Communications to SIMAI Congress,
ISSN 1827-9015, Vol. 2 (2007)

DOI: 10.1685/CSC06152

Model reduction for nonlinear differential-algebraic equations

T. Voß*

Delft Center of Systems and Control, Delft University of Technology, Mekelweg 2
Delft, NL-2611 MP, The Netherlands
**E-mail: t.voss@tudelft.nl*

Abstract.

In this paper we extend the Trajectory Piecewise Linear (TPWL) model order reduction (MOR) method for nonlinear differential algebraic equations (DAE). The TPWL method is based on combining several linear reduced models at different time points, which are created along a typical trajectory, to approximate the fully nonlinear model.
We discuss how error control is used to select the linearisation tuples for linearisation, the choice of the linear MOR method and how we can create a globally reduced subspace. Then we study how to combine the locally linearised reduced systems to create a global TPWL model. Finally, we discuss a numerical result of the TPWL method.

Keywords: Model order reduction, TPWL, nonlinear DAE, PMTBR, circuit simulation

1. Introduction

Nowadays a lot of circuits which are used in many fields are not only purely digital or analogue. These circuits are a mixture of analogue and digital and are called mixed-signal circuits. For developing these large circuits there is a need of tools which can simulate these circuits efficiently during the design phase as well as during the verification phases, i.e. also offering accuracy, robustness and different parameters.

Digital circuits behave nonlinear with respect to the source/input values. However, the time-varying behavior may show several smooth periods. But the behavior changes rapidly when there is an important change in the inputs. The digital part in mixed-signal designs contains also several sub-circuits that are reused several times. The only difference between these parts is that they have different inputs. So simplifying these circuits could give a speed up for the transient analysis.

To do this we could use MOR methods, which are based on linear or quadratic reduction[3] or nonlinear methods, e.g. proper orthogonal decomposition (POD[5]). However, these methods are mostly developed for weakly nonlinear systems. This makes these methods not so useful in circuit simulation, which often deals with highly nonlinear circuits. To overcome this issue, a Trajectory Piecewise Linear (TPWL)[4] approach for ordinary differential equations (ODE) was developed. We will show how we can adapt these method to DAEs.

In the next section we present our TPWL approach for nonlinear DAEs. In Section 3 we show how the method performs in practice. Finally in Section 4 we draw our conclusions.

Licensed under the Creative Commons Attribution Noncommercial No Derivatives

2. Trajectory Piecewise Linear Model Order Reduction

In this section we discuss how we can apply the TPWL method to a nonlinear DAE which is used to describe the dynamical behavior of a circuit. The DAE system to which we want to apply the TPWL method is

$$\frac{d}{dt}\mathbf{q}(t,\mathbf{x}) + \mathbf{j}(t,\mathbf{x}) + B\mathbf{u}(t) = \mathbf{0}, \mathbf{x}(0) = \mathbf{x}_0$$

where $\mathbf{q}, \mathbf{j} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$, $B \in \mathbb{R}^{n\times m}$ and $\mathbf{u} : \mathbb{R} \to \mathbb{R}^m$. Here $\mathbf{q}$ represents the capacitance and the inductance, $\mathbf{j}$ the resistances while B is the input distribution matrix and $\mathbf{u}$ is the given input for the circuit.

The idea behind the TPWL method is to linearize the system at special time points t_i along a typical trajectory. The trajectory itself should represent the full nonlinear behaviour of the system. Then we reduce each locally linearized system with a linear model reduction technique and store the basis of each locally reduced subspace S_i. With the help of the S_i we compute a globally reduced subspace S. S is then used as the subspace for all locally linearized system. The final TPWL model is a weighted sum of all locally linearized reduced systems. The TPWL model can then be solved by a standard DAE time integrator, e.g. a backward differential formula (BDF). In the following subsection we show how we apply the described steps.

