituting (10), (11) and (12) into (9), we have

$$EXD_{(m)}\phi_{(m)}(t) = AX\phi_{(m)}(t) + BU\phi_{(m)}(t), \tag{13}$$

or simply

$$EXD_{(m)} = AX + BU. \tag{14}$$

To solve the unknown matrix X, we rewrite (14) using the Kronecker product notations. Based on the Kronecker product property, we have

$$\left(D_{(m)}^T \otimes E - I_m \otimes A\right) vec(X) = \left(I_m \otimes B\right) vec(U), \tag{15}$$

where I_m is an identity matrix of the dimension m. In fact, we do not have to solve (15) directly as the differential matrix

$$D_{(m)} = \frac{2}{h}\begin{bmatrix} 1 & -2 & \cdots & (-1)^{m-1}2 \\ & 1 & \cdots & (-1)^{m-2}2 \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix}$$

is a strictly triangular matrix. Instead, the coefficient matrix X can be solved column by column.

B. Adaptive Time Step

OPM can be extended to handle adaptive time steps. For a given time span of interest $[0, T)$, we redefine the BPFs as

$$\phi_i(t) = \begin{cases} 1, & \sum_{j=0}^{i-1} h_j \le t < \sum_{j=0}^{i} h_j \\ 0, & \text{otherwise} \end{cases}, \tag{16}$$

where h_0, h_1, ..., h_{m-1} are adaptive time steps satisfying $h_0 + h_1 + \cdots + h_{m-1} = T$.

Consequently, the integral matrix and the differential matrix become

$$\tilde{H}_{(m)} = \begin{bmatrix} h_1 & & & \\ & h_2 & & \\ & & \ddots & \\ & & & h_{m-1} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 1 & \cdots & 1 \\ & \frac{1}{2} & \cdots & 1 \\ & & \ddots & \\ & & & \frac{1}{2} \end{bmatrix}.$$

$$\tilde{D}_{(m)} = 2 \begin{bmatrix} 1 & -2 & \cdots & (-1)^{m-1}2 \\ & 1 & \cdots & (-1)^{m-2}2 \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix} \begin{bmatrix} h_1 & & & \\ & h_2 & & \\ & & \ddots & \\ & & & h_{m-1} \end{bmatrix}^{-1} \tag{17}$$

Using the new differential matrix $D_{(m)}$, we can solve X similarly by solving

$$\left(\tilde{D}_{(m)}^T \otimes E - I_m \otimes A\right) vec(X) = (I_m \otimes B)\, vec(U). \tag{18}$$

The time step h_i can be determined on the fly by some error control mechanism.

IV. OPM for Simulating Fractional and High-Order Differential Systems

In this section, the OPM method is extended to both fractional differential systems. The high-order differential system is a special case of the fractional differential system when the differential index is a positive integer.

The fractional differential systems under investigation can be depicted as

$$E\frac{d^\alpha}{dt^\alpha}x(t) = Ax(t) + Bu(t). \tag{19}$$

Here α is the differential index which may not be integral.

To solve the system in (19), we have to find the operational matrix for the fractional differential operator. $\frac{d^\alpha}{dt^\alpha}$. We start from the order-1 differential operational matrix in (7). Replacing the Q_m matrix with a scalar variable q, we have

$$D_{(m)} = \frac{2}{h}\frac{1-q}{1+q}\bigg|_{q=Q_m}. \tag{20}$$

Thus,

$$D_{(m)}^\alpha = \left(\frac{2}{h}\frac{1-q}{1+q}\right)^\alpha\bigg|_{q=Q_m}. \tag{21}$$

