A fast repetitive transient analysis of large circuits.

Andrzej Kubaszek
Rzeszow University of Technology
Faculty of Electrical Engineering
ul. W. Pola 2, 35-959 Rzeszow
e-mail: kubaszek@prz.rzeszow.pl

***Abstract* -- An algorithm for the repetitive transient analysis is proposed. The main idea of this algorithm is a generation of the formula which is used in the repetitive loop in such a way, that its calculation time is quite independent of the circuit scale. Advanced techniques used in the repetitive frequency analysis are implemented in the time domain by utilising polynomial number method.**

I. INTRODUCTION

This paper presents a special way of the repetitive time domain analysis. The repetitive transient analysis is to be understood here as calculation of the same characteristic function of the circuit, for example step or impulse response, by many changes of some parameters of the circuit. A better way to achieve a high speed of this type of calculation is to determine the characteristic function as the numerical and symbolic expression, where numerical coefficients depend on the circuit topology and the value of the fixed parameters while only variables are symbolical parameters. The Laplace transform of the characteristic function is in the form of rational function [1]. For two symbolical parameters y_1, y_2 any characteristic function can be expressed as

$$F(s, y_1, y_2) = \frac{N(s, y_1, y_2)}{D(s, y_1, y_2)} \qquad (1)$$

where s is complex variable, $N(s, y_1, y_2)$ and $D(s, y_1, y_2)$ are polynomials. The exponents of y_1 and y_2 are equal 0 or 1, and the exponents of s depend on the number of the circuit elements. When simulation time is to be independent of circuit complexity , then variable s should „disappear" before repetitive analysis loop is performed.

This repetitive transient analysis imitates a method for frequency domain [2, 3], where to calculate function for K points of frequency, the vector of expressions (2) is generated. Each element of this vector corresponds to one frequency point, and is function of symbolical parameters. The complex variable s is substituted by numerical value $j\omega_k$ and „disappears" in numerical coefficients of polynomials. This takes memory resources, but powerfully speeds up the repetitive analysis loop.

$$\left[F(j\omega_k, y_1, y_2)\right] = \left[\frac{N(j\omega_k, y_1, y_2)}{D(j\omega_k, y_1, y_2)}\right] \qquad (2)$$

Repetitive analysis in the time domain can use frequency domain method by utilising inverse FFT algorithm [4]. Disadvantages of this method are leakage and aliasing errors. In symbolic transient analysis differential formula could be approximated by difference formula, which leads to special discrete models of capacitors and inductors. However, simulation time is dependent on circuit complexity ([5], table 1).

II. THE POLYNOMIAL NUMBERS

In the repetitive transient analysis algorithm described in this paper a direct substitution for variable s is applied, by utilising polynomial number method [6, 7]. Through this substitution the function (1) becomes a

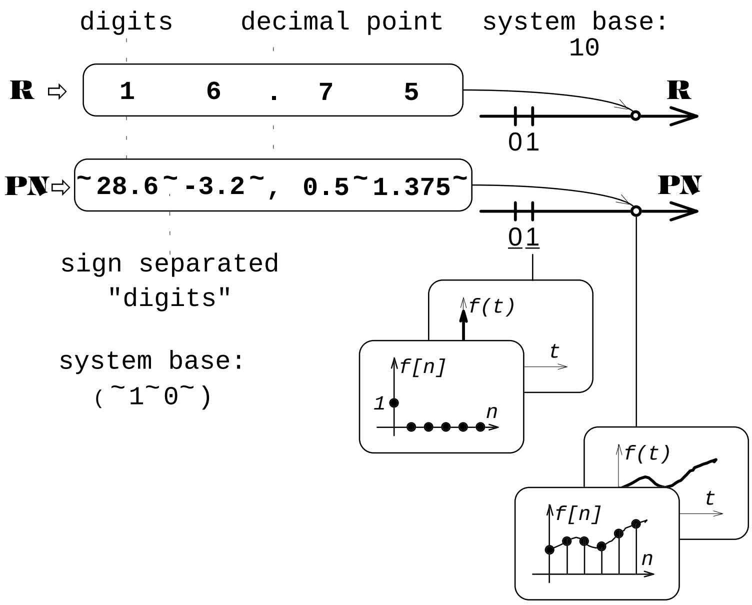

Fig. 1. The polynomial number is generalisation of the real number.

function of symbolical variable y_1, y_2 only and calculation time of its value is quite independent of circuit complexity.

