Application of extrapolation algorithms in nonlinear circuit simulation and optimization with SPICE OPUS

Borut Wagner, Árpád Bűrmen, Janez Puhan, Sašo Tomažič, Tadej Tuma

University of Ljubljana, Faculty of Electrical Engineering, Tržaška cesta 25, Ljubljana, Slovenia

Key words: extrapolation algorithms, circuit simulation, circuit optimization, integrated circuit design, SPICE.

Abstract

In this paper the extrapolation algorithms for vector sequence acceleration are presented. Four extrapolation algorithms are described and their application to circuit simulation is discussed. Steady state evaluation times are compared for the presented extrapolation algorithms and the direct method on real-world test circuits. Results show that the most appropriate extrapolation algorithm for evaluating the steady state of a circuit is the epsilon algorithm. The epsilon algorithm was implemented in SPICE OPUS circuit simulator. The implemented epsilon algorithm was used for optimizing the steady-state response of a test circuit. The accelerated evaluation makes the optimization of steady-state response possible.

Uporaba ekstrapolacijskih postopkov pri simulaciji in optimizaciji nelinearnih vezij s programskim paketom SPICE OPUS

Ključne besede: ekstrapolacijski postopki, simulacija električnih vezij, optimizacija električnih vezij, načrtovanje integriranih vezij, SPICE.

Povzetek

V članku so predstavljeni ekstrapolacijski postopki, ki se uporabljajo za pospeševanje konvergence zaporedij. Opisani so štirje postopki in njihov način uporabe pri simulaciji električnih vezij. Podana je primerjava časov za računanje stacionarnega stanja testnih električnih vezij brez in z uporabo ekstrapolacijskih postopkov. Za implementacijo v programski paket SPICE OPUS je bil izbran epsilon algoritem, ki se je izkazal za najbolj primernega. Postopek je bil uporabljen tudi pri optimizaciji testnega vezja. Zaradi hitrejšega izračuna samega stacionarnega stanja se je pohitril tudi celoten postopek optimizacije stacionarnega stanja in postal primeren za praktično uporabo.

1 Introduction

Extrapolation algorithms for vector sequences /1,2,3,4/ are used for accelerating sequences that converge slowly. The limit of a sequence can be calculated efficiently by evaluating only a few terms of the sequence without the explicit knowledge of the sequence generator. Using an extrapolation algorithm and only few terms of the sequence, a new initial term of the sequence can be calculated. With the new initial term, further terms of the sequence are evaluated by the sequence generator. The procedure is iterated until the differences between consequent terms of the sequence are small enough to assume that we are close to the limit of the sequence.

When computing the steady-state response /5/ of a nonlinear circuit all signals in the circuit are assumed to have the same fundamental frequency f and the period $T = 1/f$. The circuit is simulated starting at some initial condition until steady state is reached. Simulation results $\mathbf{x}(t)$, $t \geq 0$ represent node voltages and branch currents of the circuit at time t. The sequence $\{\mathbf{x}_0^{(i)} = \mathbf{x}(iT)\}$, $i = 0,1,2,3...$ is convergent if the circuit has a steady-state response with period T.

To accelerate the computation of steady state, the circuit is simulated for n periods and the extrapolation method is used on vectors $\mathbf{x}_0^{(0)}, \mathbf{x}_0^{(1)}, \mathbf{x}_0^{(2)}, ...\mathbf{x}_0^{(n)}$ to compute the new initial vector $\mathbf{x}_1^{(0)}$. Then the circuit is simulated starting with initial conditions $\mathbf{x}_1^{(0)}$ and the new sequence $\{\mathbf{x}_1^{(i)} = \mathbf{x}(iT)\}$, $i = 0,1,2,3...$ is extracted from the response of the circuit. This is repeated k-times until $\left\| \mathbf{x}_k^{(n)} - \mathbf{x}_k^{(0)} \right\|$ is small enough to assume the circuit is in steady state.

The paper is organized as follows. In Section 2 a brief description of four extrapolation algorithms (epsilon algorithm, rho algorithm, theta algorithm and topological epsilon algorithm) is given. In Section 3 the simulation of electrical circuits with SPICE OPUS is presented. Problems that occur when steady-state response of circuits has to be computed are presented. The acceleration of the steady-state response computation by means of extrapolation algorithms is described. In Section 4 simulation times for the test circuits (Greinacher rectifier, narrow-band filter, switching power supply) are compared for the direct approach (transient analysis until all initial transients die off) and for the accelerated steady-state computation by means of extrapolation algorithms. Section 5 describes the implementation details of the selected extrapolation algorithm in SPICE OPUS. In Section 6, the implemented algorithm is used in the optimization of a test circuit. Section 7 concludes the paper.

2 Extrapolation algorithms

Vector sequence

$$\left\{\mathbf{x}_0^{(i)}\right\},\ i=0,1,2,3... \tag{1}$$

is generated from the initial vector $\mathbf{x}_0^{(0)}$ using the sequence generator $\mathbf{x}_0^{(i+1)}=\mathbf{F}(\mathbf{x}_0^{(i)})$, $i=0,1,2,3...$. If the sequence $\left\{\mathbf{x}_0^{(i)}\right\}$ is convergent, the limit of the sequence is denoted by $\mathbf{x}$.

An extrapolation algorithm $\mathbf{E}$ generates a new vector sequence $\left\{\mathbf{x}_{k+1}^{(0)}\right\}$, $k=0,1,2,3...$ with the following two steps

$$\mathbf{x}_{k+1}^{(0)}=\mathbf{E}(\mathbf{x}_k^{(0)},\mathbf{x}_k^{(1)},\mathbf{x}_k^{(2)}...\mathbf{x}_k^{(n_k)})\ ,\ k=0,1,2,3... \tag{2}$$

and

$$\mathbf{x}_k^{(i)}=\mathbf{F}(\mathbf{x}_k^{(i-1)})\ ,\ i=1,2,3...n_k \text{ for each } k=0,1,2,3...\ . \tag{3}$$

The extrapolation algorithm generates the first term $\mathbf{x}_{k+1}^{(0)}$ of a new sequence $\left\{\mathbf{x}_{k+1}^{(i)}\right\}, i=0,1,2...$ from the n_k+1 terms of sequence $\left\{\mathbf{x}_k^{(i)}\right\}, i=0,1,2...n_k$.