2.1. *Creating the locally linearized models*

The disadvantage of the standard linearization methods is that we can only trust the results if the solutions stays close to the linearization tuple (LT), time and space, around which we have created the linearized model. To overcome this disadvantage the idea is to take several linearized models to create the TPWL model. These LTs will be taken along a trajectory which represents the typical behavior of that system. If we do this we can trust the results as long as the solution stays close to one of the LT. Figure 1 on page 3 illustrates a typical situation. In this situation we have 5 LTs $(\tilde{\mathbf{x}}_0, \ldots, \tilde{\mathbf{x}}_4)$, which are created along trajectory **A**, and their related accuracy region (the gray region). We can see that trajectory **B** and **C** stay in the accuracy region also if they have different inputs **B** or different initial values **C**. And as long as the trajectories stay in the accuracy region we can also be sure that we have a good approximation to the original system also with different inputs an/or initial values.

We will discuss an approach which will chose as many LTs as needed to reach a given accuracy and as less as possible to get the maximum speed up in the TPWL model.

To get the LTs we need a nonlinear solution of the original model, e.g. created by a backward differential formula (BDF) approach, but only with a low accuracy because it is enough if the LTs are close to the exact trajectory. The reason why we need only a low accuracy is that we just want that the accurate, exact trajectory stays in the accuracy region of all LTs. Because the accuracy region is normally relatively large it is not important that the LT is the exact solution. With this in mind we see that it is a good idea to include the selection of the LTs directly in such a method.

Similarly to a step size controller, we have the problem that accuracy of the actual LT depends on the future LTs, because to calculate the global subspace we use *all* LTs we create along the typical trajectory. So we can only make local accuracy assumptions. Therefore we use a quite simple strategy for selecting a new LT.

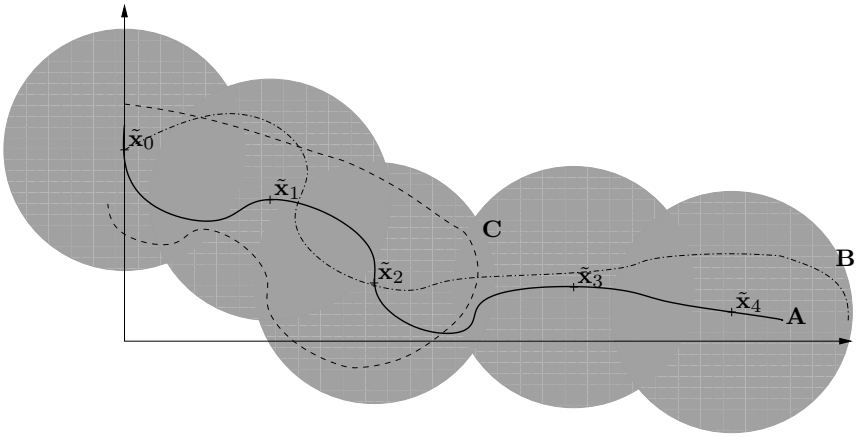


Fig. 1. Creating the linearization tuples, and curves with different sources and/or initial values

Algorithm 2.1 Linearization tuple controller

1. Set an accuracy factor $\varepsilon > 0$
2. Linearize the system around the last (i-th) LT $(\tilde{\mathbf{x}}_i, t_i)$. So we get

$$C_i\dot{\mathbf{x}} + G_i\mathbf{x} + B_i\mathbf{u}(t) = 0$$

   where $C_i = \frac{\partial \mathbf{q}}{\partial \dot{\mathbf{x}}}(\tilde{\mathbf{x}}_i, t_i) \in \mathbb{R}^{n\times n}$, $\det C_i = 0$ and $G_i = \frac{\partial \mathbf{j}}{\partial \mathbf{x}}(\tilde{\mathbf{x}}_i, t_i) \in \mathbb{R}^{n\times n}$. Save C_i, G_i and B_i.
3. Reduce the i-th linearized system to dimension $r \ll n$ with a linear model reduction method, e.g. 'Poor Mans' TBR (PMTBR)[2] or a Krylov approach[1] , and project the system to this locally reduced subspace which is spanned by P_i.