The polynomial number method refers to the approach of numerical operators described by Bellert [8]. A polynomial number (PN) is a generalised number (fig. 1). „Digits” of this number are elements of an arbitrary field, for example real number field. The PN can correspond one to one to the function $f(t)$ or the sequence $\{f_k\}$ (table 1). The Polynomial number method is computer aided operational calculus and PN operation and the function can be implemented simply and easily.

Let us consider a series of samples, for instance:

$$\{3.8\ , 2.5\ , 1.2\ , 0.5\ , ...\} \tag{3}$$

Function $F(z)$ is the Z transform of this series:

$$F(z) = 3.8 + 2.5\, z^{-1} + 1.2z^{-2} + 0.5z^{-3} + ... \tag{4}$$

Polynomial number corresponding to $F(z)$ has „digits” equal to the value of samples (sign „ ˜ ” is used to separate the „digits” and „ ˜, ” points position of coefficient at z^0):

$$\underline{F}_Z = ({}^{\sim}3.8^{\sim}, 2.5^{\sim}1.2^{\sim}0.5^{\sim}...^{\sim}) \tag{5}$$

Variable z performs a role of a base of positional system:

$$z = 1\, z^1 + 0\, z^0 = ({}^{\sim}1^{\sim}0^{\sim}) \tag{6}$$

The Z transform could be utilised to find approximate solution of the differential equation. The approach presented below is based on exchanging operation of integration by its discrete equivalent. In the operational calculus a function $\{x(t)\}$ and its Laplace transform $X(s)$ satisfy a formula:

$$\{x(t)\} = X(s)\, s\, \{1\} \tag{7}$$

where $\{1\}$ is function with value 1 for all $t > 0$ and s is Heaviside’s operator:

$$s = \frac{1}{\int_0^t} \tag{8}$$

TABLE 1.

SIGNALS AND NUMBERS

Type of signal	***Equivalent number***	***Computer implementation***
DC	real number, +, -, *, /	M bits
AC	complex number, +, -, *, /	2 real numbers (2m bits)
transient	polynomial number, +, -, *, /	N real numbers (N-amount of samples)

$\{x(t)\}$ denotes function x, and is used to distinguish it from $x(t)$ - value of function x at the point t [9]. $X(s)$ is as if the measure of function x in the operator space with „unit” (s {1}).

Expression

$$X(s)\, s \tag{9}$$

describes which operations should be done with constant function {1} to obtain $\{x(t)\}$. Substituting constant function {1} by series of constant samples and operator s by its discrete equivalent, we get a formula for approximate series of samples of function x :

$$\underline{x} = X(\underline{p})\, \underline{p}\, ({}^{\sim}0.5^{\sim}, 1^{\sim}1^{\sim}1^{\sim}...^{\sim}) \tag{10}$$

where $\underline{p}$ is the fixed polynomial number corresponding to the algorithm of the numerical integration. For example, when trapezoidal method is considered

$$\begin{aligned} \underline{p} &= \frac{2}{h}\frac{1-z^{-1}}{1+z^{-1}} = \frac{2}{h}\frac{({}^{\sim}1^{\sim}, -1^{\sim})}{({}^{\sim}1^{\sim}, 1^{\sim})} = \\ &= \frac{2}{h}({}^{\sim}1^{\sim}, -2^{\sim}2^{\sim} -2^{\sim}2^{\sim}...) \end{aligned} \tag{11}$$

where h is the sampling period. „Digits” of the polynomial number $\underline{x}$ are approximately equal to samples of function x:

$$\underline{x} \cong ({}^{\sim}x(0)^{\sim}, x(1h)^{\sim}x(2h)^{\sim}x(3h)^{\sim}...^{\sim}) \tag{12}$$

Some explanation should be added to the series

$$({}^{\sim}0.5^{\sim}, 1^{\sim}1^{\sim}1^{\sim}...^{\sim}) \tag{13}$$

in (10) as a digital equivalence of the constant function of value equal to 1 (fig. 2.a). In practice the series of constant samples is used more often (fig. 2.b):

$$({}^{\sim}1^{\sim}, 1^{\sim}1^{\sim}1^{\sim}...^{\sim}) \tag{14}$$

but by using the approximation (14) all responses seem to be displace by a half of the sampling period. That means, that approximation (14) correspond to step function shown in fig. 2.c, hence approximation (13) is more suitable.

