SANDIA REPORT

SAND2011-6223
Unlimited Release
Printed Printed October 2011

Accelerating Transient Simulation of Linear Reduced Order Models

Brad Bond, Eric Keiter, Ting Mei, Heidi Thornquist

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multiprogram laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Sandia National Laboratories

SAND2011-6223
Unlimited Release
Printed Printed October 2011

Accelerating Transient Simulation of Linear Reduced Order Models

Brad Bond, Eric Keiter, Ting Mei, Heidi Thornquist
Electrical and Microsystems Modeling

Sandia National Laboratories
P.O. Box 5800
Mail Stop 0316
Albuquerque, NM 87185-0316

Abstract

Model order reduction (MOR) techniques have been used to facilitate the analysis of dynamical systems for many years. Although existing model reduction techniques are capable of providing huge speedups in the frequency domain analysis (i.e. AC response) of linear systems, such speedups are often not obtained when performing transient analysis on the systems, particularly when coupled with other circuit components. Reduced system size, which is the ostensible goal of MOR methods, is often insufficient to improve transient simulation speed on realistic circuit problems. It can be shown that making the correct reduced order model (ROM) implementation choices is crucial to the practical application of MOR methods. In this report we investigate methods for accelerating the simulation of circuits containing ROM blocks using the circuit simulator Xyce.

Contents

List of Figures

Chapter 1

Introduction

Model order reduction (MOR) techniques have been used to facilitate the analysis of dynamical systems for many years. Although existing model reduction techniques are capable of providing huge speedups in the frequency domain analysis (i.e. AC response) of linear systems, such speedups are often not obtained when performing transient response on the systems, particularly when coupled with other circuit components. That is, computing $\hat{H}(s) = \hat{L}^T(s\hat{C} + \hat{G})\hat{B}$ is extremely fast relative to $H(s) = L^T(sC + G)B$, while solving $\hat{C}\dot{\hat{x}} + \hat{G}\hat{x} = \hat{B}u$ for $x(t)$ is slow compared to solving the original system $C\dot{x}+Gx = Bu$. In this report we investigate methods for accelerating the simulation of circuits containing ROM blocks using the circuit simulator Xyce [1, 2]

1.1 Model Reduction in Xyce

A linear circuit (e.g. RLC network) can be modeled using Modified Nodal Analysis (MNA) [3], resulting in the following state-space model

$$C\dot{x} + Gx = Bu \qquad\qquad y = L^T x. \tag{1.1}$$

Here the state variables x correspond to node voltages and inductor currents. Standard projection methods for model reduction approximate the state x using a linear combination of basis vectors, $x = V\hat{x}$, where $\hat{x}$ are the state variables of the reduced model. Applying the projection approximation to system (1.1) and left-multiplying by the left-projection matrix W results in the reduced model

$$\hat{C}\dot{\hat{x}} + \hat{G}\hat{x} = \hat{B}u \qquad\qquad y = \hat{L}^T\hat{x} \tag{1.2}$$

The projection matrices V, W are chosen such that the input-output behavior of the reduced model matches the input-output behavior of the original system.

The reduced model can be thought of as an N-port circuit element that can be simulated by itself, or connected to other, possibly nonlinear, circuit elements or blocks. Transient simulation of the resulting system in Xyce uses a dynamic timestep scheme to integrate forwards through time. The step size is chosen based on local truncation error approximations, and is roughly taken as

$$\Delta t = \min_i \left| \frac{\alpha}{\frac{d^k x_i}{dt^k}} \right|^{\frac{1}{k}} \tag{1.3}$$

where α is some constant and k is related to the order of the differencing scheme. For example, when using trapezoidal rule, $k = 3$, so

$$\Delta t = \min_i \left| \frac{\alpha}{\dddot{x}_i} \right|^{\frac{1}{3}}. \tag{1.4}$$

1.2 Simulation Difficulties Arising from Reduced Model Blocks

Based on our experiments, there are two main reasons that transient simulation of the reduced model is slow. First, the reduced model typically requires many more time steps. The standard time integration routine in Xyce automatically determines the time step size based on estimations of the local truncation error at each step. The size of the step is precisely tuned for circuit examples, where the state values correspond to node voltages and branch currents that arise is common circuits. Since the states in the reduced model correspond to linear combinations of states in the large system, each reduced state by itself is physically meaningless, and therefore generally does not take values in the range corresponding to standard voltages and currents. Second, although the size of the reduced model matrices are smaller, the number of nonzeros is not as significantly reduced. This is because C, G, B, L are typically extremely sparse, whereas $\hat{C}, \hat{G}, \hat{B}, \hat{L}$ are all extremely dense. This makes the linear system solves at every time step (or at every Newton iteration, if the system is nonlinear) more expensive.

In general, dense matrices, and state variables taking a wide range of values, should not cause difficulties for a transient simulator. However, Xyce has been finely tuned to handle sparse circuit problems, and therefore performance is sacrificed when considering other types of problems.

Chapter 2

The Problems

First, we consider the problem of the simulator taking too many time steps during transient simulation of a reduced model.

2.1 Dynamic Step Size Selection – Too many timesteps

As briefly explained in Section 1.1, the step sizes taken by the transient simulator are chosen based on approximations of the local truncation error at each step. To better illustrate why the reduced models cause the simulator to take too many time steps, we first look at two simple examples.