For each k , $n_k+1\le d$, $d=\dim(\mathbf{x}_k^{(0)})$ has to be satisfied. If $n_k+1>d$ the extrapolation algorithm is overdetermined.

The sequence $\left\{\mathbf{x}_{k+1}^{(0)}\right\}, k=0,1,2,3...$ generated by (2) and (3) converges faster than sequence (1) to the same limit $\mathbf{x}$.

In the following subsections four extrapolation algorithms are described: epsilon algorithm, rho algorithm, theta algorithm and topological epsilon algorithm. The algorithms differs only in the definition of $\mathbf{E}$.

2.1 Epsilon algorithm

For each k in (2) a two-dimension array depicted in Fig. 1 is formed by

$$\boldsymbol{\varepsilon}_{-1}^{(i)} = \mathbf{0} \ , \ i = 1,2,3 \ldots n_k \tag{4a}$$

$$\boldsymbol{\varepsilon}_0^{i} = \mathbf{x}_k^{(i)} \ , \ i = 0,1,2 \ldots n_k \tag{4b}$$

$$\boldsymbol{\varepsilon}_{j+1}^{(i)} = \boldsymbol{\varepsilon}_{j-1}^{(i+1)} + (\boldsymbol{\varepsilon}_j^{(i-1)} - \boldsymbol{\varepsilon}_j^{(i)})^{-1} \ , \ i = 0,1,2 \ldots n_k \ , \ j = 0,1,2 \ldots n_k - 1 \text{ where } i + j + 1 \le n_k \tag{4c}$$

$$\begin{array}{ccccc}
 & \boldsymbol{\varepsilon}_0^{(0)} & & & \\
\boldsymbol{\varepsilon}_{-1}^{(1)} & & \boldsymbol{\varepsilon}_1^{(0)} & & \\
 & \boldsymbol{\varepsilon}_0^{(1)} & & \ddots & \\
\boldsymbol{\varepsilon}_{-1}^{(2)} & & \boldsymbol{\varepsilon}_1^{(1)} & & \\
 & \boldsymbol{\varepsilon}_0^{(2)} & & \cdots & \boldsymbol{\varepsilon}_{n_k}^{(0)} \\
 & & \boldsymbol{\varepsilon}_1^{(2)} & & \\
\vdots & \vdots & \vdots & & \\
 & \boldsymbol{\varepsilon}_0^{(n_k-1)} & & \cdot^{\cdot^{\cdot}} & \\
\boldsymbol{\varepsilon}_{-1}^{(n_k)} & & \boldsymbol{\varepsilon}_1^{(n_k-1)} & & \\
 & \boldsymbol{\varepsilon}_0^{(n_k)} & & &
\end{array}$$

Figure 1: Two-dimension array for the epsilon algorithm

The procedure described by (4), is the so-called epsilon algorithm /1/. The inverse of the vector in (4c) is computed as

$$\mathbf{x}^{-1} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2^2} \quad . \tag{5}$$

If n_k is an even number, $\boldsymbol{\varepsilon}_{n_k}^{(0)}$ is the result of the epsilon algorithm, so the extrapolation function (2) is defined as

$$\mathbf{E}(\mathbf{x}_k^{(0)}, \mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)} \ldots \mathbf{x}_k^{(n_k)}) = \boldsymbol{\varepsilon}_{n_k}^{(0)} \quad . \tag{6}$$

2.2 Rho algorithm

The rho algorithm /1/ is similar to the epsilon algorithm. For each k in (2) n_k has to be even. The result is $\boldsymbol{\rho}_{n_k}^{(0)}$ obtained from

$$\boldsymbol{\rho}_{-1}^{(i)} = \mathbf{0} \ , \ i = 1,2,3 \ldots n_k \tag{7a}$$

$$\boldsymbol{\rho}_0^{i} = \mathbf{x}_k^{(i)} \ , \ i = 0,1,2 \ldots n_k \tag{7b}$$

$$\boldsymbol{\rho}_{j+1}^{(i)} = \boldsymbol{\rho}_{j-1}^{(i+1)} + (j+1)\left[\boldsymbol{\rho}_j^{(i-1)} - \boldsymbol{\rho}_j^{(i)}\right]^{-1} \ , \ i = 0,1,2 \ldots n_k \ , \ j = 0,1,2 \ldots n_k - 1 \text{ where } i + j + 1 \le n_k \ . \tag{7c}$$

The inverse of the vector in (7c) is calculated in the same manner as in (5).

2.3 Theta algorithm

Extrapolation by theta algorithm /1/ (for an even n_k) is computed using the following equations

$$\vartheta_{-1}^{(i)} = \mathbf{0} \ , \ i = 1,2,3 \ldots n_k \tag{8a}$$

$$\vartheta_0^{i} = \mathbf{x}_k^{(i)} \ , \ i = 0,1,2 \ldots n_k \tag{8b}$$

$$\vartheta_{2j+1}^{(i)} = \vartheta_{2j-1}^{(i+1)} + \left[\Delta \vartheta_{2j}^{(i)}\right]^{-1} \ , \qquad i = 0,1,2 \ldots n_k \ , \ j = 0,1,2 \ldots \frac{n_k}{2} - 1 \text{ where } i + 2j + 1 \le n_k \tag{8c}$$

$$\vartheta_{2j+2}^{(i)} = \vartheta_{2j}^{(i+1)} + \left(\left[\Delta\vartheta_{2j}^{(i+1)}\right], \left[\Delta\vartheta_{2j+1}^{(i+1)}\right] \right) \cdot \left(\Delta^2\vartheta_{2j+1}^{(i+1)}\right)^{-1}, \tag{8d}$$

$$i = 0,1,2\ldots n_k \ , \ j = 0,1,2\ldots \frac{n_k}{2} - 1 \text{ where } i + 2j + 2 \le n_k$$

The abbreviations in (8c) and (8d) are

$$\Delta\vartheta_j^{(i)} = \vartheta_j^{(i+1)} - \vartheta_j^{(i)} \tag{8e}$$

$$\Delta^2\vartheta_j^{(i)} = \Delta\vartheta_j^{(i+1)} - \Delta\vartheta_j^{(i)} \tag{8f}$$

and the inverses in (8c) and (8d) are calculated by (5).