$$C_i^r\dot{\mathbf{y}} + G_i^r\mathbf{y} + B_i^r\mathbf{u}(t) = 0$$

   where $C_i^r = P_i^\top C_i P_i$, $G_i^r = P_i^\top G_i P_i$ and $B_i^r = P_i^\top B$ with $P_i \in \mathbb{R}^{r_i\times n}$. $\mathbf{y} \in \mathbb{R}^r$ is the approximation to $\mathbf{x}$ with $\mathbf{x} \approx P_i\mathbf{y}$. Save P_i.
4. Simulate both the locally linearized reduced system with $\mathbf{y}_0 = P_i^\top\tilde{\mathbf{x}}_i$ and the original nonlinear system with a step size determined from the original nonlinear system. If at t the absolute distance between the two solutions $\frac{||P_i\mathbf{y}-\mathbf{x}||}{||\mathbf{x}||}$ becomes bigger than ε we set the $(i+1)$-th LT to $(\mathbf{x}, t)$ and go to step 2.

From the overview we know that the final TPWL model consists of a weighting of several reduced linearized systems. The basis for the globally reduced subspace is created by merging all locally reduced subspaces. This is done in such a way that the globally reduced subspace represents the most dominant parts of the locally reduced subspaces. So a good approximation for the global subspace is then just the actual locally reduced subspace. With this in mind we just create the locally reduced subspace with a linear model reduction technique, we have to do this anyway for the construction of the globally reduced subspace, and we use this to project the local linearized subspace. Next we simulate both systems, the original and the locally linearized reduced system, until the distance between both systems is bigger then a given bound. At his point we then set a new LT. Algorithm 2.1 on this page shows the procedure to find LT $i+1$. We continue with this procedure until we have reached the end of the given trajectory.

Algorithm 2.2 Creating the globally reduced subspace

1. Define $\tilde{P} = [P_1, \ldots, P_p]$.
2. Calculate the SVD of $\tilde{P}$. So $\tilde{P} = U\Sigma V^\top$ with $U = [u_1, \ldots, u_n] \in \mathbb{R}^{n\times n}$,$\Sigma \in \mathbb{R}^{n\times rp}$ and $V \in \mathbb{R}^{rp\times rp}$.
3. Define P as $[u_1, \ldots, u_r]$.
4. Create the p locally linearized reduced systems given as $C_{ir}\dot{\mathbf{y}} + G_{ir}\mathbf{y} + B_{ir}\mathbf{u}(t) = \mathbf{0}$ with $C_{ir} = P^\top C_i P$, $G_{ir} = P^\top G_i P$ and $B_{ir} = P^\top B_i$

An extension to this approach is to calculate several typical trajectories to create a bigger accuracy region. However the more LTs we have, the more memory for saving the TPWL model we need, and the more involving the weighting procedure will be.

2.2. *Creating the globally reduced subspace*

After we have created p linearized systems and the p related locally reduced subspaces we have to construct the globally reduced subspace. The reason why we need a globally reduced subspace is that we want a smooth transition from one accuracy region to another accuracy region while solving the TPWL model. If we would have for each local subspace a separate reduced subspace the transition from on the another subspace would be way to difficult.

Let us assume we have p locally reduced subspaces which are spanned by $P_i \in \mathbb{R}^{n\times r_i}, i = 1, \ldots, p$. P_i is the optimal reduced subspace for the i-th locally linearized system. The columns of P_i span the i-th subspace, so one idea is to create a new matrix $\tilde{P}$ which contains all columns of the P_i's. So $\tilde{P} := [P_1, \ldots, P_p]$ spans then the union of all reduced subspaces. Of course the columns of $\tilde{P}$ are in general linearly dependent and also the number of columns is in general larger then n, so $\tilde{P}$ is not a good global projection matrix. It is even high likely that several P_i are quite similar because the linearized systems are also. Hence the columns of these P_i are quite dominant in the matrix $\tilde{P}$. To extract the span of the most dominant columns of $\tilde{P}$, and so the most dominant part of the union of the locally reduced subspaces, we use a singular value decomposition (SVD) of $\tilde{P} = U\Sigma V^\top$. Then U contains the most dominant columns of $\tilde{P}$, and so from all P_i, ordered by their importance. As the globally reduced subspace we take the one which is spanned by the first r columns of U. With this globally reduced subspace we can establish a smooth transition from one local system to another one. Summing up we get the Algorithm 2.2.