2.1.1 Case #1: Poorly scaled states

We begin with the original "large" system

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad y = \begin{bmatrix} 1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{2.1}$$

Using the following projection relation, this can reduced to a single state system as follows

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}\hat{x}, \qquad \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 10^{-9} \\ 10^{-9} \end{bmatrix}, \qquad \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} 10^{9} \\ 10^{9} \end{bmatrix} \tag{2.2}$$

Selecting $x_0 = 10^{-3}$ and simulating both the original large model and the reduced model for one second yields the results shown in Figure 2.1. Note that although the models produce the same solution, the reduced model reuiqres 10× more timesteps over the same time interval.

2.1.2 Case #2: High Frequency Junk

To see another, more subtle, case that can cause problems, consider the four-state "reduced" system

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -\lambda & \phi & 0 & 0 \\ -\phi & -\lambda & 0 & 0 \\ 0 & 0 & -\lambda & \omega \\ 0 & 0 & -\omega & -\lambda \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \qquad y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}. \tag{2.3}$$

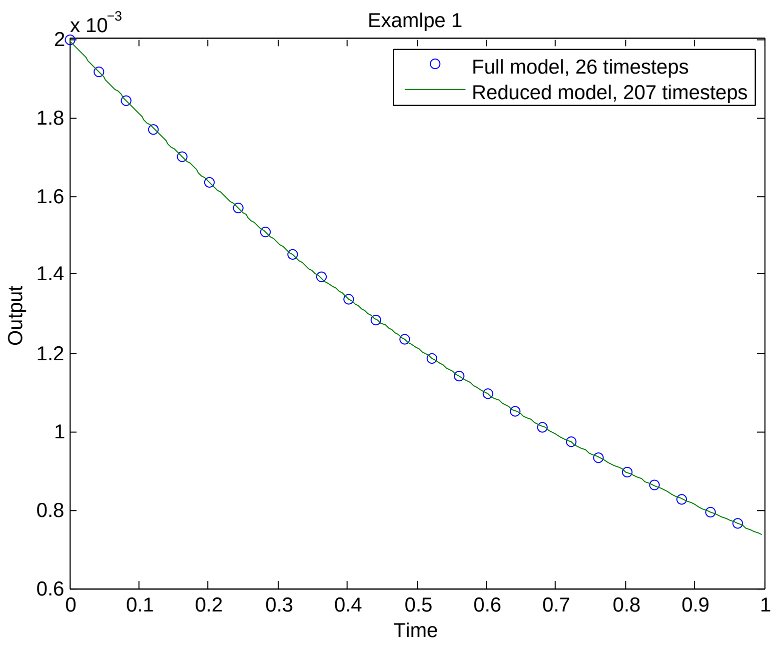

Figure 2.1. Transient response of large system (2.1) and the corresponding reduced model. Due to poorly scaled variables, the reduced system required 10× more time steps than the large system.

Here we are assuming this system is the result of a model reduction procedure, and we do not care about the original large system from which it was created. From (2.3) we can see that the solution will be oscillating decaying exponentials, and that x_1, x_2 are decoupled from x_3, x_4. In fact, if we define the initial conditions as along with the initial condition

$$\begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \\ x_4(0) \end{bmatrix} = \begin{bmatrix} \eta \\ \eta \\ \gamma \\ \gamma \end{bmatrix}, \tag{2.4}$$

then we can write the analytical solution as

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} = \begin{bmatrix} \eta\ e^{-\lambda t}\left[\cos(\phi t) + \sin(\phi t)\right] \\ \eta\ e^{-\lambda t}\left[\cos(\phi t) - \sin(\phi t)\right] \\ \gamma\ e^{-\lambda t}\left[\cos(\omega t) + \sin(\omega t)\right] \\ \gamma\ e^{-\lambda t}\left[\cos(\omega t) - \sin(\omega t)\right] \end{bmatrix}. \tag{2.5}$$

So the solution is two pairs of complex exponentials that oscillate at different frequencies but decay at the same rate.

When simulating this system in Xyce, the timesteps will depend on the third derivatives with respect to time of the analytic solution (2.5), which is

$$\frac{d^3}{dt^3}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} = \begin{bmatrix} -\eta\lambda^3\phi^3 e^{-\lambda t}\left[-\cos(\phi t) - \sin(\phi t)\right] \\ -\eta\lambda^3\phi^3 e^{-\lambda t}\left[-\cos(\phi t) + \sin(\phi t)\right] \\ -\gamma\lambda^3\omega^3 e^{-\lambda t}\left[-\cos(\omega t) - \sin(\omega t)\right] \\ -\gamma\lambda^3\omega^3 e^{-\lambda t}\left[-\cos(\omega t) + \sin(\omega t)\right] \end{bmatrix}. \tag{2.6}$$

To see what all of this means, plug in the following parameters

$$\phi = 1 \qquad \omega = 10^5 \qquad \lambda = 0.1 \qquad \eta = 10 \qquad \gamma = 10^{-2} \tag{2.7}$$

The solution (2.5) becomes

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} = \begin{bmatrix} 10\ e^{-0.1t}\left[\cos(t) + \sin(t)\right] \\ 10\ e^{-0.1t}\left[\cos(t) - \sin(t)\right] \\ 10^{-2}\ e^{-0.1t}\left[\cos(10^5 t) + \sin(10^5 t)\right] \\ 10^{-2}\ e^{-0.1t}\left[\cos(10^5 t) - \sin(10^5 t)\right] \end{bmatrix}, \tag{2.8}$$

so that the output $y(t)$ is a decaying oscillation at 1Hz, with a *very* small component on top oscillating at 10^5Hz

$$y(t) = e^{-0.1t}\left[20\cos(t) + 0.02\cos(10^5 t)\right] \tag{2.9}$$

The high frequency oscillation is 0.1% of the low frequency signal, and therefore adds a negligible effect.