For each k and even n_k the theta algorithm extrapolation function $\mathbf{E}$ is defined as

$$\mathbf{E}(\mathbf{x}_k^{(0)}, \mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)} \ldots \mathbf{x}_k^{(n_k)}) = \vartheta_{n_k}^{(0)} \tag{9}$$

2.4 Topological epsilon algorithm

If the inverse of a vector $\mathbf{x}$ is defined as

$$\mathbf{x}^{-1} = \frac{\mathbf{y}}{(\mathbf{y}, \mathbf{x})} \ , \tag{10}$$

the inverse is the so-called inverse of vector $\mathbf{x}$ with respect to $\mathbf{y}$.

In the topological epsilon algorithm the inverses in odd and even terms are computed with respect to different vectors.

For each interation k of the extrapolation algorithm, the topological epsilon algorithm /1/ is defined by

$$\boldsymbol{\varepsilon}_{-1}^{(i)} = \mathbf{0} \ , \ i = 1,2,3\ldots n_k \tag{11a}$$

$$\boldsymbol{\varepsilon}_0^{i} = \mathbf{x}_k^{(i)} \ , \ i = 0,1,2\ldots n_k \tag{11b}$$

$$\boldsymbol{\varepsilon}_{2j+1}^{(i)} = \boldsymbol{\varepsilon}_{2j-1}^{(i+1)} + \frac{\mathbf{y}}{(\mathbf{y} \, , \Delta\boldsymbol{\varepsilon}_{2j}^{(i)})} \ , \ i = 0,1,2\ldots n_k \ , \ j = 0,1,2\ldots \frac{n_k}{2} - 1 \text{ where } i + 2j + 1 \le n_k \tag{11c}$$

$$\boldsymbol{\varepsilon}_{2j+2}^{(i)} = \boldsymbol{\varepsilon}_{2j}^{(i+1)} + \frac{\Delta\boldsymbol{\varepsilon}_{2j}^{(i)}}{(\Delta\boldsymbol{\varepsilon}_{2j}^{(i)} \, , \Delta\boldsymbol{\varepsilon}_{2j+1}^{(i)})} \ , \ i = 0,1,2\ldots n_k \ , \ j = 0,1,2\ldots \frac{n_k}{2} - 1 \text{ where } i + 2j + 2 \le n_k \tag{11d}$$

Inverses in the odd terms (11c) are computed with respect to an arbitrary $\mathbf{y}$ such that all terms $\boldsymbol{\varepsilon}_{2j+1}^{(i)}$ exist. Inverses in (11d) are computed with respect to $\Delta\boldsymbol{\varepsilon}_{2j}^{(i)}$.

The result of the topological epsilon algorithm is $\boldsymbol{\varepsilon}_{n_k}^{(0)}$, so the extrapolation function $\mathbf{E}$ is defined as

$$\mathbf{E}(\mathbf{x}_k^{(0)}, \mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)} ... \mathbf{x}_k^{(n_k)}) = \boldsymbol{\varepsilon}_{n_k}^{(0)} .$$

3 Simulation of electrical circuits

Nonlinear dynamical electrical circuits can be described by a system of ordinary differential equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathrm{t}) \tag{12}$$

where $\mathbf{x}(t)$ represents the node voltages and the branch currents of the circuit. In transient analysis, (12) is solved starting from the initial value $\mathbf{x}(0)$. The resulting waveforms are represented by vector functions

$$\mathbf{x}(t) \ , \ 0 \le t \le t_n . \tag{13}$$

For further reference, $\mathbf{x}(t)$ is a column vector with d components:

$$\mathbf{x}(t) = \left[{}_1x(t), \ {}_2x(t), \ \cdots \ {}_\mathrm{d}x(t) \right]^\mathrm{T} . \tag{14}$$

(12) is numerically integrated and solved using the Newton-Raphson iterative metod for times $t = 0 \le t_1 \le t_2 \le \cdots \le t_n$. Time step $\Delta t_i = t_{i+1} - t_i$ should be small enough to avoid numerical errors. Decreasing the time step increases the computation time as more points have to be evaluated.

3.1 SPICE OPUS circuit simulator and optimizer

The simulation of electrical circuits can be done with the SPICE OPUS circuit simulator and optimizer /6/. Circuit simulation is a part of every circuit optimization /7/

where parameters of the circuit are sought, subject to design requirements. During optimization, a circuit is simulated many times, so individual simulations should be as short as possible.

When computing the steady-state response of a circuit, a problem occurs when the period of the steady-state response is much smaller than the largest time constant of the circuit. Such circuits must be simulated for hundreds or even thousands of periods before they reach steady state. To obtain sufficient accuracy, a hundred or more time points must be evaluated per period of the response. Therefore, several million time points have to be evaluated before steady state is reached.

Using the extrapolation algorithms described in Section 2 computing steady-state response of such electrical circuits can be significantly accelerated.

3.2 Application of extrapolation algorithms in circuit simulation

To compute the steady-state response of a circuit by means of an extrapolation algorithms, the circuit is simulated (solving (12)) starting from the initial value $\mathbf{x}(0)=\mathbf{0}$ for times $0 \le t \le t_n + t_{del}$. Terms of sequence (1) are extracted from the resulting waveforms $\mathbf{x}(t)$:

$$\mathbf{x}_0^{(i)} = \mathbf{x}(t_{del} + iT) \ , \quad i = 0,1,2 \ldots n \quad , \quad n = \lfloor \mathrm{t_n} / \mathrm{T} \rfloor \qquad (15)$$

where T is the period of the steady-state response of the circuit and t_{del} is the time where the first term $\mathbf{x}_0^{(0)}$ is sampled. Then the extrapolation algorithm is used on sequence $\{\mathbf{x}_0^{(i)}\}$ to generate the extrapolated vector $\mathbf{x}_1^{(0)}$ which represents the initial value $\mathbf{x}(0)$ for a new simulation.

This iterative process is repeated k-times until $\left\|\mathbf{x}_k^{(n)} - \mathbf{x}_k^{(0)}\right\|$ in the last (k-th) iteration is small enough to assume that the circuit is in steady state.

