Time dependent weight functions for the Trajectory Piecewise-Linear approach?

Juan Pablo Amorocho and Heike Faßbender

Abstract Model order reduction (MOR) has become an ubiquitous technique in the simulation of large-scale dynamical systems (i.e. 10^4 and more equations). One technique for non-linear MOR is the trajectory piecewise-linear approach (TPWL). TPWL approximates a non-linear differential system by a weighted sum of linear systems which have a significantly reduced number of equations. One open question is which weighting schemes are appropriate. We discuss whether a time dependent weighting scheme which is computationally faster as well as computationally cheaper than the originally proposed one is appropriate.

1 The Trajectory Piecewise-Linear approach (TPWL)

Model order reduction (MOR) speeds up the simulation of high dimensional dynamical systems by reducing its dimension while keeping the output error small. MOR for linear time-invariant systems is well understood and used in industry, however for non-linear systems there is still much research to do. The trajectory piecewise-linear approach [4] is a numerical method for the model order reduction of non-linear systems described by first-order, non-linear ordinary differential equations and non-linear differential-algebraic equations. Here we consider the ordinary differential equation:

Juan Pablo Amorocho D.
Institut *Computational Mathematics*, Technische Universität Braunschweig, D-38023 Braunschweig, e-mail: jamoroch@tu-braunschweig.de

Heike Faßbender
Institut *Computational Mathematics*, Technische Universität Braunschweig, D-38023 Braunschweig, e-mail: h.fassbender@tu-braunschweig.de

$$\frac{d}{dt}x(t) = f(x(t)) + Dv(t), \quad 0 \le t \le T, \quad x(0) = x_0, \tag{1a}$$

$$y(x(t)) = L^T x(t), \tag{1b}$$

where $x(t) \in \mathbb{R}^N$ is the state vector, $f : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ is a non-linear function, $D \in \mathbb{R}^{N\times M}$ is the input matrix, $v(t) \in \mathbb{R}^M$ is the input, $L^T \in \mathbb{R}^{P\times N}$ is the matrix mapping the state vector into the output, and $y(x(t)) \in \mathbb{R}^P$ is the output of the system. The TPWL reduces the high computational cost of the evaluation of the non-linearity by approximating equation (1a) through a non-linear convex combination of s different linear models around (t_i, x_i), $i = 0, \ldots, s-1$ where $x_i = x(t_i)$. Let $\hat{f}(x(t))$ be the first order Taylor approximation of $f(x(t))$ around x_i: $\hat{f}(x(t)) = f_i + F_i(x(t) - x_i)$, where $f_i = f(x_i)$ and $F_i \in \mathbb{R}^{N\times N}$ is the Jacobian matrix of $f(x(t))$ evaluated at x_i. With this, the piecewise approximation of system (1) is given by

$$\frac{d}{dt}\hat{x}(t) = \sum_{i=0}^{s-1} \hat{w}_i(\hat{x}(t))(F_i\hat{x}(t) + B_i u(t)) \tag{2a}$$

$$\hat{y}(\hat{x}(t)) = L^T\hat{x}(t) \tag{2b}$$

with the weighting functions $\hat{w}_i : \mathbb{R}^N \longrightarrow [0,1]$, and $\sum_{i=0}^{s-1} \hat{w}_i(x(t)) = 1$ for all $t \in [0,T]$, $B_i = [D, f_i - F_i x_i] \in \mathbb{R}^{N\times(M+1)}$, and $u(t) = [v(t), 1]^T \in \mathbb{R}^{M+1}$. The tuples (t_i, x_i) are found via a simulation of system (1a), see [4] for more details. The resulting trajectory is called the *training* trajectory. Next, linear model order reduction is applied to the system (2) as follows: Build the projector $\pi_q = V_q W_q^T$, such that $q \ll N$ and $V_q, W_q \in \mathbb{R}^{N\times q}, W_q^T V_q = I_q$ (I_q: identity matrix of dimension q). Next, use π_q to apply a Petrov-Galerkin projection to system (2). The resulting dynamical system is called the TPWL model of system (1) given by:

$$\frac{d}{dt}x_r(t) = \sum_{i=0}^{s-1} w_i(x_r(t))(F_i^r x_r(t) + B_i^r u(t)), \tag{3a}$$