However, let's look at the third derivatives of the solution and see how the simulator will select the step sizes

$$\frac{d^3}{dt^3}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} = \begin{bmatrix} -10^{-2}e^{-0.1t}\left[-\cos(t) - \sin(t)\right] \\ -10^{-2}e^{-0.1t}\left[-\cos(t) + \sin(t)\right] \\ -10^{10}e^{-0.1t}\left[-\cos(10^5 t) - \sin(10^5 t)\right] \\ -10^{10}e^{-0.1t}\left[-\cos(10^5 t) + \sin(10^5 t)\right] \end{bmatrix}. \tag{2.10}$$

It's clear from (2.10) that the step size is going to be determined by the high frequency components, because they are 12 orders of magnitude larger than the low frequency components. That is, even though the high frequency signal is negligible, the simulator will try to take small timesteps to capture the microsecond dynamics of its oscillations.

To really see how much the high frequency components affect step size, we will run a regular transient simulation using dynamic timestepping, and then modify the timestepping routine slightly so it ignores the high frequency components in the step size computation and run the transient again. The results, shown in Figure 2.2, show that ignoring the high frequency components causes the simulator to take just 51 steps, and captures the output accurately. However, including the high frequency components causes the simulator to take almost 10,000 steps – an increase by a factor of 200!

In general we won't have a problem where the states are nicely partitioned between the important components and unimportant components, so it won't be so simple to determine what to ignore. Also, we would prefer not to modify the simulator for each system we need to simulate, and instead only modify the model we put into it.

2.2 Dense Matrix Blocks

Next, we consider the problem resulting from dense ROM blocks being inserted into the larger system. The original state space model (1.1) is sparse because the state variables x are the node

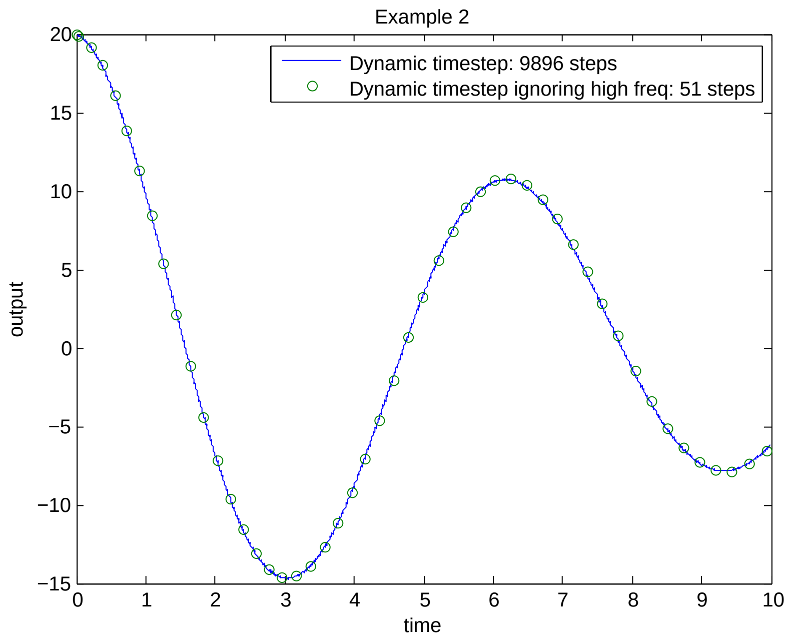

Figure 2.2. Transient simulation of system (2.3) using parameters defined in (2.7).

voltages, and circuit elements only connect neighboring nodes. The reduced model results from projection, which can be interpreted as a rotation of the coordinate system followed by truncation of state variables. After rotation, the state variables no longer correspond to node voltages, so there is no reason to expect the conservation equations to depend on a small number of the state variables, which means the system matrices will be dense.

Chapter 3

Solution # 1: Linear Transformation

One solution to the problems of poorly scaled variables and dense matrices is to transform the linear ROM model to one that has properly scaled variables and sparser (or nicely structured) matrices.

3.1 Scaling the Variables

First, we consider the issue of state variables taking non-physical (or rather, non-circuit) values, which causes the time-integrator to take too many steps. This time-stepping problem can be solved by "simply" scaling the variables in the reduced model (the idea of scaling is simple, but determining the scaling factors (efficiently) is non-trivial). Recall that the state variables in the reduced model $\hat{x}$ are related to the state variables of the original system x via $x = V\hat{x}$. If x are the circuit state variables (current, voltage), then the reduced variables $\hat{x}$ should be the same order of magnitude as x, i.e. $|x_i| \approx |\hat{x}_i|$. Suppose $|x_i| \approx \alpha$ for each element of x. We want to construct V such that $|\hat{x}_i| \approx \alpha$ as well. Given a state x, its image in the reduced space is $\hat{x} = V^T x$, and therefore each individual reduced state is the image of the large state x on each column of V, i.e. $\hat{x}_i = V_i^T x$, where V_i is the i^{th} column of V. Thus, given a set of solutions $x(t)$, the projection matrix V should be chosen such that the state images in the reduced space are order of magnitude α, i.e. $|\hat{x}_i| =\approx \alpha$.