2.3. *Creating the TPWL reduced order model by weighting*

Now we have p locally linearized reduced systems which lay all in the same globally reduced subspace, but we still need to combine them to get a global TPWL model. We do this by calculating a weighted sum of local models

$$\sum_{i=1}^{p} w_i(\mathbf{y}) \left(C_{ir}\dot{\mathbf{y}} + G_{ir}\mathbf{y} + B_i\mathbf{u}(t)\right) = 0.$$

To see how we should choose the weights we take a look to a simple example, see Figure 2. In this example we have 3 LTs $\mathbf{x}_0, \mathbf{x}_1$ and $\mathbf{x}_2$ and the related accuracy region, shown

as circles. We also have 3 possible trajectory points of the TPWL model. $\mathbf{y}_0$ lays only in the accuracy region of $\mathbf{x}_1$ so the related local system should have the biggest influence to the TPWL model. Hence we should choose $w_1 \approx 1$ and $w_0,\ w_2 \approx 0$. If we look to $\mathbf{y}_1$ we see that this point lays in the accuracy region related to $\mathbf{x}_1$ and $\mathbf{x}_2$ so we should take a combination of both local models this means that we should choose the weights as following: $w_1 + w_2 \approx 1$ and $w_3 \approx 0$. For $\mathbf{y}_2$ we have the situation that the solution has left *all* accuracy regions so we should stop the simulation at this point or give at least a warning.

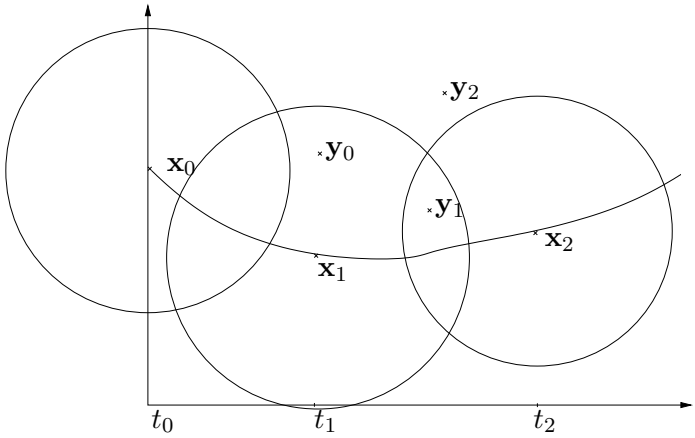


Fig. 2. Simple TPWL model

A template for a weighting procedure is described in Algorithm 2.3.

Algorithm 2.3 Weighting template

given p LTs $(t_{l_i}, \mathbf{y}_{l_i})$, $i = 0, \ldots, p-1$ and $b = 0$
for $i = 0$ to $p - 1$
    if $\mathbf{y} \in \mathcal{B}((t_{l_i}, \mathbf{y}_{l_i}), (\delta_t, \delta_{\mathbf{y}}))$
        $0 \ll w_i \leq 1.$
        $b = 1$
    else
        $0 \leq w_i \ll 1$
    end
end
if $b = 0$
    Create warning
end
Such that $\sum_{i=0}^{p-1} w_i = 1$

After calculating the weights we normalize them to get a convex combination of the locally linearized reduced systems. If we choose the weight in the way as described in the example we get a distance-depending weighting scheme as shown in Algorithm 2.4.

There also is an extended approach which uses instead of the distance an approximation of the linearization error to calculate the weights.[6] This approach is more complex because we have to compute an estimate of the Hessian's of $\mathbf{q}$ and $\mathbf{j}$, but it produces an even better TPWL model.

