Trajectory Piecewise Linear Approach for Nonlinear Differential-Algebraic Equations in circuit simulation

T. Voß[1], R. Pulch[2], E.J.W. ter Maten[3], A. El Guennouni[4]

[1]University of Groningen,Faculty of Mathematics and Natural Sciences, Nijenborgh 4, 9747 AG Groningen, The Netherlands, t.voss@rug.nl
[2]Bergische Universitt Wuppertal, Fachbereich Mathematik, Gaustr. 20, 42119 Wuppertal, Germany, pulch@math.uni-wuppertal.de
[3]Philips Semiconductors, High Tech Campus 48, 5656 AE Eindhoven, The Netherlands, jan.ter.maten@philips.com
[4]Magma Design Automation, Eindhoven, The Netherlands

Summary. 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 full nonlinear model.
We discuss how to select the linearization tuples for linearization and the choice of linear MOR method. Then we study how to combine the local linearized reduced systems to create a global TPWL model. Finally, we show a numerical result.

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. The digital part in mixed-signal designs contains also several sub-circuits that are reused several times. 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 [PH00] or nonlinear methods, e.g. proper orthogonal decomposition (POD) [VOL99]. However, these methods are mostly developed for weakly nonlinear systems. Therefore these methods are not so useful in circuit simulation, which often deals with highly nonlinear circuits. To overcome this issue, a TPWL [REW03] approach for ordinary differential equations (ODE) was developed. We will show how we can adapt this method to DAEs. We handle here only DAEs of index 1, because a large number of circuits can be modeled with an index 1 DAE.
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.

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 we want to reduce is

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

where $\mathbf{q}, \mathbf{j} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$, $\tilde{B} \in \mathbb{R}^{n \times m}$ and $\tilde{\mathbf{u}} : \mathbb{R} \to \mathbb{R}^m$. Here $\mathbf{q}$ represents contributions from capacitances and the inductances, $\mathbf{j}$ represents contributions from resistances while $\tilde{B}$ is the input distribution matrix and $\tilde{\mathbf{u}}$ 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 behavior 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 systems. 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. In the following subsection we show how we apply the described steps.

2.1 Creating the local linearized models

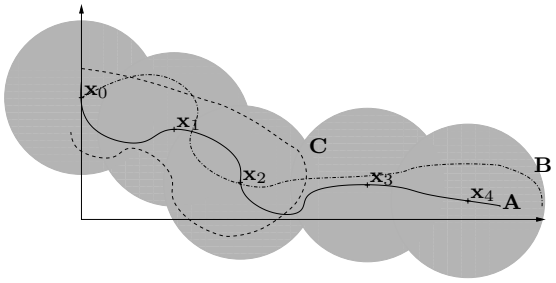

Fig. 1: Creating the linearization tuples, and curves with different sources and/or initial values

The disadvantage of the standard linearization methods is that we can only trust in the results if the solutions stays close to the linearization tuple (LT), time and solution space, around which we have created the linearized model[1]. To overcome this drawback 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 in the results as long as the solution stays close to one of the LT. Figure 1 illustrates a typical situation. In this situation we have 5 LTs $(x_0, \ldots, x_4)$, 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 even 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 of the original system.
We will discuss an approach which will choose as many LTs as needed to reach a given accuracy and as few as possible to get the maximum speed-up in the TPWL model. In [REW03] they propose more accurate methods for selecting the LTs, but these methods are only applicable to ODEs, because they assume that $\frac{\partial \mathbf{q}}{\partial \mathbf{x}}(\mathbf{x}_0, t_0)$ is regular, but since we handle DAEs so $\frac{\partial \mathbf{q}}{\partial \mathbf{x}}(\mathbf{x}_0, t_0)$ is singular.
To get the LTs we need a solution of the original model. But we only need low accuracy of this solution because it is enough if the LTs are close to the exact trajectory. The reason why we need only low accuracy is that we just need the LTs to stay close to the exact trajectory. With this in mind, we see that it is a good idea to include the selection of the LTs directly in a solver of a nonlinear DAE.
Similarly to a step size controller, the accuracy of the actual local reduced model depends on the selection of the future LTs. The reason for this is that for calculating the global reduced subspace we use *all* local subspaces we have created along the trajectory, also future ones. In consequence, we can only make local accuracy assumptions and so, we use a quite simple strategy for selecting a new LT.
From the overview we know that the final TPWL model consists of a weighting of several reduced linearized systems. We create the reduced linearized systems by projecting all of them into the same reduced global subspace. The basis for the global reduced subspace is created by merging all locally reduced subspaces which we got during the creation of the linearized