When constant time step is used, $D_{(m)}$ has only one eigenvalue with m multiplicity. Thus eigendecomposition may not exist because of insufficient eigenvectors. In that case, directly using the matrix power command in MATLAB will not yield the correct answer. To obtain $D_{(m)}^\alpha$, we expand (21) as a polynomial of q and keep the terms up to q^{m-1}. As a result,

$$D_{(m)}^\alpha = \rho_{\alpha,m}(Q_m), \tag{22}$$

where $\rho_{\alpha,m}$ represents a polynomial of order $m-1$ for order-α differentiation. As an illustrative example, we give the operational matrix for order-$3/2$ differentiation with $m = 4$. Expanding (21) with $\alpha = 1.5$ and keeping the first 4 terms, we obtain

$$\rho_{3/2,4}(q) = (2/h)^{3/2}(1 - 3q + \frac{9}{2}q^2 - \frac{11}{2}q^3), \tag{23}$$

which results in

$$D_{(4)}^{3/2} = (2/h)^{3/2} \begin{bmatrix} 1 & -3 & 4.5 & -5.5 \\ & 1 & -3 & 4.5 \\ & & 1 & -3 \\ & & & 1 \end{bmatrix}. \tag{24}$$

It can be straightforwardly verified that $\left(D_{(4)}^{3/2}\right)^2 = \left(D_{(4)}\right)^2$.

On the other hand, if we use adaptive time steps with no two steps being exactly the same, we can directly compute

$$\tilde{D}_{(m)}^\alpha = \left(2 \begin{bmatrix} \frac{1}{h_1} & \frac{-2}{h_2} & \cdots & \frac{(-1)^{m-1}2}{h_{m-1}} \\ & \frac{1}{h_2} & \cdots & \frac{(-1)^{m-2}2}{h_{m-1}} \\ & & \ddots & \vdots \\ & & & \frac{1}{h_{m-1}} \end{bmatrix} \right)^\alpha \tag{25}$$

using eigendecomposition-based methods.

Using the fractional operational matrix $D_{(m)}^\alpha$ (or $\tilde{D}_{(m)}^\alpha$ if adaptive time step is used), the response can be directly solved as

$$x(t) = X\phi_{(m)}(t), \tag{26}$$

with coefficient matrix X solved from

$$\left(\left(D_{(m)}^\alpha\right)^T \otimes E - I_m \otimes A\right) vec(X) = (I_m \otimes B)\, vec(U). \tag{27}$$

Because $D_{(m)}^\alpha$ (or $\tilde{D}_{(m)}^\alpha$) is guaranteed to be triangular regardless of the value of α, we do not have to solve (27) directly. Instead, we can solve X column by column. Here we let

$$\left(D_{(m)}^\alpha\right)^T = \begin{bmatrix} d_{11} & & & \\ d_{21} & d_{22} & & \\ d_{31} & d_{32} & d_{33} & \\ d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix} \tag{28}$$

Because the entries in the operational matrix $D_{(m)}^\alpha$ does not enjoy the special patterns as that used in linear system simulation, solving each column of X of (28) involves manipulation of all the previous columns.

Complexity. Solving each column of X requires one matrix-vector solving and $O(m)$ matrix-vector multiplication. Because E and A are both sparse matrices with $O(n)$ nonzero entries, the complexities of one matrix-vector solving and one matrix-vector multiplication are $O(n^\beta)$ and $O(n)$, respectively.

TABLE I
COMPARISON OF DIFFERENT SIMULATION METHODS IN TERMS OF ACCURACY AND COMPLEXITY

Method	CPU time	Relative Error
FFT-1	6.09 ms	-29.2 dB
FFT-2	40.7 ms	-46.5 dB
OPM	3.56 ms	–

Generally, $1 < \beta < 2$. Hence the overall complexity is $O(n^\beta m + nm^2)$. When handling extremely large systems ($n \gg m$), the complexity can be written simply as $O(n^\beta m)$.

V. NUMERICAL RESULTS

A. A Fractional Differential System Example

The model we use in this example originates from transmission line analysis [7], [8]. The fractional model reads

$$E\frac{d^{1/2}}{dt^{1/2}}x(t) = Ax(t) + Bu(t),$$
$$y(t) = Cx(t), \tag{29}$$

where $x(t) \in \mathbb{R}^7$ and $u(t), y(t) \in \mathbb{R}^2$, i.e., the model has 7 state variables and 2 inputs/outputs.

We want to obtain the response of the model in the time span of $[0, 2.7ns)$ with the step number $m = 8$. To simulate this model using the proposed OPM method, the operational matrix $D_{(8)}^{1/2}$ is first calculated. Then the coefficient matrix of the response is computed using the proposed OPM method. As a comparison, we also simulate the model using fast Fourier transform (FFT). The input signal is first converted to the frequency domain using FFT. Then the response in the frequency domain $X(j\omega_i)$ is calculated at different sampling points. The response is then converted back to time domain using the inverse fast Fourier transform (IFFT). The responses are computed using the FFT method with both 8 frequency sampling points and 100 sampling points.