We can assume that steady state has been reached when the relative and the absolute tolerance criteria

$$\left|{}_l x_k^{(n)} - {}_l x_k^{(0)}\right| \le \delta_a + \max\left\{\left|{}_l x_k^{(n)}\right|, \left|{}_l x_k^{(0)}\right|\right\}\delta_r \tag{16}$$

for all $l = 1, 2, \cdots d$ are satisfied.

In practice, it is sufficient if δ_a and δ_r are less than 10^{-5} and 10^{-4}, respectively and greater than the precision of the data representation (i.e. the relative and absolute precision of *double* is 10^{-14} and 10^{-320}, respectively).

A flow chart of the extrapolation algorithm is depicted in Fig. 2.

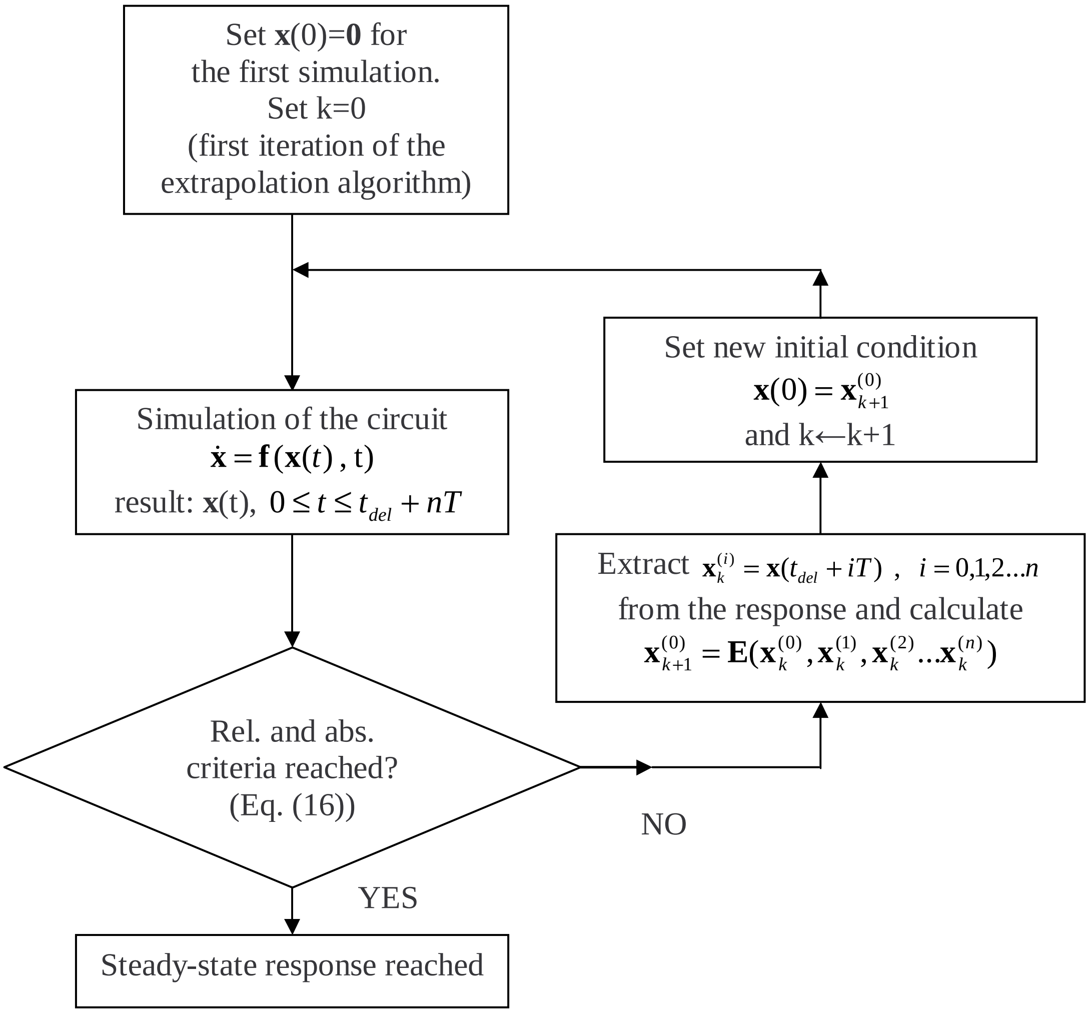

Set $\mathbf{x}(0)=\mathbf{0}$ for
the first simulation.
Set k=0
(first iteration of the
extrapolation algorithm)
Simulation of the circuit
$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathrm{t})$
result: $\mathbf{x}$(t), $0 \le t \le t_{del} + nT$
Rel. and abs.
criteria reached?
(Eq. (16))
YES
NO
Steady-state response reached
Extract $\mathbf{x}_k^{(i)} = \mathbf{x}(t_{del} + iT)$, $i = 0,1,2...n$
from the response and calculate
$\mathbf{x}_{k+1}^{(0)} = \mathbf{E}(\mathbf{x}_k^{(0)}, \mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)} ... \mathbf{x}_k^{(n)})$
Set new initial condition
$\mathbf{x}(0) = \mathbf{x}_{k+1}^{(0)}$
and k←k+1

Figure 2: Flow chart of the extrapolation algorithm for determing steady-state response of a circuit.

3.3 Accelerating the computation of steady state of an electrical circuit

The steady-state response is important in the analysis of power conversion circuits, in the evaluation of nonlinear properties of narrow-band circuits, etc. For such circuits the evaluation of long transients can be accelerated using extrapolation algorithms.

In the following section the above mentioned extrapolation algorithms are tested with the Greinacher rectifier, narrow-band filter and switching rectifier test circuits.