Let β be a diagonal matrix such that $V = \tilde{V}\beta$ scales each column V_i by β_{ii}. If $\tilde{V}$ is the original projection matrix (constructed, for instance, using moment matching), then we want to select the β_{ii} such that

$$|\hat{x}_i| = |\beta_{ii} V_i^T x| \approx \alpha \tag{3.1}$$

From (3.1) we find that

$$\beta \approx \frac{\alpha}{V_i^T x} \tag{3.2}$$

We assume that α is given from the start, and V is computed using standard model reduction procedures, so the lone question is: what do we use for the state x. We are trying to find β such that $|\hat{x}_i| \approx \alpha$ for all times during transient simulation. If we know a transient solution trajectory $x(t)$ ahead of time, then those states would be a good choice from computing β as in (3.2). However, we would like to avoid solving the original full system. But we can easily compute $X(j\omega)$, the frequency response of the system at a particular frequency. Given $X(j\omega)$ in response to a particular input $u(t) = sin(\omega t)$, we can compute the steady-state response in time-domain as $x(t) = |X(j\omega)| \exp(j\angle X(j\omega)\omega t)$. Thus, $|X(j\omega)|$ tells us the magnitude of the state x at steady-state, and provides a good sample of the trajectory during transient simulation.

It is important to note that scaling the columns of V does not change the span of the projection matrix, nor does it alter the frequency response of the resulting reduced model. See Section A for more details on this subject.

To summarize, the algorithm for determining the projection matrix utilizing variable scaling is as follows.

1. Compute nominal projection matrix $\tilde{V}$ using any standard technique
2. Compute snapshots x_j of the steady-state at several frequencies of interest ω_j
3. Compute $\beta_{ii}^{j} = \frac{\alpha}{V_i^T x_j}$
4. Define β_{ii} as the average of β_{ii}^{j} over j
5. Define $V = \tilde{V}\beta$

Based on example 2, an alternative approach would be to eliminate the negligible high-frequency components from the system. However, these frequencies are determined by the eigenvalues/poles of the system, and therefore modifying them would constitute modifying the transfer function / frequency response of the system – which is what we are trying to avoid. Therefore, our only option is a coordinate transformation.

3.2 Fixing Case #1 and Case #2

In order to fix the two previously presented examples, we apply the scaling procedure described above. The first example can be fixed by simply scaling the reduced model variables, which is equivalent to scaling the columns of the projection matrix. After scaling the projection matrices to have unity norm, the resulting model now simulates in the same number of timesteps as the original model, shown in Figure 3.1.

The second example can be fixed by scaling the variables using the procedure described above. The resulting model now simulates in just 51 steps, which is the same number of steps taken when explicitly ignoring the high frequency components, as shown in Figure 3.2.

3.3 Rotating the Coordinate System

The problem resulting from dense ROM blocks is a bit more complicated than the time-stepping issue, but can be partially addressed using another change of variable. However, instead of simply scaling the projection matrix columns, we will now apply a linear transformation to them. Specifically, we want to transform W, V such that the reduced system matrices $\hat{C}$, $\hat{G}$, $\hat{B}$, $\hat{L}$ are sparse, while ensuring we do not affect the accuracy of the ROM.

Consider the reduced order linear system

$$\hat{C}\dot{\hat{x}} = \hat{G}\hat{x} + \hat{B}u, \tag{3.3}$$

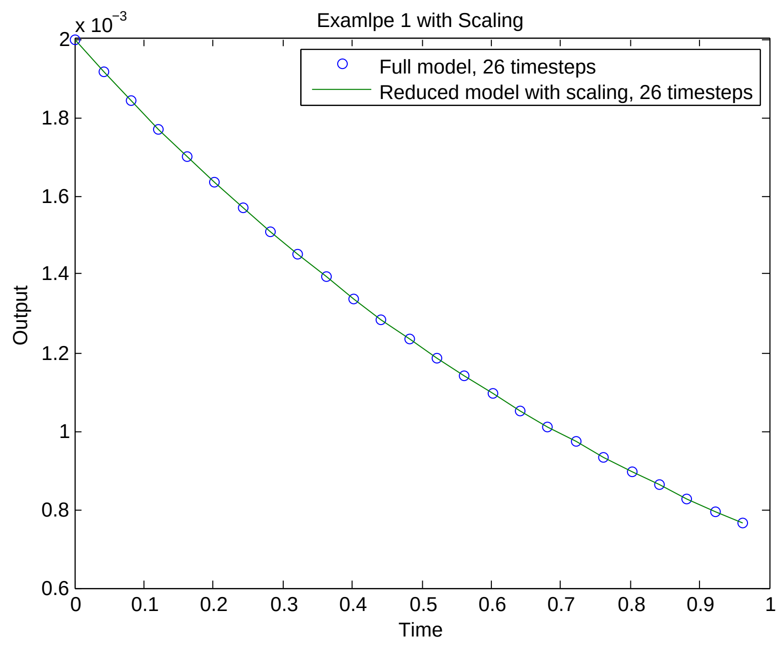

Figure 3.1. Transient simulation of full model (blue circles) and scaled reduced model (solid green line). Scaling the variables has reduced the number of steps required for the reduced model by a factor of 8.