Algorithm 2.4 Distance dependent weights

Given actual state $\mathbf{y}$, actual time t, p LTs $(t_{l_i}, \mathbf{y}_{l_i})$ and $\alpha_{\mathbf{y}}, \alpha_t \geq 0$ with $\alpha_{\mathbf{y}} + \alpha_t = 1$

1. For $i = 0, \ldots, p-1$ compute $d_i = \alpha_{\mathbf{y}} \|\mathbf{y} - \mathbf{y}_{l_i}\| + \alpha_t |t - t_{l_i}|$
2. For $i = 0, \ldots, p-1$ calculate $\tilde{w}_i = e^{-\frac{d_i \beta}{m}}$ with $m = \min_{i=0,\ldots,p-1} d_i$, $\beta > 0$
3. Normalize the weights such that the given constraints hold
   $w_i = \frac{\tilde{w}_i}{s}$ with $s = \sum_{i=0}^{p-1} \tilde{w}_i$

3. Numerical results

We will show the performance of TPWL method for two examples. These examples are rather academic, but they demonstrate the performance of the proposed method.

3.1. *Chain of inverters*

As a test circuit we have chosen a chain of inverters, which consists of 100 inverters that are connected in series. The circuit behaves nonlinearly so it is a good test for the TPWL method. A schematic sketch of the circuit can be seen in 1. Also we have dependencies between all nodes which is also not an optimal behavior for a model reduction process. The DAE which is describing the dynamics of the circuit has 104 states (voltages at nodes, and two currents through the voltage sources). For selecting the LTs we have used Algorithm 2.4. For the linear model reduction technique we used PMTBR[2] , that was adapted to deal with our DAE. In Figure 2 we see the results for two different test setups and in Table 1 we sum up the speed up for the simulation with the same input as the training input.

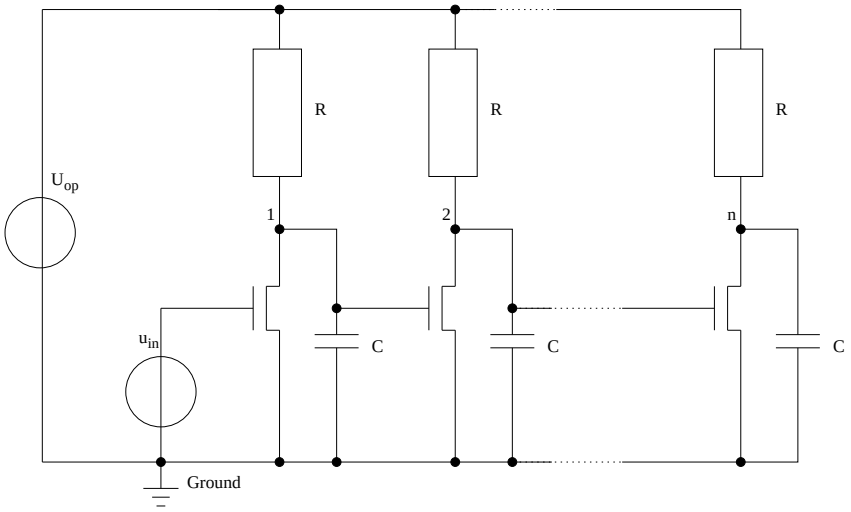


Fig. 1. Schematic sketch of the chain of inverters circuit

In the first test setup we use the same input for testing as for creating the PMTBR-TPWL model. It can be seen that the relative error is most of the time lower then the given error bound. For all orders we have to use the same number of LTs (62), which comes from the fact that the local systems only need relatively small subspaces to get