4 Comparison

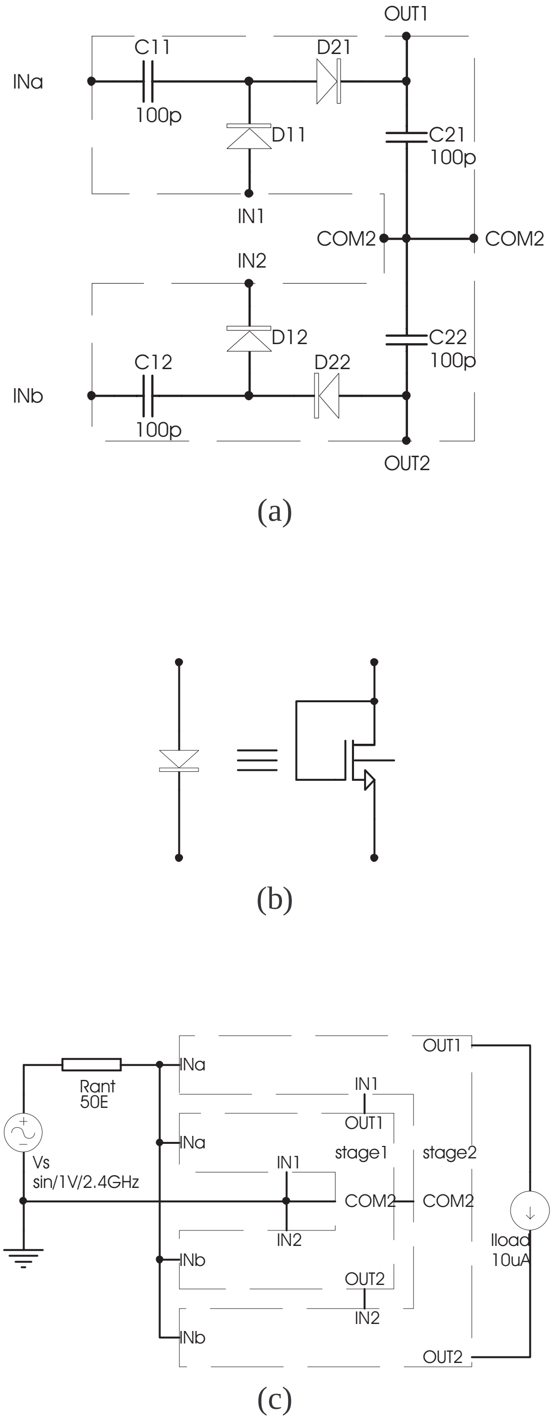
The test circuits were simulated using the direct approach (running a transient analysis until all initial transients died off). It took several thousand periods for the simulator to reach the steady state.

The steady-state responses of the circuits were also computed using the transient analysis accelerated by an extrapolation algorithm (epsilon algorithm, rho algorithm, theta algorithm and topological epsilon algorithm). The total number of periods required for computing the steady state has been greatly reduced.

In the following subsections the test circuits and the detailed results of the steady-state simulation are presented.

4.1 Test circuits

The two-stage Greinacher rectifier test circuit is depicted in Fig. 3.



(a)

(b)

(c)

Figure 3: (a) Single stage of a Greinacher rectifier.

(b) Implementation of a diode with an n-channel MOS.

(c) Test circuit: two-stage Greinacher rectifier.

Fig. 3a represents one stage of the Greinacher rectifier. Diodes in Fig. 3a are implemented as n-channel MOS transistors in 0.18µm TSMC technology (Fig. 3b). Two stages are connected together as depicted in Fig. 3c. At the input of the circuit a sinusoidal voltage source V_s with amplitude 1 V and frequency 2.4 GHz is connected in series with resistor $R_{ant} = 50\Omega$. The output of the circuit is loaded with $I_{load} = 10\mu A$.

In Fig. 4 a narrow-band filter is depicted. At the input of the circuit (node 1) a sinusoidal voltage source V_{in} with amplitude 1 V and frequency 32767.41 Hz is connected. The output of the circuit is at node 4. The quartz crystal X_1 is modelled with resistor R_s, inductor L_s and capacitors C_s and C_p.