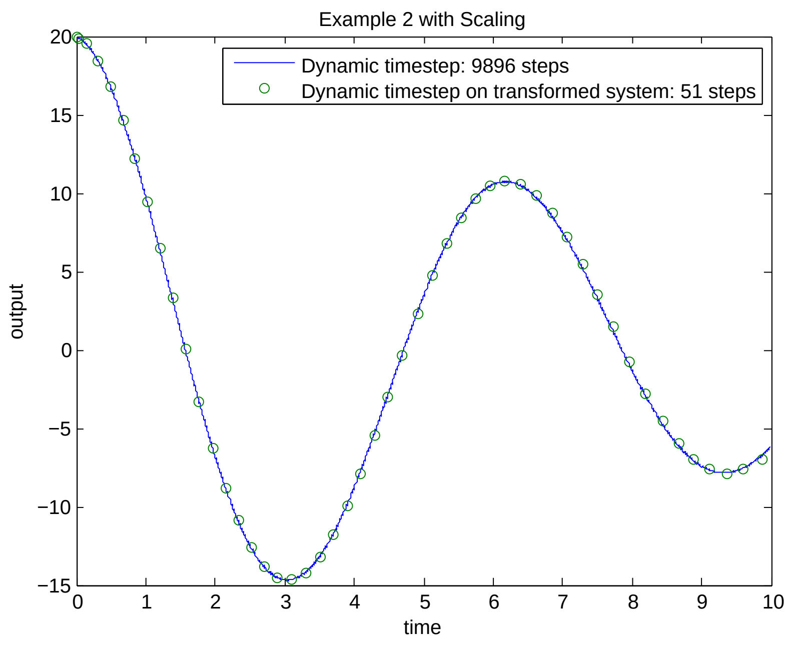

Figure 3.2. Transient simulation of unscaled reduced model (solid blue line) and scaled reduced model (green circles). Scaling the variables has decreased the number of required timesteps by a factor of almost 200.

and suppose it has purely real eigenvalues and permits the generalized eigenvalue decomposition $\hat{C}T = \hat{G}TD$, where T are the eigenvectors and D contains the eigenvalues. Using the change of coordinates $\hat{x} = T\tilde{x}$, we can rewrite (3.3) as

$$\hat{C}T\dot{\tilde{x}} = \hat{G}T\tilde{x} + \hat{B}u$$

Applying the generalized eigenvalue relation yields

$$\hat{G}TD\dot{\tilde{x}} = \hat{G}T\tilde{x} + \hat{B}u \tag{3.4}$$

which we can premultiply by $T^{-1}\hat{G}^{-1}$ to obtain

$$D\dot{\tilde{x}} = \tilde{x} + (T^{-1}\hat{G}^{-1}\hat{B})u \tag{3.5}$$

What once was $\hat{C}$ is now the diagonal matrix D, and what once was $\hat{G}$ is now an identity matrix. Thus, we have transformed the dense system (3.3) into a sparse (diagonal) system. After solving (3.5) for $\tilde{x}$, we can obtain the solution to the original system via the transform $\hat{x} = T\tilde{x}$.

To see how this transformation can be absorbed into the projection matrix, we will rewrite (3.4) after un-collapsing the projection

$$W^TGVTD\dot{\tilde{x}} = W^TGVT\tilde{x} + W^TBu$$

$$(W^TGVT)^{-1}W^TGVTD\dot{\tilde{x}} = (W^TGVT)^{-1}W^TGVT\tilde{x} + (W^TGVT)^{-1}W^TBu$$

Therefore, if V_0 and W_0 are the nominal projection matrices (that produce a dense reduced model), we define $V = V_0T$ and $W = W_0(W_0^TGV_0T)^{-T}$.

Although the previously described transformation sparsified the given example, there are two important details that need to be addressed. First, it was previously assumed the system has purely real eigenvalues. If that is not the case, then we cannot diagonalize the system while keeping the matrices real. However, we can perform a generalized Schur decomposition (i.e. QZ decomposition) to obtain quasi-upper-triangular matrices or quasi-diagonal matrices.

A larger problem, however, is that we are assuming the number of ports (i.e. inputs and outputs) is small relative to the number of states in the original system. This means that B is tall and skinny. If the number of ports is comparable to the number of states in the large system, then $\hat{B}$ will be dense and contain $O(Nq)$ nonzeros. Unfortunately, it is impossible to simultaneously sparsify $\hat{C}, \hat{G}, \hat{B}$. This is sort of the whole point of model reduction – we are decreasing the order of the system, but as a result, the ROM will be highly coupled (i.e. dense matrices). If we want the states to be decoupled (sparse matrices), then the inputs must touch every state (dense $\hat{B}$ matrix).

Chapter 4

Solution #2: Two-level Solve

An alternative solution, instead of manipulating the reduced model system, is to solve the equations corresponding to the reduced model separately. In this case we can use a different solver that can *efficiently* handle the dense ROM blocks that are coupled in the larger sparse matrices. Additionally, this avoids the problems caused by poorly scaled variables because the ROM states are now never seen by the outer level solver. This is a linear version of the two-level Newton solve presented in [4].