DOI: 10.1685/CSC06152

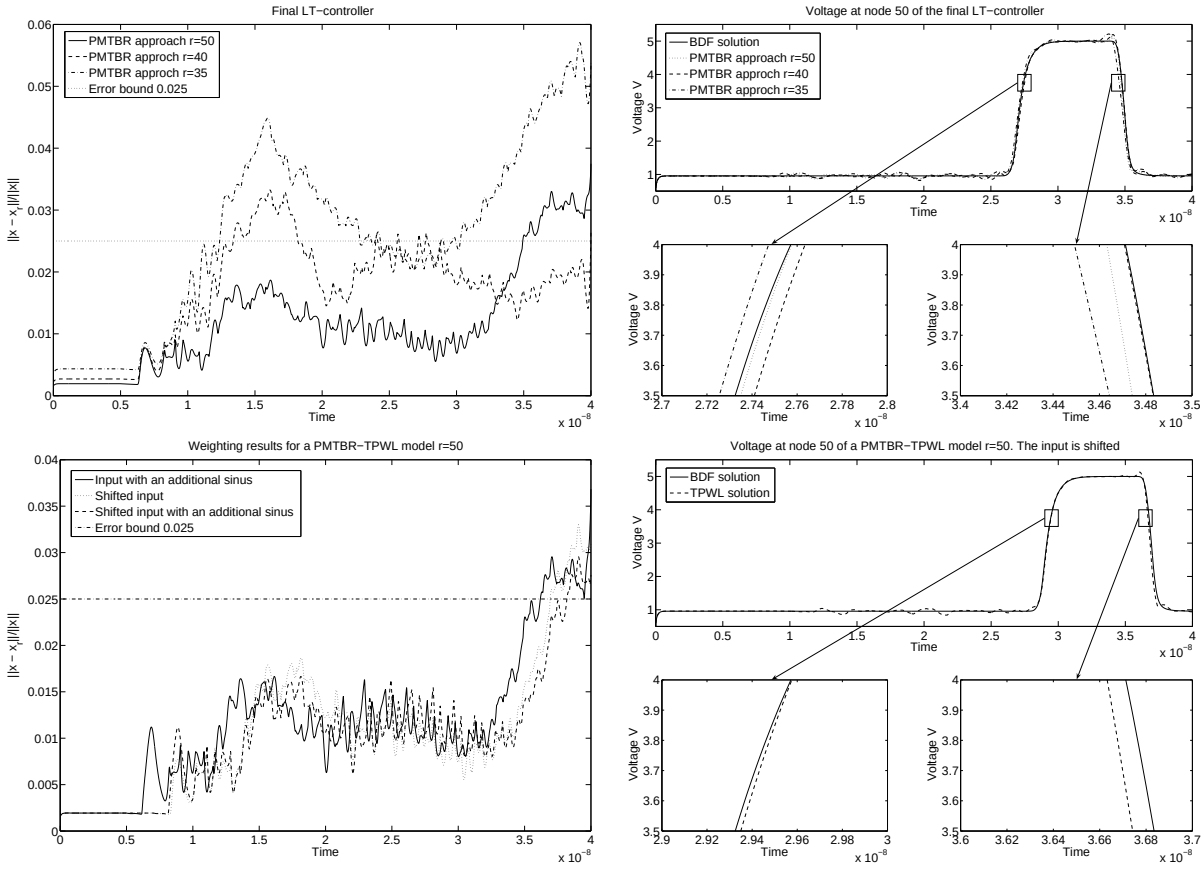


Fig. 2. Relative error of the PMTBR-TPWL method for different orders (top) and inputs (bottom)

the desired accuracy. The resulting speed up is between 5.4 and 8.3 compared to a BDF method, which needs 220s.

In the second setup we use test inputs which differ from the input we have used to create the PMTBR-TPWL model. We use for example an input which is shifted in time or which has an extra sinus wave added. The result shows that the TPWL method can handle also inputs which differ from the training input as long as the new input stays close to the original input.