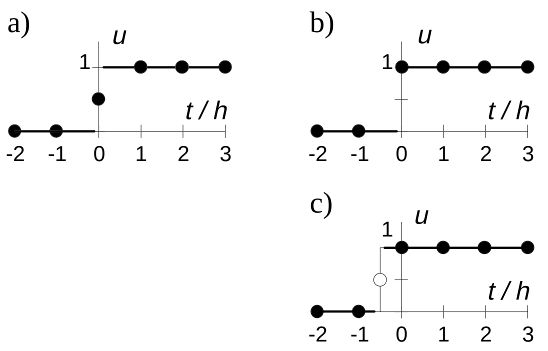

Fig. 2. Discrete approximation of the step function.

All PN operation are easy for computer implementation - both four fundamental operations +, -, *, /, and functions like power function with real exponent, exp, sin, cos. The power and exponent functions are useful for the circuit with distribute parameters. The main advantage of the polynomial number method is that formulae from operational calculus have computer implementation and the algorithms of the operations are „encapsulated" - that means, it is possible to use this method even not knowing these algorithms. This also means, that it is possible to construct PN hardware coprocessor (which will utilise FFT hardware).

III. The simple circuit example

As an example the step response of RLC circuit (fig. 3) will be calculated. The Laplace transform of the current in this circuit is equal to

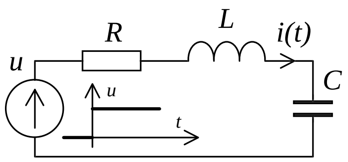

Fig. 3. RLC circuit.

$$I(s) = \frac{s^{-1} + L\,i(0) - u_C(0)s^{-1}}{R + sL + s^{-1}C^{-1}} \qquad (15)$$

Let R=0.5 , C=0.5 , L=3 , $i(0)$=0.2 , $u_c(0)$=0.7 in an arbitrary system of units. Then

$$I(s) = \frac{0.6 + 0.3s^{-1}}{0.5 + 3s + 2s^{-1}} \qquad (16)$$

Applying (10) and (16), we get

$$\underline{i} = I(\underline{p})\,\underline{p}\,(\tilde{}\,0.5\,\tilde{}, 1\,\tilde{}\,1\,\tilde{}\,1\,\tilde{}...\tilde{}) \qquad (17)$$

and if in (11) sampling step h = 0.2 then

$$\underline{i} = (0.103\,\tilde{}, 0.209\,\tilde{}\,0.214\,\tilde{}\,0.212\,\tilde{}\,0.206\,\tilde{}\,0.19\,\tilde{}\,...) \qquad (18)$$

„Digits" of $\underline{i}$ are samples of $i(t)$ what is shown in fig. 4.

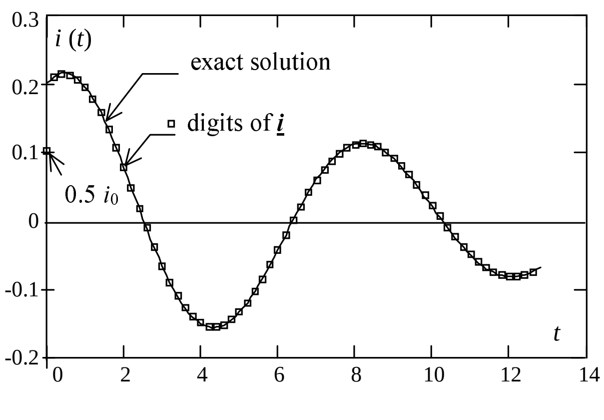

Fig. 4. Exact and approximate step response.