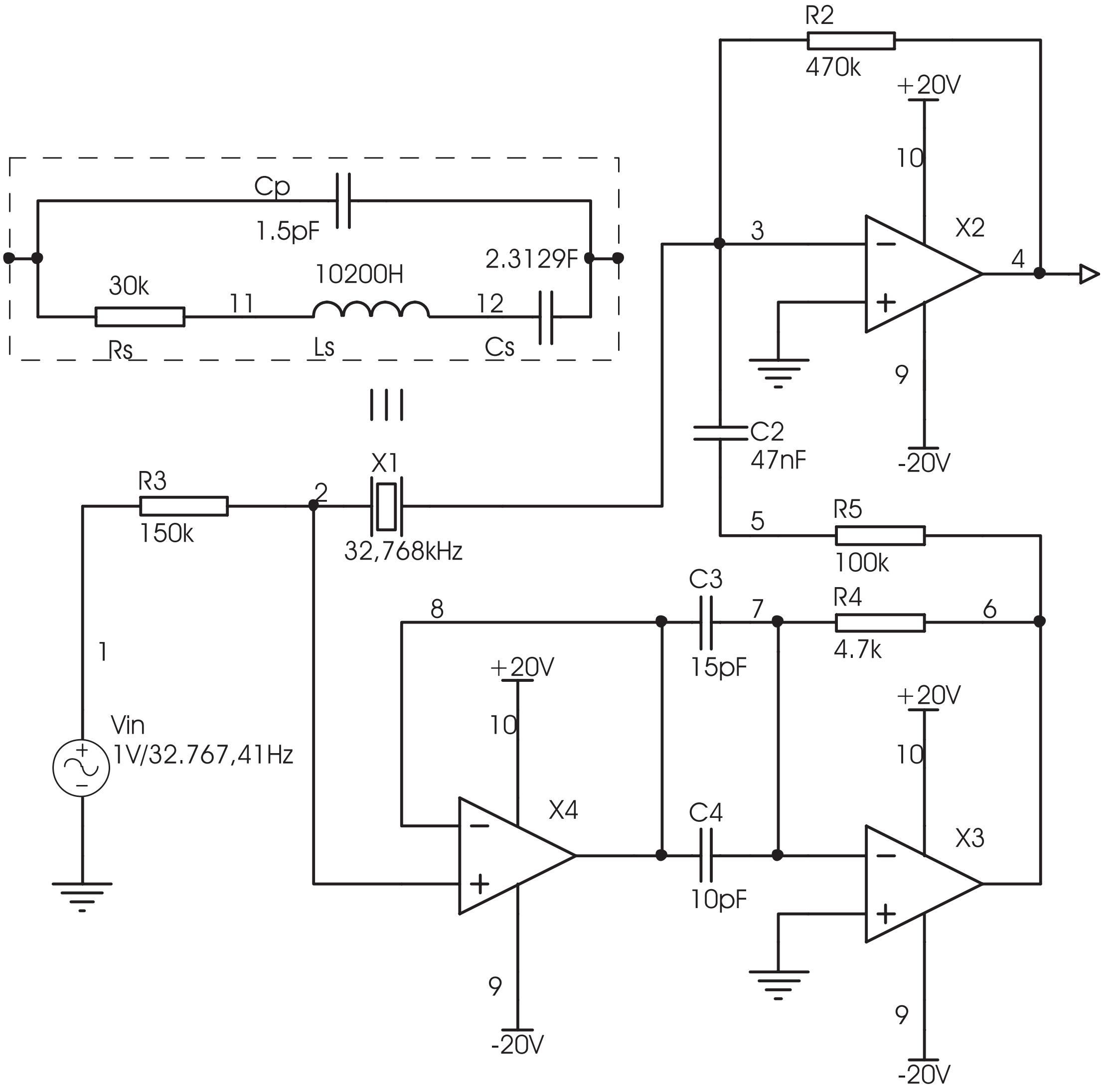


Figure 4: Narrow-band filter.

The circuit in Fig. 5 is a switching power supply. The power source (DC voltage 20 V) is connected to node 1. DC voltage of the V_{pulse} is 10 V, amplitude is 20 V, rise and fall times are $10ns$, and the duty-cycle is 9%. At the output (node 4) $R_{load} = 1k\Omega$ is connected.

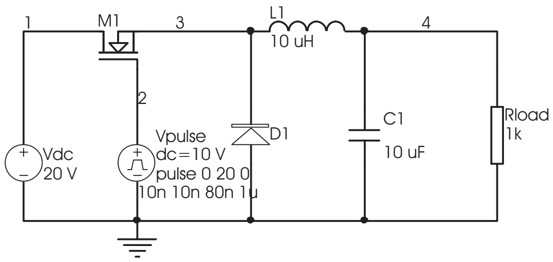


Figure 5: Switching power supply.

4.2 Results

The steady-state response of the Greinacher rectifier is shown in Fig. 6.

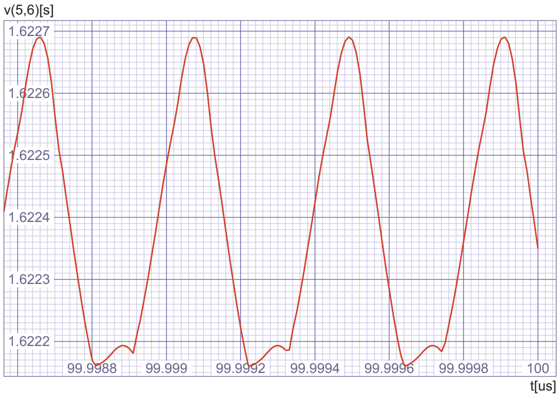


Figure 6: The steady-state response of the two-stage Greinacher rectifer (loaded with $I_{load} = 10\mu A$).

The circuit reaches the steady state in $100\,\mu s$ or in 240000 periods of the signal V_s. The steady state of the circuit was also computed with extrapolation algorithm. Parameter n_k (equations (4), (7), (8) and (11)) was set to 6. The extrapolation algorithm was stopped when the relative and the absolute criteria (16) were satisfied ($\delta_a = 10^{-6}$ and $\delta_r = 10^{-6}$). The number of iterations of the extrapolation algorithm (k)

and the number of periods the circuit was simulated before it reached the steady state ($\sum_k (n_k + t_{del} / T)$) are listed in Table 1.

	Greinacher rectifier		Narrow-band filter		Switching power supply	
	Iterations	**Periods**	**Iterations**	**Periods**	**Iterations**	**Periods**
Epsilon algorithm	52	330	3	54	4	24
Rho algorithm	37	222	3	87	5	30
Theta algorithm	99	594	9	144	12	72
Topological epsilon algorithm	274	1644	3	54	4	24
Direct transient analysis	/	240000	/	65535	/	100000

Table 1: The number of iterations of the extrapolation algorithm and the number of simulated periods in the computation of the steady-state response.

Using an extrapolation algorithm, the steady-state response of the test circuit was computed more than 140 times faster than with the direct approach (transient analysis until all initial transients die off).

The narrow-band filter took 2 s (65535 periods) of the signal V_{in} before it reached steady state (Fig. 7). Computing the steady state with the epsilon, rho, theta and topological epsilon algorithm took 54, 87, 144 and 54 periods, respectively (Table 1) with $\delta_a = 5 \cdot 10^{-6}$ and $\delta_r = 5 \cdot 10^{-6}$.

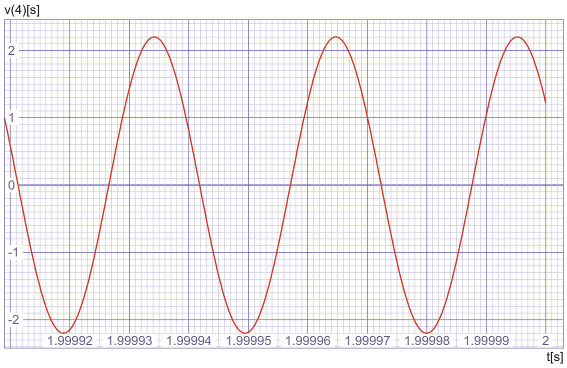


Figure 7: The steady-state response of the narrow-band filter.

The steady-state response of the switching power supply is depicted in Fig. 8. Steady state is reached in 100 ms or in 100000 periods of the signal V_{pulse}. With the epsilon, rho, theta and topological epsilon algorithm computing steady state took 24, 30, 72 and 24 periods, respectively (Table 1) with $\delta_a = 10^{-6}$ and $\delta_r = 10^{-7}$.

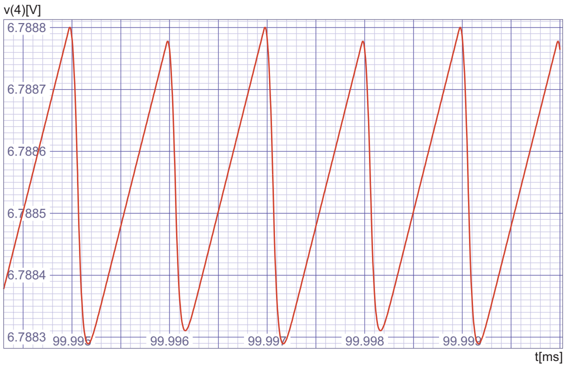


Figure 8: The steady-state response of the switching power supply.