[1] For simplicity we sometimes omit the time dependency of the LT

models. This is done in such a way that the global reduced subspace represents the most dominant parts of the locally reduced subspaces. So a good approximation for the globally reduced subspace is then just the actual locally reduced subspace. This means that we create the local reduced subspace with a linear model reduction technique and we use this subspace to create the local linearized subspace. Next we simulate both systems, the original and the local linearized reduced system, until the distance between the solutions of both systems is bigger then a given bound. At this point we set a new LT. Algorithm 1 shows the procedure to find LT $i+1$. We continue with this procedure until we have reached the end of the given

Algorithm 1 Linearization tuple controller

1. Set an accuracy factor $\varepsilon > 0$
2. Linearize the system around the last (i-th) LT (x_i, t_i). So we get
$$C_i\dot{\mathbf{x}} + G_i\mathbf{x} + B\mathbf{u}(t) = 0$$
where $C_i = \frac{\partial \mathbf{q}}{\partial \mathbf{x}}(\mathbf{x}_i, t_i) \in \mathbb{R}^{n\times n}$, $\det C_i = 0$ and $G_i = \frac{\partial \mathbf{j}}{\partial \mathbf{x}}(\mathbf{x}_i, t_i) \in \mathbb{R}^{n\times n}$. Save C_i, G_i and B.
3. Reduce the i-th linearized system to dimension $r_i \ll n$ with a linear model reduction method, e.g. 'Poor Man's TBR (PMTBR) [PS05], and project the system to this locally reduced subspace which is spanned by P_i. Be aware of that r_i can be different for all different LTs
$$C_i^r\dot{\mathbf{y}} + G_i^r\mathbf{y} + B^r\mathbf{u}(t) = 0$$
where $C_i^r = P_i^\top C_i P_i$, $G_i^r = P_i^\top G_i P_i$ and $B^r = P_i^\top B$ with $P_i \in \mathbb{R}^{n\times r_i}$. $\mathbf{y} \in \mathbb{R}^{r_i}$ 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 $\mathbf{y}_0^i = P_i^\top \mathbf{x}_i$ and the original system with a step size determined from the original nonlinear system. If at t the relative 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.

trajectory.
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 global reduced subspace

After we have created p linearized systems and p related local reduced subspaces we have to construct the global reduced subspace. We need a global reduced subspace because we want a smooth transition from one accuracy region to another accuracy region while solving the TPWL model. If we had for each local subspace a separate reduced subspace the transition from one to another subspace would be way too difficult.
Let us assume we have p local reduced subspaces which are spanned by $P_i \in \mathbb{R}^{n\times r_i}, i = 0, \ldots, p-1$. The columns P_i span the optimal reduced subspace for the i-th local linearized system. 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 than 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 most dominant columns of $\tilde{P}$, and so the most dominant part of the union of the local 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 global reduced subspace we take the one which is spanned by the first r columns of U. With this global reduced subspace we can establish a smooth transition from one local system to another one. Summarizing we obtain Algorithm 2.

Algorithm 2 Creating the global 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 local 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 $B_{ir} = P^\top B$

2.3 Creating the TPWL reduced order model by weighting

Now we have p locally linearized reduced systems which are all lying in the same global 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

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

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. $\mathbf{y}_0$ lies 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 lies 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 following situation: the solution has left *all* accuracy regions so we should stop the simulation at this point or give at least a warning. A template for a weighting procedure is described in Algorithm 3.