$$y_r(x_r(t)) = (L^r)^T x_r(t), \tag{3b}$$

where $x_r(t) \in \mathbb{R}^q$, and $w_i : \mathbb{R}^q \longrightarrow [0,1]$, $\sum_{i=0}^{s-1} w_i(x_r(t)) = 1$ for all $t \in [0,T]$. Matrices F_i^r, B_i^r and L^r are given by $F_i^r = W_q^T F_i V_q, B_i^r = W_q^T B_i, (L^r)^T = L^T V_q$. Note that $x_r(t) = W_q^T\hat{x}(t)$. The use of the weights $w_i(x_r(t))$ in (3a) instead of $\hat{w}_i(\hat{x}(t))$ aims to reduce the cost of the computation of the weight function from $O(Ns)$ to $O(qs)$.

In case the non-linearity of (1a) is not too pronounced, the TPWL model (3a) can predict the evolution of the solution trajectory even for initial conditions and/or input signal different from the ones used to set up the TPWL model.

In [4] two algorithms to build the projector π_q are proposed: One that includes all linear models and another that only uses the linear model at the initial condition. The latter is faster than the former, but the former yields a better approximation than the latter. Furthermore, [4] uses Krylov subspace methods to build π_q. However this is not the only option, [7] employs balanced truncation methods, and [1] combines the proper orthogonal decomposition method (POD) together with the TPWL approach.

In our implementation we use the algorithm given in [3] using only the linear model at the initial condition and setting $W_q = V_q$ which yields an orthogonal projector.

In building the piecewise approximation (2) two questions arise: (i) how to select the linear models along the training trajectory? (ii) how to choose suitable weight functions? In [4] the extraction starts with the linear model found at the initial condition x_0 and continues to simulate and select linear models keeping the training trajectory covered by balls $\mathscr{B}^{\alpha}_{x_i}, i = 0,\ldots,s-1$ of constant radius α each centered at x_i. Starting at the linear model at x_0, a new linear model is selected at (t_i, x_i) when $x(t_i)$ does not longer belong to $\mathscr{B}^{\alpha}_{x_{i-1}}$. This process is continued until $x(T)$ is reached. Here we adopt the procedure from [4].

Concerning the second question, it is fair to say that suitable weight functions should adequately approximate the system's non-linearity by the resulting convex combination of the linearized models. In [4] the weights are given by

$$w_i(\widetilde{x}_r(t)) = \frac{e^{-\beta d_i/m}}{\sum_{i=0}^{s-1} e^{-\beta d_i/m}}, \quad i = 0,\ldots,s-1 \tag{4}$$

where $d_i = \|\widetilde{x}_r(t) - \widetilde{x}_i\|_2$, and $m = \min\limits_{i=0,\ldots,s-1} d_i$ and β is a positive parameter usually set to 25. This ensures that the weights change rapidly as $\widetilde{x}_r(t)$ evolves in the state space. Here, $\widetilde{x}_i$ are given by $[\widetilde{x}_0, \widetilde{x}_1, \ldots, \widetilde{x}_{s-1}] = [W^T x_0, W^T x_1, \ldots, W^T x_{s-1}]$, where either $W = W_q$ as in (3a) or W results from the orthonormalization of the columns of $\widetilde{W} = \{W_q, x_0, \ldots, x_{s-1}\}$ and $\widetilde{x}_r(t)$ is given by $\widetilde{x}_r(t) = W^T \hat{x}(t)$. In the latter case, in (3a) W is used instead of W_q. However, this increases the dimension of the reduced space which is undesirable.

An alternative to compute d_i without having to build W is to project $x_r(t)$ back to the full state space, i.e. $V_q x_r(t)$ and compute the d_i as $\|V_q x_r(t) - x_i\|_2$ [5]. We use this approach in our experiments. We refer to the reader to [2, 6] for other weight functions.