4.1 Simple Illustrative Example

Consider two non-dynamical systems (e.g. resistor networks) connected by a single port, as shown in Figure 4.1. This interconnected system can be modeled using KCL, resulting in the following

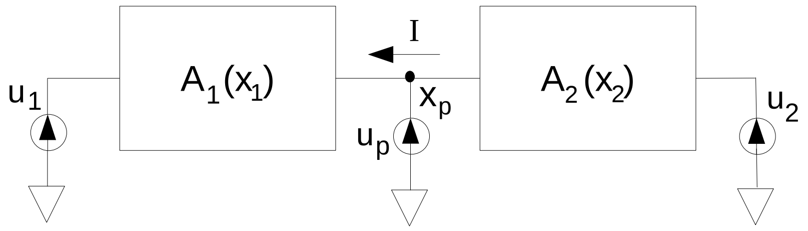

Figure 4.1. Example showing two loosely coupled (i.e. connected by one port) system blocks, each with input sources.

system of state space equations

$$\begin{bmatrix} F_1 \\ F_p \\ F_2 \end{bmatrix} = \begin{bmatrix} A_1 & A_{1p} & 0 \\ A_{p1} & A_p & A_{p2} \\ 0 & A_{2p} & A_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_p \\ x_2 \end{bmatrix} - \begin{bmatrix} B_1 u_1 \\ B_p u_p \\ B_2 u_2 \end{bmatrix} = 0. \tag{4.1}$$

Here $x_1 \in \mathbb{R}^N$ are the internal variables of the first block, $x_p \in \mathbb{R}^p$ are the port variables (in this case $p = 1$), $x_2 \in \mathbb{R}^q$ are the internal variable of the second block, $A_1 \in \mathbb{R}^{N\times N}$, $A_{1p} \in \mathbb{R}^{N\times p}$, $A_{p1} \in \mathbb{R}^{p\times N}$, $A_p \in \mathbb{R}^{p\times p}$, $A_{p2} \in \mathbb{R}^{p\times q}$, $A_{2p} \in \mathbb{R}^{q\times p}$, and $A_2 \in \mathbb{R}^{q\times q}$. Let us further assume that A_1, A_{1p}, and A_{p1} are sparse, while A_2 and A_{2p} are dense, which would arise when the second block is a reduced order model.

Because the system contains a large sparse block and a large dense block, it would be inefficient to use either a dense solver or a sparse solver. However, suppose we are only interested in solving for the voltages in the first block (i.e. x), which is realistic if the second block is a reduced model. Since the problem is weakly coupled (i.e. there are a small number of ports connecting the two blocks), it is possible to eliminate the large dense second block from the system ahead of time, resulting in a purely sparse system.

4.1.1 Intuitive Explanation

As shown in Figure 4.1, the two blocks are coupled through the current I. As it is written, this current I depends on voltages in the first block, x_1, and voltages in the second block, x_2. However, it is possible to develop an expression for I that depends *only* on the port voltage x_p, and none of the internal variables x_2. Essentially we are going to replace the second block with equivalent resistors between x_p and ground.

The current I can be expressed as

$$I = B_p u_p + \tilde{A}_p x_p + \tilde{B} u_2, \tag{4.2}$$

where $\tilde{A}_p$ is the equivalent conductance of the second block, and $\tilde{B}$ are equivalent sources to account for the sources that were connected to the second block. Since conductance relates changes in voltage to changes in current, we can compute this expression as follows.

$$\begin{aligned}\tilde{A}_p &= \frac{dF_p}{dx_p} = \frac{\partial F_p}{\partial x_2}\frac{\partial x_2}{\partial x_p} + \frac{\partial F_p}{\partial x_p} \\ \tilde{B} &= \frac{dF_p}{du} = \frac{\partial F_p}{\partial x_2}\frac{\partial x_2}{\partial u}\end{aligned}$$

The equations F_p are just the conservation equations in the previous system and are known explicitly. Note that the term $\partial F_p/\partial u_p$ is B_p, which was separated out in (4.2) because it is already available explicitly, so it is not present in $\tilde{B}$. The unknown terms here are $\partial x_2/\partial x_p$ and $\partial x_2/\partial u$, which can be obtained by differentiating F_2 with respect to x_p and u

$$\begin{aligned}\frac{\partial F_2}{\partial x_p} &= \frac{\partial F_2}{\partial x_2}\frac{\partial x_2}{\partial x_p} \\ \frac{\partial F_2}{\partial u} &= \frac{\partial F_2}{\partial x_2}\frac{\partial x_2}{\partial u}\end{aligned}$$

Solving for $\partial w/\partial v$ and $\partial w/\partial u$ and plugging into the previous expression, we find the equivalent conductance and inputs as

$$\begin{aligned}\tilde{A}_p &= -\frac{\partial F_p}{\partial x_2}\frac{\partial F_2}{\partial x_2}^{-1}\frac{\partial F_2}{\partial x_p} + \frac{\partial F_p}{\partial x_p} \\ \tilde{B} &= -\frac{\partial F_p}{\partial x_2}\frac{\partial F_2}{\partial x_2}^{-1}\frac{\partial F_2}{\partial u}\end{aligned}$$

These partial derivative terms can be extracted directly from system (4.1), resulting in

$$\begin{aligned}\tilde{A}_p &= -A_{p2}A_2^{-1}A_{2p} + A_p \\ \tilde{B} &= -A_{p2}A_2^{-1}B_2,\end{aligned}$$

which can each be solved using a dense solver.