IV. Repetitive transient analysis

The repetitive transient analysis algorithm described in this paper is based on the fast algorithm for repetitive AC analysis [10]. The main idea of this approach is that time for repetitive loop calculation is independent of the circuit scale. Hence the global algorithm is divided into two parts: the preparation part which leads to a formula with symbolical parameters and the repetitive loop part in which fast calculation for large series of fixed value of symbolical arguments is performed.

Let consider an arbitrary circuit with almost all parameters given numerically and 2 parameters which are symbols y_1, y_2 . The Laplace transform of impulse or step response is function $F(s, y_1, y_2)$ (see 1). Applying (10) to (1) we can get series of samples of this time domain response as „digits" of the polynomial number $\underline{F}$:

$$\underline{F}\,(y_1, y_2) = F\,(\underline{p}, y_1, y_2)\,\underline{p}\,(\tilde{}\,0.5\,\tilde{}, 1\,\tilde{}\,1\,\tilde{}\,1\,\tilde{}...\tilde{}) \qquad (19)$$

Simplifying (19), we obtain

$$\underline{F}(y_1, y_2) = \frac{\underline{N}_0 + \underline{N}_1 y_1 + \underline{N}_2 y_2 + \underline{N}_3 y_1 y_2}{\underline{D}_0 + \underline{D}_1 y_1 + \underline{D}_2 y_2 + \underline{D}_3 y_1 y_2} \qquad (20)$$

$\underline{N}_0$ .. $\underline{N}_3$, $\underline{D}_0$ .. $\underline{D}_3$ are the fixed polynomial numbers, and depend on the circuit topology and its numerical parameters.

The purpose of the preparation phase is to determine $\underline{N}_0$ .. $\underline{N}_3$, $\underline{D}_0$ .. $\underline{D}_3$. It could be done in different ways. The one of them is to exchange all impedance of the inductors sL_k and all admittance of the capacitors sC_m by its numerical representations: $\underline{p}L_k$, $\underline{p}C_m$, where L_k and C_m are fixed real numbers and $\underline{p}$ is fixed polynomial number.

In the repetitive loop y_1, y_2 are substituted by series of the fixed real values. The most "expensive" operation of formula (20) is polynomial number division. Asymptotic time complexity of this operation and of FFT is of the same rank.

V. Numerical experiments

As it was mentioned above, time of repetitive simulation of formula (20) plays the main role in global time of calculation. This time depends mainly on number of the polynomial numbers „digits". When the polynomial numbers have 64 „digits", that is when 64 time samples are taken into consideration, then calculation time for formula (20) is equal 4.2ms (Pentium 130MHz). For 128 samples it is equal 15ms. The most important is, that calculation time depends on the number of samples and quantity of symbolical parameters, but is quite independent of circuit complexity.

As an example, an analysis of 1000 variants of parameter in SPICE (parametric transient analysis) and using described algorithm will be compared. The test circuit is filter with 8 operational amplifiers and RC elements (fig. 5, 6). The time of the step response

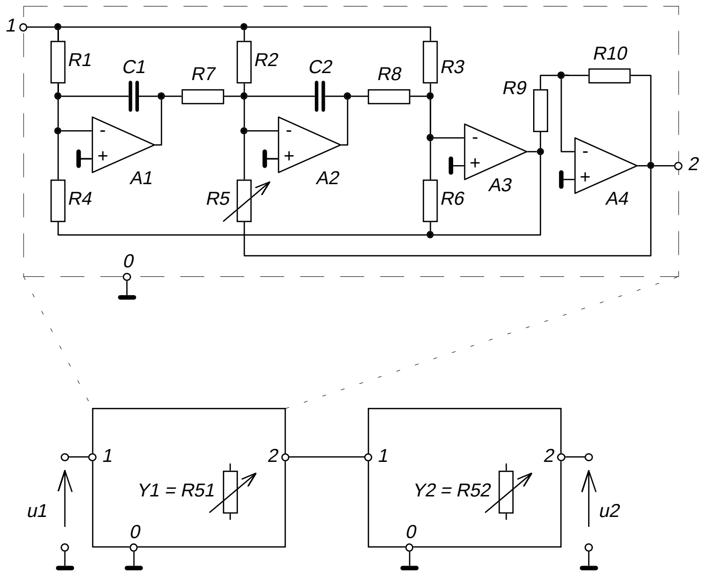

Fig. 5. Test circuit.