To measure the global accuracy of the OPM algorithm, we use the relative error

$$err = 20\log\left(\frac{\|y_{OPM}(t) - y_{FFT}(t)\|_2}{\|y_{OPM}(t)\|_2}\right). \tag{30}$$

as a metric. The results of both FFT-1 and FFT-2 are compared with that of OPM. The relative error and CPU time of each algorithm are recorded in Table I. It can be seen from this table that the the waveform solved by FFT-2, which uses more frequency sampling points, is closer to that solved by OPM. We can thereby conclude that the OPM enjoys a good accuracy. On the other hand, the CPU time of OPM is the least, even less than FFT-1. That is because the FFT method involves complex number computation while OPM does not.

B. A High Order Differential System Example

The system used in this example is a 3-D power grid structure with resistors, capacitors and inductors. A second-order differential model can be generated using nodal analysis (NA) due to the existence of inductors. On the other hand, a DAE model can be constructed using modified nodal analysis (MNA) by treating the currents flowing through inductors as state variables. The size of the second-order differential model is $75\ K$ while the order of the DAE model is $110\ K$. We simulate the second-order model using the proposed OPM algorithm. Meanwhile, we simulate the DAE model using backward Euler (b-Euler) and trapezoidal rule as comparison. The CPU time and relative error of each algorithm are reported in Table II.

TABLE II
COMPARISON OF DIFFERENT SIMULATION METHODS

Method	Step	Runtime	Average Relative Error
b-Euler	$h = 10$ ps	334.7 s	-91 dB
	$h = 5$ ps	691.7 s	-92 dB
	$h = 1$ ps	3198 s	-127 dB
Gear	$h = 10$ ps	359.1 s	-134 dB
Trapezoidal	$h = 10$ ps	347.2 s	-137 dB
OPM	$h = 10$ ps	314.6 s	–

VI. CONCLUSION

In this paper, the OPM method for simulating various system models (in the representation of ODEs, DAEs, high-order differential equations and FDEs) is proposed. OPM uses operational matrix instead of finite difference rule to approximate the derivative terms in system models. OPM has similar accuracy and complexity with advanced transient analysis methods for simulating linear systems (ODEs and DAEs). On the other hand, OPM can efficiently simulate fractional differential systems (FDEs), which can not be trivially handled by existing time-domain methods. Moreover, adaptive time step can be used in the scheme of OPM to provide a more flexible simulation with lower runtime.

REFERENCES

[1] R. Bellman, R. Kalaba, and J. Lockett, *Numerical inversion of the Laplace transform*. American Elsevier Publication, 1966.

[2] C. Cheng, Y. Tsay, and T. Wu, "Walsh operational matrices for fractional calculus and their application to distributed systems," *Journal of the Franklin Institute*, vol. 303, no. 3, pp. 267–284, 1977.

[3] B. Davies and B. Martin, "Numerical inversion of the Laplace transform: A survey and comparison of methods," *Journal of computational physics*, vol. 33, no. 1, pp. 1–32, 1979.

[4] C. Hwang, T. Guo, and Y. Shih, "Numerical inversion of multidimensional Laplace transforms via block-pulse functions," in *IEE Proceedings of Control Theory and Applications*, vol. 130, no. 5, 1983, pp. 250–254.

[5] A. Cohen, *Numerical methods for Laplace transform inversion*. Springer Verlag, 2007.

[6] P. Gómez and F. Uribe, "The numerical Laplace transform: An accurate technique for analyzing electromagnetic transients on power system devices," *International Journal of Electrical Power & Energy Systems*, vol. 31, no. 2-3, pp. 116–123, 2009.

[7] D. Baleanu, Z. Guven, and J. Machado, *New trends in nanotechnology and fractional calculus applications*. Springer Verlag, 2010.

[8] Z. Yanzhu and X. Dingyu, "Modeling and simulating transmission lines using fractional calculus," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*. IEEE, 2007, pp. 3115–3118.