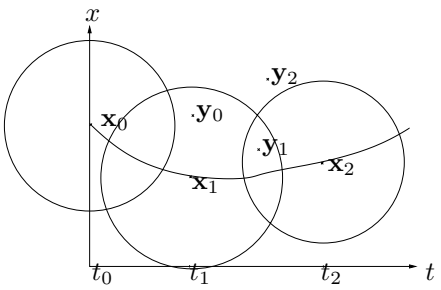

Fig. 2: Simple TPWL model

After calculating the weights we normalize them to get a convex combination of the local linearized reduced systems. If we choose the weight in the way described in the example we get a distance depending weighting scheme as shown in Algorithm 4.

There also is an extended approach which uses instead of the distance an approximation of the linearization error to calculate the weights [VO05]. This approach is more complex because we have to compute an estimate of the Hessian's of $\mathbf{q}$ and $\mathbf{j}$ but in this way we obtain an even better TPWL model.

To summarize we have changed, compared to [REW03], the way how the LTs are chosen. We also tried several linear model order reduction techniques to see which performs the best. Additionally we tried to improve the weighting procedure.

Algorithm 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 $\mathbf{y}$ lies in the accuracy region of the i-th LT
    $0 \ll w_i \leq 1$, $b = 1$
  else
    $0 \leq w_i \ll 1$
  end
end
if $b = 0$
  Create warning
end
Such that $\sum_{i=0}^{p-1} w_i = 1$

Algorithm 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 Example

We show how the TPWL method performs on a practical example. As a test circuit we have chosen a chain of inverters, which consists of 100 inverters, connected in series. The circuit behaves nonlinearly so it is a good test for the TPWL method. Additionally, 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. For selecting the LTs we have used Algorithm 4. For linear model reduction technique we used PMTBR [PS05], that was adapted to deal with our DAE.
In Figure 3 we illustrate the results of our test setup. The upper picture shows the relative error of the TPWL model compared to a highly accurate solution, calculated with a backward differential formula (BDF) method. The lower one compares the solution for the voltage of node 50 of the TPWL method to the solution of the BDF method. As input for both figures we used a slightly different input than for training the TPWL model. We used an input signal with a added sinus wave and/or delayed input signal.
In Table 1 we sum-up the speed-up for the simulation with the same input as the training input. *Extr. time + BDF* is the time it needs to create a TPWL model including the time for BDF method. *Simul. time* is the time needed to simulate the final TPWL model.

	r	# LTs	Extr. time + BDF	Simul. time	max. error	Speed-up	mean error
PMTBR	50	62	240s (220s BDF)	41s	0.037484	5.4	0.012051
PMTBR	40	62	236s (220s BDF)	31s	0.033233	7.2	0.016672
PMTBR	35	62	233s (220s BDF)	27s	0.057046	8.3	0.026128

Table 1: Final LT controller

It can be seen that the relative error is most of the time lower then the given error bound, see Figure 3. 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 the desired accuracy.