Now that we have an expression for the current I that depends only on the port voltages x_p and inputs u, we can rewrite the original system of equations independent of x_2 as

$$\begin{bmatrix} A_1 & A_{1p} \\ A_{p1} & \tilde{A}_p \end{bmatrix} \begin{bmatrix} x_1 \\ x_p \end{bmatrix} = \begin{bmatrix} B_1 u_1 \\ B_p u_p + \tilde{B} u_2 \end{bmatrix}. \tag{4.3}$$

Here $\tilde{A}_p$ is a dense block, but it is size $p \times p$, and $p << q$, so the system can be efficiently solved using a sparse solver.

4.1.2 Mathematical Explanation

The basic idea behind the two-level solve is to use a Schur complement to eliminate the w variables corresponding to the dense block by first symbolically solving for w in terms of v. For example, consider the last set of equations

$$A_{2p} x_p + A_2 x_2 - B_2 u_2 = 0.$$

This can be solved for x_2 in terms of x_p as

$$x_2 = A_2^{-1} \left(B_2 u_2 - A_{2p} x_p \right).$$

Now, the second block of equations, which are

$$A_{p1} x_1 + A_p x_p + A_{p2} x_2 - B_p u_p = 0,$$

can be rewritten using the new expression for w, resulting in

$$A_{p1} x_1 + A_p x_p + A_{p2} \left(A_2^{-1} (B_2 u_2 - A_{2p} x_p) \right) - B_p u_p = 0.$$

Finally, we can rewrite the full linear system now using only the variables x_1 and x_p

$$\begin{bmatrix} A_1 & A_{1p} \\ A_{p1} & A_p - A_{p2} A_2^{-1} A_{2p} \end{bmatrix} \begin{bmatrix} x_1 \\ x_p \end{bmatrix} - \begin{bmatrix} B_1 u_1 \\ B_p u_p - A_{p2} A_2^{-1} B_2 u_2 \end{bmatrix} = 0. \tag{4.4}$$

4.1.3 Benefit

In both approaches, the end result is that we have eliminated the $N \times N$ dense block A_2, and replaced it with the $p \times p$ dense block $A_{p2} A_2^{-1} A_{2p}$, where $p << q$. Furthermore, we can use efficient dense solvers to compute $A_2^{-1} A_{2p}$ since this step is completely separate from solving the rest of the sparse system.

4.2 Discretized Circuit Equations

Now suppose each of the blocks in Figure 4.1 is a dynamical circuit modeled using MNA, so that the full system has the form

$$\begin{bmatrix} F_1 \\ F_p \\ F_2 \end{bmatrix} = \begin{bmatrix} \dot{q}_1(x_1, x_p) \\ \dot{q}_p(x_1, x_p, x_2) \\ \dot{q}_2(x_p, x_2) \end{bmatrix} + \begin{bmatrix} f_1(x_1, x_p) \\ f_p(x_1, x_p, x_2) \\ f_2(x_p, x_2) \end{bmatrix} = 0 \tag{4.5}$$

Again, we denote the internal ROM states as x_2, the port nodes of the ROM as x_p, and everything external to the ROM as x_1.

As before, we want to eliminate the variables x_2 from the system. The second block of equations can be expanded as

$$F_p = \dot{q}_p(x_1, x_p) + f_{p1}(x_1, x_p) + f_{p2}(x_2)$$

Therefore, our goal is to replace the term $f_{p2}(x_2)$ with a new term $\Delta f_p(x_p)$ that represents the total current flowing out of the ROM into the port nodes x_p. To do this, we simply solve the third block of equations for x_2 in terms of x_p. However, first we must discretize the ODEs to obtain an algebraic set of equations that can be solved. The standard discretization yields a system of the form

$$F(x^t, x^{t-1}) = \alpha(q - q^{t-1}) + \beta f + (1-\beta) f^{t-1} = 0$$

Here α and β are determined by the time discretization method chosen. In general, we assume $\alpha = 1/\Delta t$, and $\beta = 1$ for Backward Euler, or $\beta = 1/2$ for Trapezoidal Rule. Since the ROM is linear, we can write $q = Cx$ and $f = Gx$, and the last block of equations becomes

$$F_2 = \beta G_{2p} x_p + (\alpha C_2 + \beta G_2) x_2 + (1-\beta) G_{2p} x_p^{t-1} + ((1-\beta) G_2 - \alpha C_2) x_2^{t-1}. \tag{4.6}$$

Solving this last block for x_2 yields

$$x_2 = -(\alpha C_2 + \beta G_2)^{-1} \left(\beta G_{2p} x_p + (1-\beta) G_{2p} x_p^{t-1} + ((1-\beta) G_2 - \alpha C_2) x_2^{t-1} \right). \tag{4.7}$$

Since the ROM is linear, we have

$$f_{p2}(x_2) = G_{p2} x_2,$$

and substituting in the expression for x_2 yields

$$\Delta f_p(x_p) = -G_{p2}(\alpha C_2 + \beta G_2)^{-1} \left(\beta G_{2p} x_p + (1-\beta) G_{2p} x_p^{t-1} + ((1-\beta) G_2 - \alpha C_2) x_2^{t-1} \right)$$

Thus, the new system of equations to be solved by the simulator is

$$\begin{bmatrix} F_1 \\ F_p \end{bmatrix} = \begin{bmatrix} \dot{q}_1(x_1, x_p) \\ \dot{q}_p(x_1, x_p, x_2) \end{bmatrix} + \begin{bmatrix} f_1(x_1, x_p) \\ f_{p1}(x_1, x_p) + \Delta f_p(x_p) \end{bmatrix} = 0 \tag{4.8}$$

In the terminology of a two-level solver, Solving system (4.8) is the “outer” solve, and the “inner” solve occurs when we solve (4.6) for x_2, as in (4.7)..