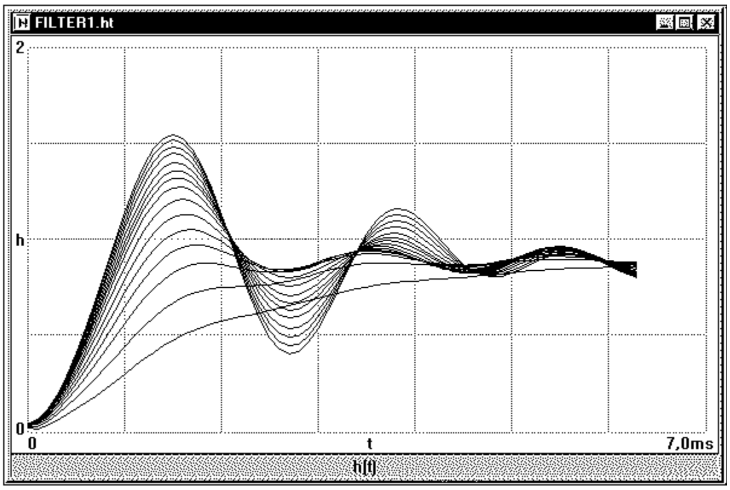

Fig. 6. Series of the step responses.

calculation for SPICE is about 156 seconds, and for described algorithm it is 6 seconds. Of course the comparison is only in terms of quality, because both methods of analysis are quite different.

VI. Conclusion

This paper presents an approach to the repetitive time domain analysis based on the polynomial number method. The polynomial numbers in the transient analysis play similar role as complex numbers in AC analysis - these numbers allow to compute a solution of differential equations in the algebraic manner. This resemblance was used to construct the numerical and symbolic analysis algorithm for time domain in a way known for the frequency domain. The algorithm is fast, because time for repetitive loop calculation is independent of the circuit scale. The main advantages of the presented method are speed and possibility to take into account elements with distributed parameters [7]. The main limits are a limited time of signal observation because of the fixed number of points that are used and a constant sampling period.

References

[1] P.M. Lin, "Symbolic Network Analysis",- *Elsevier, Amsterdam*. 1991

[2] R. Dmytryshyn, "The Use of Symbolic-Numerical Methods for Electronic Circuit Analysis", *Proc. of ISCAS'93* , Chicago, USA ,1993, pp. 1655-1657

[3] R. Dmytryshyn, "Polynomial methods for Symbolic Analysis of Electrical Circuit", Thesis for the Degree of Technical Sciences, Ukraina, Lviv, P. 281 (in Ukrainian), 1996

[4] M. Hassoun, E. Ackerman, "Symbolic simulation of large-scale circuits in both frequency and time domain", *Proc. of 33th MWSCAS*, USA, 1990, pp. 707-710.

[5] S.E. Greenfield, M. Hassoun, " Direct hierarchical symbolic transient analysis of linear circuits ", *IEEE Intern. Symp. on CaS*, New York, USA, 1994, pp 29-32.

[6] A. Kubaszek, "The Polynomial Number Method for Computation Signal Waveforms in Frequency Dependent Parameter Interconnection Lines", *ECCTD-91*, Copenhagen 1991., pp. 709-718.

[7] A. Kubaszek, "Computer aided signal propagation analysis by polynomial number method", *Doctoral thesis*, Technical University of Poznan, Poznan 1994, p. 101.

[8] S. Bellert, "Numerical operators' method", *Rozprawy Elektrotechniczne*, t.1, z. 4, 1959.

[9] J. Mikusinski, "Operational calculus", *Pergamon Press*, 1983.

[10] R. Dmytryszyn, A. Kubaszek, "Sequence of Expressions Generation for the Repetitive Analysis Acceleration ", *Proc. of SMACD'98*, Kaiserslautern 1998.