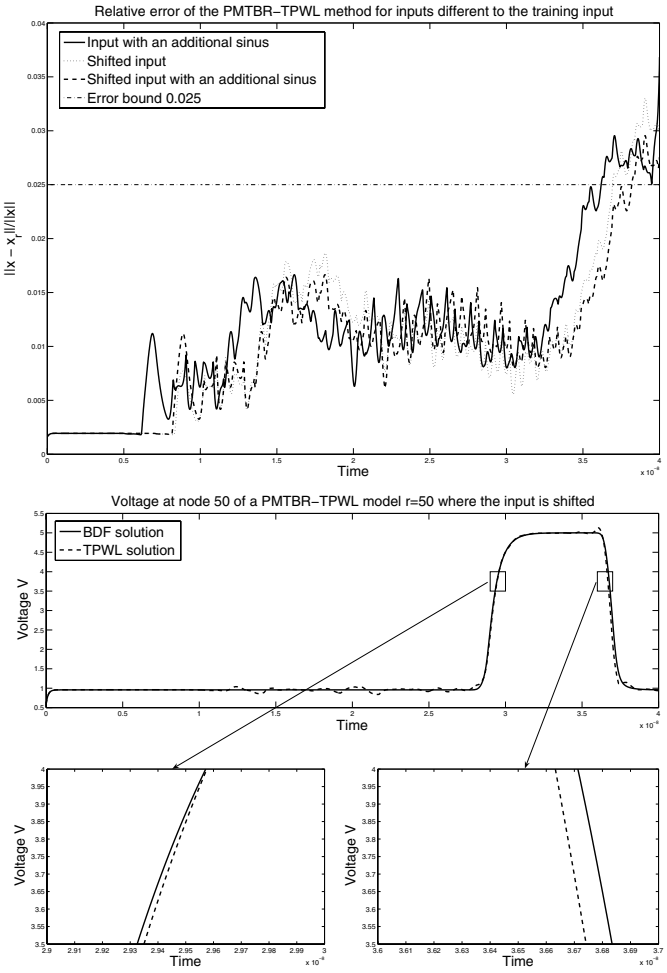

Fig. 3: Relative error of the PMTBR-TPWL method compared to high accurate BDF method (top) Voltage at node 50 compared to a high accurate BDF solution (bottom)

The resulting speed-up is between 5.4 and 8.3 compared to a BDF method, which needs 220s. This speed-up is related to the fact that we are solving linear reduced systems instead of the full nonlinear DAEs, which require a nonlinear solver including Newton iterations. The speed-up would be even bigger if we used a circuit with a lot of devices which have all to be evaluated. We can also notice that in our example the extraction time is quite high, since we have used a high accuracy for the BDF-method during the extraction. We can reduce this time if we use a lower accuracy during the extraction because we only need a rough approximation of the solution to create the TPWL model.
Even if we can reduce the extraction time using a lower accuracy during the extraction time, one can argue that a full simulation of the circuit is just a bit more expensive. This is of course correct but if we think about that we reduce parts of the circuit which are reused a lot, it is easy to see that we still get a quite large speed-up. So we propose to use the TPWL method in a kind of library approach.

4 Conclusion

The TPWL method, applied to nonlinear DAEs, 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. We can also use the well-developed linear model reduction techniques. And we are able to create a linearization tuple controller that can be used directly in a BDF method. In future we plan to apply this method to a more practical example, then we will also address open questions, e.g. what happens if the circuit contains a feed back. We also aim to improve the method for selecting the LTs and the weighting procedure and to quantify the speed-up obtained a priori.

Acknowledgment

Thanks go to Ir. Arie Verhoeven, University of Technology Eindhoven, The Netherlands and Prof. Dr. Michael Gnther, University of Wuppertal, Germany.

References

[PS05] Phillips, J., Silveira, L.M.: Poor Man's TBR: A simple model reduction scheme, IEEE transactions on computer-aided design of integrated circuits and systems. Vol. 14 No. 1, 2005

[PH00] Phillips, J.R.: Projection frameworks for model reduction of weakly nonlinear systems, DAC 2000, Los Angeles, California, 2000

[REW03] Rewieński, M.J.: A trajectory piecewise-linear approch to model order reduction of nonlinear dynamical systems, PhD Thesis, Massachusetts Institute of Technology, 2003

[VOL99] Volkwein, S.: Proper Orthogonal Decomposition and Singular Value Decomposition, SFB-Preprint No. 153, University of Graz, 1999

[VO05] Voß, T.: Model reduction for nonlinear differential algebraic equations, MSc. Thesis University of Wuppertal, 2005; Unclassified Report PR-TN-2005/00919, Philips Research Laboratories, 2005