2 Time dependent weight functions

State-dependent weight function are quite expensive to evaluate. Therefore, one might want to consider weight functions which are cheaper to evaluate, e.g., weight functions that are only a function of time. The idea behind these weight functions is simple: At each t_i we define a piecewise weight function $w_i(t)$ as

$$w_i(t) = \begin{cases} w_i^{(l)}(t) & \text{if } t \in [t_{i-1}, t_i] \\ w_i^{(r)}(t) & \text{if } t \in [t_i, t_{i+1}] \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i = 0,\ldots,s-1, \tag{5}$$

where $w_0^{(l)}(t) = w_{s-1}^{(r)}(t) = 0$. Here, the functions $w_i^{(l)}(t)$ and $w_i^{(r)}(t)$ are cubic polynomials such that $w_i^{(l)}(t_i) = w_i^{(r)}(t_i) = 1$, $w_i^{(l)}(t_{i-1}) = \frac{d}{dt} w_i^{(l)}(t_{i-1}) = \frac{d}{dt} w_i^{(l)}(t_i) = 0$

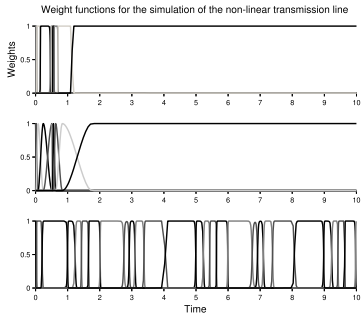

Fig. 1 Top: State-dependent weights (4) from the training input. Middle: Time-dependent weights (5) from training and test input. Bottom: state-dependent weights (4) from test input.

and $w_i^{(r)}(t_{i+1}) = \frac{d}{dt}w_i^{(r)}(t_i) = \frac{d}{dt}w_i^{(r)}(t_{i+1}) = 0$. If $t_{s-1} < T$, then we need an additional condition for w_{s-1}: $w_{s-1}(t) = 1, t \geq t_{s-1}$. With these cubic splines, (3a) approximates (1a) for each $t \in [t_i, t_{i+1}]$ by at most two linear differential equations,

$$\frac{d}{dt}x_r(t) = w_i^{(r)}(x_r(t))(F_i^r x_r(t) + B_i^r u(t)) + w_{i+1}^{(l)}(x_r(t))(F_{i+1}^r x_r(t) + B_{i+1}^r u(t)). \quad (6)$$

Figure 1 shows an example of time-dependent and state-dependent weights for the example considered in Section 3. The TPWL model constructed is simulated once for a training input (which was also used for simulating the original problem when setting up the TPWL model) and once for a test input. The time-dependent weight function does not change for different input functions, while the state-dependent one does change. As can be seen, considered as functions of time, in case of the training input, both weighting functions do have a similar behavior, the state-dependent weights are almost piecewise constant (either 0 or 1), while the time-dependent weight functions are smoother and allow (in the sense of the above equation) for a better overlap of the two linear models which live on the interval $[t_i, t_{i+1}]$. But for the test input, the state-dependent weights allow for a much stronger switch between the different linear models than the time-dependent weights do.

The time-dependent weights differ from those in [2, 4, 6] in several ways. First, the computation of the time-dependent weights is $O(s)$ instead of $O(qs)$, as the computation of the time dependent weights requires an evaluation of a scalar polynomial of degree 3 per time step. In contrast, in order to compute the weight functions (4) sums of vectors of dimension q have to be determined in every weight w_i per time step. Time-dependent weight functions not only speed up the TPWL, but also reduce the storage, as there is no need to build the basis W as described at the end of the previous section. Additionally, the use of these new weights makes the piecewise system a *linear* time dependent one as the weights are no longer state depended. This would enable the use of the theory of *linear* time dependent systems to analyze the TPWL model.

But (6) has a serious problem when the TPWL model is simulated with initial conditions and/or input signal different from the ones used to set up the TPWL

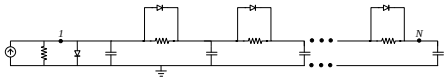

Fig. 2 Non-linear transmission line. [4]

model. In that case, there is no guarantee that, for instance, in the first time interval $[0,t_1]$ the linear models F_0 and F_1 are the best linear models to approximate the dynamics. It might be better to use, say, F_j and F_k. State-dependent weight functions might be able to allow for this choice, a time-dependent weight function can not achieve this, as it will always be local with respect to time. Therefore, the use of time-dependent weight functions is very limited, unless the TPWL model is build from several simulation runs with different parameter setting for the initial condition and input. In that case one might be able to modify the time-dependent weights approach above to be comparative to state-dependent weights.