3.2. *Chain of Diodes*

Next we considered the following circuit model which consists of $N = 300$ diodes, where $I_s = 10^{-14}$ A, $V_T = 0.0256$ V, $C = 10^{-12}$ F, $R = 10^4$ Ω.

$$\begin{aligned}
V_1 - U_{\text{in}}(10^9 t) &= 0, \\
i_E - g(V_1, V_2) &= 0, \\
g(V_1, V_2) - g(V_2, V_3) - C\dot{V}_2 - \tfrac{1}{R}V_2 &= 0, \\
&\vdots \\
g(V_{N-1}, V_N) - g(V_N, V_{N+1}) - C\dot{V}_N - \tfrac{1}{R}V_N &= 0, \\
g(V_N, V_{N+1}) - C\dot{V}_{N+1} - \tfrac{1}{R}V_{N+1} &= 0,
\end{aligned}
\qquad
g(V_a, V_b) = \begin{cases} (I_s e^{\frac{V_a - V_b}{V_T}} - 1) & \text{if } V_a - V_b > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

$$U_{\text{in}}(t) = \begin{cases} 20 & \text{if } t \le 10 \\ 170 - 15t & \text{if } 10 < t \le 11 \\ 5 & \text{if } t > 11 \end{cases}$$

To test the performance of the PMTBR-TPWL model we use the same input as we used for creating the reduced order model. We see again that the relative error is most of the time lower then the given error bound. Additionally we can see that for a lower number of LTs (42) we need higher order models (100) and that with a higher number of

	org.	red.	red.	red.	org.	red.	red.	red.
Problem	3.1	3.1	3.1	3.1	3.2	3.2	3.2	3.2
r	104	50	40	35	302	100	50	10
# LTs	-	62	62	62	-	42	50	60
Extr. time	-	240	236	233	-	206	285	290
Simul. time	220	41	31	27	142	2.3	1.5	1.1
max. error	-	3.7%	3.3%	5.7%	-	4.1%	4%	4%
Speed up	-	5.4	7.2	8.3	-	62	95	129
mean error	-	1.2%	1.7%	2.6%	-	2.1%	1.9%	1.9%