5 Implementation of an extrapolation algorithm in SPICE OPUS

5.1 Choice of the extrapolation algorithm

The results listed in Table 1 show that for the selected test circuits the best of the extrapolation algorithms is the epsilon algorithm. Therefore we chose it for the implementation in SPICE OPUS.

The epsilon algorithm was also chosen due to it's simplicity and good results obtained with other circuits the extrapolation algorithm was tested on /8,9/.

5.2 SSSE analysis

The new *SSSE* (Steady-State Shooting with Extrapolation) analysis was implemented in the SPICE OPUS circuit simulator /6/. The syntax of the *ssse* analysis is as follows:

ssse <freq> [step [skip [periods]]] [history]

< > ... *required parameter*

[ ] ... *optional parameter*

freq represents the fundamental frequency (f) of the steady-state response, *step* represents the basis for the variable time step (same as the first argument for the transient analysis, default value: 10/ *freq*), *skip* represents the time skipped (t_{del} in (15)) before the response is sampled (default value: 0) and *periods* gives the number of periods (n_k) that are taken into account for sampling (default value: 2).

The iteration limit of the epsilon extrapolation algorithm is given by the *itl2* option. Convergence (see (16)) is defined with the absolute and the relative tolerances given by products of *sssetol**vntol*, *sssetol**abstol*, and *sssetol**reltol* options. Steady state is reached when the difference (16) is within tolerances.

By default only the last transient run results (steady-state response) are saved in a plot as final results of the ssse analysis. If the *history* flag is set then all transient iterations performed during the steady-state analysis are saved in plots.

The steady-state response of the Greinacher rectifier in Fig. 6 was obtained with the following SPICE OPUS commands.

```
.options itl2=1000
.options sssetol=1e-2
.options abstol=1e-4
.options reltol=1e-4
.control
set xmumult=1
ssse 2.4g 2p 0 6 history
.endc
```

The *xmumult* option is set to 1 to turn off numerical oscillation detection, which can interfere with the extrapolation algorithm.

6 Optimization of electrical circuits

Circuit simulation is a part of every circuit optimization /7/ where parameters of the circuit are sought subject to design requirements. During optimization, a circuit is simulated many times, so individual simulations should be as short as possible.

If the circuit is being optimized and the steady-state response is computed in every iteration of the optimization, fast computation of the steady-state response is a prerequisite for realistic optimization times.

6.1 Optimization of the Greinacher rectifier test circuit

To demonstrate the application of the *ssse* command, an optimization proccess was run on the test circuit in Fig. 3.

Table 2 lists the optimization parameters.

Parameter 0	Width of transistors modelling diodes D11 and D12 (Stage 1)
Parameter 1	Width of transistors modelling diodes D21 and D22 (Stage 1)
Parameter 2	Width of transistors modelling diodes D11 and D12 (Stage 2)
Parameter 3	Width of transistors modelling diodes D21 and D22 (Stage 2)
Parameter 4	Capacitance of all capacitors of Stage 1 and Stage 2

Table 2: The optimization parameters.

The optimization was started at *Initial value* of the parameters in Table 3. The explicit constrains are defined by the columns *Minimum value* and *Maximum value*. The parameters were allowed to change by the value of the *Increment* column in Table 3.

	Initial value	**Minimum value**	**Maximum value**	**Increment**
Parameter 0	10 µm	0.22 µm	100 µm	0.18 µm
Parameter 1	10 µm	0.22 µm	100 µm	0.18 µm
Parameter 2	8 µm	0.22 µm	100 µm	0.18 µm
Parameter 3	8 µm	0.22 µm	100 µm	0.18 µm
Parameter 4	100 pF	10 pF	300 pF	10 pF

Table 3: Initial, minimum, maximum values, and increment of optimization parameters.

The circuit behavior was described with the cost function which consists of penalty functions of measurement values multiplied by weights /10/.

Table 4 shows the measurements, measurements goals, norms and weights.

If a *measurement value* violates a *measurement goal* by *norm*, the contribution to the cost function equals *measurement weight*.

		goal	**weight**	**norm**
Measurement 1	V_{DC_load}	> 2 V	3	1 V
Measurement 2	V_{ripple_load}	< 250 μV	1	100 μV
Measurement 3	V_{DC_noload}	> 3 V	1	1 V
Measurement 4	V_{ripple_noload}	< 140 μV	1	100 μV

Table 4: Measurements performed on the test circuit.

Measurements are defined by

$$V_{DC\_load} = \left. \left\{ \frac{1}{T} \int_{(n_k-1)T}^{n_k T} (v_{OUT2}(t) - v_{OUT1}(t)) dt \right\} \right|_{I_{load}=10\mu A} \tag{17a}$$

$$V_{ripple\_load} = \left. \left\{ \max_{t\in[(n_k-1)T, n_k T]} (v_{OUT2}(t) - v_{OUT1}(t)) - \min_{t\in[n_k-1)T, n_k T]} (v_{OUT2}(t) - v_{OUT1}(t)) \right\} \right|_{I_{load}=10\mu A} \tag{17b}$$

$$V_{DC\_noload} = \left. \left\{ \frac{1}{T} \int_{(n_k-1)T}^{n_k T} (v_{OUT2}(t) - v_{OUT1}(t)) dt \right\} \right|_{I_{load}=0} \tag{17c}$$

$$V_{ripple\_noload} = \left. \left\{ \max_{t\in[(n_k-1)T, n_k T]} (v_{OUT2}(t) - v_{OUT1}(t)) - \min_{t\in[(n_k-1)T, n_k T]} (v_{OUT2}(t) - v_{OUT1}(t)) \right\} \right|_{I_{load}=0} . \tag{17d}$$

Measurements 1 and 2 were made with $I_{load} = 10\mu A$ (Fig. 3c). Measurements 3 and 4 were made with no load ($I_{load} = 0$).

The optimization took 2964 iterations. Every iteration of the optimization included two SSSE analyses.

On an AMD Athlon XP 2500+ (1.83 GHz clock) computer with 512 MB of RAM, the optimization took 3 hours and 10 minutes. Optimum parameters are shown in Table 5 and the final measurement values in Table 6.