3 Numerical Experiments

We have tested our weight representations on a non-linear transmission line [4] shown in Fig. 2. Applying the modified nodal analysis to this circuit yields the ordinary differential equation

$$\begin{aligned}
x_1'(t) &= -2x_1(t) + x_2(t) - i_{d_{1,0}} - i_{d_{1,2}} + v(t) \\
x_j'(t) &= x_{j-1}(t) - 2x_j(t) + x_{j+1}(t) + i_{d_{j-1,j}} - i_{d_{j,j+1}},\ j = 2,\ldots,N-1 \\
x_N'(t) &= x_{N-1} - x_N + i_{d_{N-1,N}},
\end{aligned}$$

where x_k is the kth node voltage and $i_{d_{k,l}}(t) = \exp(40(x_k(t) - x_l(t))) - 1$ is the diode's current between nodes k and l. $l = 0$ means the ground node.

We have simulated the above circuit using the TPWL with two different inputs, one for the training input

$$v(t) = \begin{cases} -1 & \text{if } 0 \le t < 0.5 \\ 1 & t \ge 0.5 \\ 0 & \text{otherwise} \end{cases}$$

and one for the test input $v(t) = \cos(\frac{\pi}{2}t)$, and with two different weight functions: the state dependent weight function (4) and the time dependent weight function (5). In the figures and tables below we refer to each case as *Rewienski* and *splines*, respectively. The simulations run from 0 to 10 time units, the initial condition is zero, $N = 100$, and the output $y(t)$ is the voltage at node 1. The linear models of the TPWL model are extracted using the algorithm described in [4], the number of extracted linear models is 10 and the matrix V_q is build using the algorithm of [3] and has rank 10. The simulation was done using MATLAB R2007b' implicit ODE solver

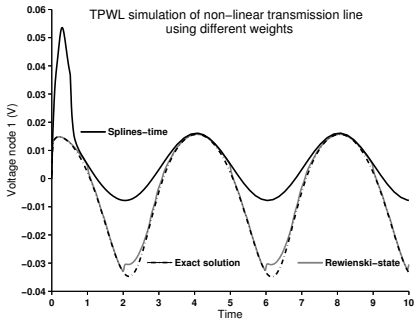

Fig. 3 Simulation of a non-linear transmission line using TPWL.

`ode15s` on a Intel Core 2 DUO L7700 @ 1.80Ghz with 2.0Gb of RAM memory running a Linux kernel 2.6.26-2-686.
Figure 3 shows the output of the simulation, where the exact solution is obtained by a simulation of the full system. This is a *worse case scenario* where the shape of the training input is different from the test input. The bad performance of the time-dependent weights can be understood by looking at the weight functions in Figure 1. The test input requires the TPWL to strongly switch between the linear models, something model (6) does not do. One might be tempted to reverse the order of the inputs so that the TPWL is set up with a more oscillating input. However our experiments show that model (6) will still perform poorly, but more interestingly model (1) using weights (4) initially follows the exact solution, but eventually diverges.

Acknowledgements This work, part of the SyreNe network, is supported by the German Federal Ministry of Education and Research (BMBF) grant no. 03FAPAE2.

References

1. Bechtold, T., Striebel, M., Mohaghegh, K., ter Maten, E.J.W.: Nonlinear model order reduction in nanoelectronics: Combination of POD and TPWL. In: PAMM, vol. 8, pp. 10,057–10,060. WILEY-VCH Verlag GmbH & Co. (2008). 79th Annual Meeting of the International Association of Applied Mathematics and Mechanics (GAMM)
2. Dong, N., Roychowdhury, J.: Automated nonlinear macromodeling of output buffers for high-speed digital applications. In: ACM/IEEE Design Automation Conference, pp. 51–56 (2005).
3. Odabasioglu, A., Celik, M., Pileggi, L.T.: PRIMA: passive reduced-order interconnect macromodeling algorithm. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **17**(8), 645–654 (1998). DOI 10.1109/43.712097
4. Rewiensky, M.: A trajectory piecewise-linear approach to model order reduction of nonlinear dynamical systems. Ph.D. thesis, Massachusetts Institute of Technology (2003)
5. Striebel, M., Rommes, J.: Model order reduction of non-linear systems: status, open issues, and applications. Tech. Rep. CSC/08-07, Technische Universität Chemnitz (2008)
6. Tiwary, S., Rutenbar, R.A.: Scalable trajectory methods for on-demand analog macromodel extraction. In: ACM/IEEE Design Automation Conference, pp. 403–408 (2005).
7. Vasilev, D., Rewiensky, M., White, J.: A TBR-based trajectory piecewise-linear algorithm for generating accurate low-order models for analog circuits and mems. DAC pp. 490–495 (2003)