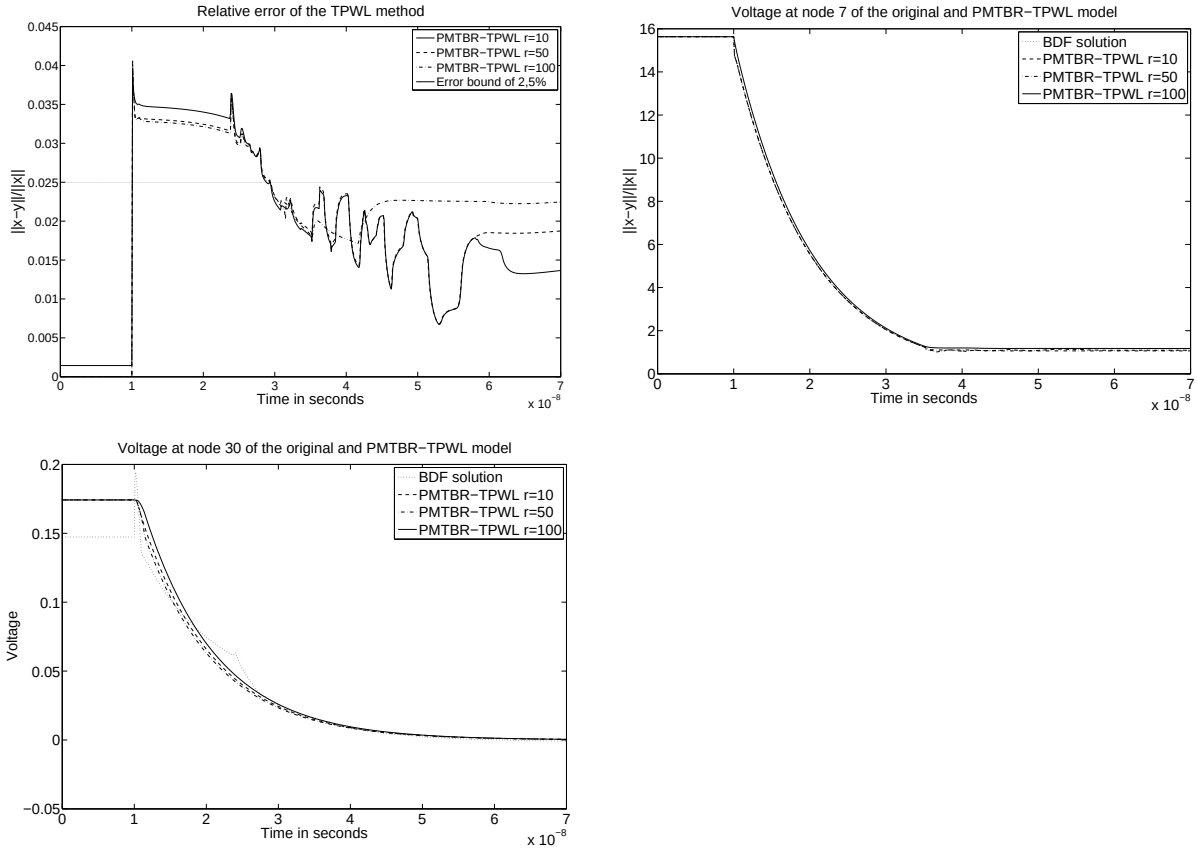


Fig. 3. Relative error and Voltages at specific nodes of the PMTBR-TPWL method

LTs (60) we can use lower order models. But it is also the other way around: the locally reduced systems are more accurate if we choose a high order. Then the accuracy region is (maybe) larger. So we need less LTs for high orders. The speed up for this circuit is much higher (between 62 and 129). An explanation for this is that the simulation of the original circuit needs a lot device evaluations which we do not need in the reduced order model.

4. Conclusion

The TPWL method, applied to nonlinear DAEs, which are used to describe circuits, is a promising technique to reduce the simulation time. It has several advantages compared to

other methods. First of all we can get a big speed up in simulation time, in our test factor 8.3, because we are only solving small linear systems to approximate our system. We can use the well-developed linear model reduction techniques to increase the performance of our methods. And we are also able to create a linearization tuple controller that can be used directly in a BDF method, which is a big advantage because we get a fast model extraction technique. And we even can improve the properties of the TPWL model if we construct a good weighting procedure. The last thing we want to mention is that the TPWL method also has the nice property that it is scalable. This means that by using different linearization tuple controllers, linear model reduction techniques and weighting methods, we can change the method from a fast method but not so accurate method to a slower but also much more accurate method. This means that the user can decide what he desires: Speed or accuracy.

Acknowledgment

Thanks go to Dr. E. Jan W. ter Maten, Dr. Theo Beelen and Dr. Bratislav Tasić, (Philips Research, now NXP semiconductors Research, Eindhoven, The Netherlands), Ir. Arie Verhoeven (Eindhoven University of Technologie, The Netherlands), Dr. Ahmed El Guennouni (Magma Design Automation, Eindhoven, The Netherlands), Prof. Dr. Michael Günther and Dr. Roland Pulch (University of Wuppertal, Germany), for stimulating discussion.

REFERENCES

1. L.T. Pileggi A. Odabasioglu, M. Celik. Prima: Passive reduced-order interconnect macromodeling algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(8):645–654, 1998.
2. J. Phillips and L.M. Silvera. Poor man's tbr: A simple model reduction scheme. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(1):43–55, 2005.
3. J.R. Phillips. Projection frameworks for model reduction of weakly nonlinear systems. In *DAC*, Los Angeles, California, 2000.
4. M.J. Rewieński. *A trajectory piecewise-linear approch to model order reduction of nonlinear dynamical systems*. PhD thesis, Massachutes Institute of Technology, MA, USA, june 2003.
5. S. Volkwein. Proper orthogonal decomposition and singular value decomposition, sfb-preprint no. 153, 1999.
6. T. Voß. Model order reduction for nonlinear differential algebraic equations in circuits simulation. Master's thesis, Bergische Universität Wuppertal, Germany, 2005.