	Value
Parameter 0	16.60 µm
Parameter 1	21.10 µm
Parameter 2	27.76 µm
Parameter 3	26.68 µm
Parameter 4	300 pF

Table 5: Optimum parameter values after the optimization.

	Before optimization	**After optimization**	**Improvement**
Measurement 1	1.624 V	1.794 V	10.5 %
Measurement 2	326.0 µV	249.9 µV	23.3 %
Measurement 3	2.670 V	2.911 V	9.0 %
Measurement 4	176.5 µV	138.9 µV	21.3 %

Table 6: Measurement values before and after the optimization.

For Measurements 2 and 4 the goals were achieved, while Measurements 1 and 3 violated the goals. The final cost function value was 0.6792.

Table 6 also lists the measurement values before optimization (at initial values of the parameters in Table 3). Average improvement of the measurements is 16 %.

In Sections 4.2 and 5.1 we have demonstrated that the acceleration ratio for the Greinacher rectifier test circuit (in case of the epsilon algorithm) is 727 ($240000/330 \cong 727$, see Table 1). If ordinary transient analysis was used, the optimization would take more than 95 days. This clearly demonstrates that if the steady-state response of a circuit has to be computed in an optimization loop, an efficient extrapolation algorithm is necessary.

7 Conclusions

Four extrapolation algorithms (epsilon, rho, theta and topological epsilon algorithm) were described in this paper. Using these algorithms the steady-state computation has been accelerated several hundred times.

The extrapolation algorithms were tested on three test circuits (Greinacher rectifier, narrow-band filter and switching power supply). The epsilon algorithm was shown to be the most appropriate for the implementation in the SPICE OPUS circuit simulator. The steady-state responses of the test circuits were computed several hundred times faster than with the direct approach (transient analysis until all initial transients die off). A new SPICE OPUS analysis utilizing the epsilon algorithm was implemented.

The Greinacher rectifier test circuit was optimized using the SSSE analysis. The purpose of the optimization was to show that the optimization of the steady-state response can be accelerated significantly by using extrapolation methods. Optimization with the SSSE analysis (accelerated transient analysis) lasted 3 hours and 10 minutes. It has been estimated that the optimization with the direct approach (direct transient analysis) would take more than 95 days.

8 Acknowledgment

The research has been supported by the Slovenian Research Agency within programme P2-0246 - Algorithms and optimization methods in telecommunications.

9 References

/1/ A. Sidi, W. F. Ford, D. A. Smith, Acceleration of Convergence of Vector Sequences, SIAM Journal on Numerical Analysis, vol. 23, no. 1, pp. 178-196, February 1986.

/2/ P. R. Graves-Morris, Extrapolation methods for vector sequences, Numer. Math. 61, pp. 475-487, 1992.

/3/ E. J. Weniger, Nonlinear Sequence Transformations: Computational Tools for the Acceleration of Convergence and the Summantion of Divergent Series, URL:
http://arxiv.org/pdf/math.CA/0107080

/4/ X. Gourdon, P. Sebah, Convergence acceleration of series, January 10, 2002, URL:
http://numbers.computation.free.fr/Constants/Miscellaneous/seriesacceleration.html

/5/ K. S. Kundert, J. K. White, A. Sangiovanni-Vincentelli, Steady-state methods for simulation analog and microwave circuits, Kluwer Academic Publishers, 1990.

/6/ SPICE OPUS circuit simulator homepage, URL:
http://www.fe.uni-lj.si/spice/
Faculty of Electrical Engineering, Electronic Design Automation Laboratory, URL:
http://www.fe.uni-lj.si/edalab/

/7/ J. Puhan, T. Tuma, Optimization of analog circuits with SPICE 3f4, Proceedings of the ECCTD '97, vol. 1, pp. 177-180, 1997.

/8/ B. Wagner, A. Bürmen, J. Puhan, I. Fajfar, T. Tuma, Computing the steady-state response of nonlinear circuits by means of the ε-algorithm, Electrotechnical review, vol. 72, no. 5, pp. 297-302, 2005.

/9/ B. Wagner, A. Bürmen, J. Puhan, I. Fajfar, T. Tuma, Uporaba ekstrapolacijskih metod pri računanju stacionarnega stanja električnih vezij (Application of extrapolation methods for steady-state computation of electrical circuits), Zbornik štirinajste mednarodne Elektrotehniške in računalniške konference ERK 2005, zv. A, str. 82-85.

/10/ BÜRMEN, Arpad, STRLE, Drago, BRATKOVIČ, Franc, PUHAN, Janez, FAJFAR, Iztok, TUMA, Tadej. Automated robust design and optimization of integrated circuits by means of penalty functions. AEÜ, Int. j. electron. commun. 2003, 57, no. 1, str. 47-56.

univ. dipl. ing. el. Borut Wagner

University of Ljubljana

Faculty of Electrical Engineering

Tržaška cesta 25, SI-1000 Ljubljana, Slovenia

E-mail: borut.wagner@fe.uni-lj.si

Tel: +386 1 4768724

Fax: +386 1 4264630

doc. dr. Árpád Bűrmen

University of Ljubljana

Faculty of Electrical Engineering

Tržaška cesta 25, SI-1000 Ljubljana, Slovenia

E-mail: arpadb@fides.fe.uni-lj.si

Tel: +386 1 4768322

Fax: +386 1 4264630

doc. dr. Janez Puhan

University of Ljubljana

Faculty of Electrical Engineering

Tržaška cesta 25, SI-1000 Ljubljana, Slovenia

E-mail: janez.puhan@fe.uni-lj.si

Tel: +386 1 4768322

Fax: +386 1 4264630

prof. dr. Sašo Tomažič

University of Ljubljana

Faculty of Electrical Engineering

Tržaška cesta 25, SI-1000 Ljubljana, Slovenia

E-mail: saso.tomazic@fe.uni-lj.si

Tel: +386 1 4768432

Fax: +386 1 4264630

prof. dr. Tadej Tuma

University of Ljubljana

Faculty of Electrical Engineering

Tržaška cesta 25, SI-1000 Ljubljana, Slovenia

E-mail: tadej.tuma@fe.uni-lj.si

Tel: +386 1 4768329

Fax: +386 1 4264630