4.3 Two-level Device Stamps

Now that we have an explicit expression for simplified system we want to solve, we simply need to determine how to obtain the new system (4.8) using ROM device stamps. That is, we cannot modify directly the MNA equation for F_p to add the Δf_p term in (4.8). Instead, the ROM model will have a stamp for $\bar{q}$ and $\bar{f}$ that affect only $F_p(x)$, and we want to choose these quantities to ensure the final system has the form of (4.8)

The “inner” solve takes place inside of the device function, while the “outer” solve is taken care of at a higher level by the time integration routine.

4.3.1 Residual Stamp

The residual stamp from the ROM affects only the port nodes x_p, and is the previously derived expression for $G_{p2}x_2$

$$\bar{f} = \Delta f_p(x_2) = -G_{p2}(\alpha C_2 + \beta G_2)^{-1}\left(\beta G_{2p}x_p + (1-\beta)G_{2p}x_p^{t-1} + ((1-\beta)G_2 - \alpha C_2)x_2^{t-1}\right) \quad (4.9)$$

Since the charge terms in the ROM are accounted for in $\bar{f}$ already, we can leave the charge stamp empty

$$\bar{q} = 0$$

4.3.2 Jacobian Stamp

The stamp into the Jacboain is

$$\frac{\partial \bar{F}_p}{\partial x_p} = \frac{\partial \bar{F}_p}{\partial x_2}\frac{\partial x_2}{\partial x_p} = G_{p2}A_2^{-1}\beta G_{2p} \quad (4.10)$$

This is the stamp to the full Jacobian $\alpha C + \beta G$. However, the device stamps individually into C and G, so we will have to define

$$\bar{C} = 0 \quad (4.11)$$

$$\bar{G} = G_{p2}A_2^{-1}\beta G_{2p} \quad (4.12)$$

Note, there are many possible ways to distribute this stamp between $\bar{C}$ and $\bar{G}$. One reason to set $\bar{C} = 0$ and not $\bar{G}$ is that $\bar{C}$ will not be called when solving for the DC operating point.

4.4 Efficient Computation of the Device Stamp

Each call to the ROM device will produce the four stamps $\bar{q}, \bar{f}, \bar{C}, \bar{G}$. The charge and capacitance stamps $\bar{q}$ and $\bar{C}$ are zero, so we can ignore those.

The conductance stamp

$$\bar{G} = \bar{G}(\alpha, \beta) = G_{p2}\left(\alpha C_2 + \beta G_2\right)^{-1}\beta G_{2p} \quad (4.13)$$

depends only on α and β, which depend on the timestep and integration order, respectively.

The residual stamp

$$\bar{f} = \bar{f}(\alpha, \beta, x_2, x_2^{t-1}) = G_{p2}x_2 \quad (4.14)$$

additionally depends on the internal state x_2 and the port voltages x_p.

The internal state also depends on port voltages and previous internal states.

$$x_2 = (\alpha C_2 + \beta G_2)^{-1}\left(\beta G_{2p}x_p + (1-\beta)G_{2p}x_p^{t-1} + ((1-\beta)G_2 - \alpha C_2)x_2^{t-1}\right) \quad (4.15)$$

It turns out that many of these terms, which are required at every call to the device function, can be reused. This is true both within a newton loop and from one timestep to the next.

At a particular time step, the first thing to be computed is the internal state at the previous timestep. However, unless we are at the very first timestep, this quantity should already be available from the previous timestep. This procedure is explained in Algorithm 1.

Algorithm 1 Compute x_2^{t-1}

```
**INPUTS:** $x_2^{t-1}$, $x_2^{t-2}$, $x_p^{t-2}$, $x_p^{t-1}$, $\alpha^{t-1}$, $\beta^{t-1}$, $A_2^t$, $A_2^{t-1}$, $C_2$, $G_2$, $G_{2p}$, $G_{p2}$
if $k = 0$ then
    if $(t = 0) \parallel$ (failed LTE test) then
        $A_2^{t-1} = \alpha^{t-1}C_2 + \beta^{t-1}G_2$
        $A_2^{t-1} \leftarrow \mathrm{LU}(A_2)$
    else
        $A_2^{t-1} \leftarrow A_2^t$
    end if
    $x_2^{t-1} \leftarrow A_2^{t-1}\backslash(\beta^{t-1}G_{2p}x_p^{t-1} + (1-\beta^{t-1})G_{2p}x_p^{t-2} + ((1-\beta^{t-1})G_2 - \alpha^{t-1}C_2)x_2^{t-1}$
else
    Load $x_2^{t-1}$
end if
RETURN: $x_2^{t-1}$
```

Next we need to construct and factor the A_2 matrix, and compute the Jacobian stamp. However, if we are not at the initial step of Newton (i.e. $k > 0$), then we can reuse the previously computed $\bar{G}$ and factored A_2 because they will not have changed. Additionally, if we are at the first step of Newton, but neither the timestep nor the integration order have changed since the last step (i.e. $\alpha^t = \alpha^{t-1}$ and $\beta^t = \beta^{t-1}$), then we can reuse $\bar{G}$ and A_2 from the previous time step. This is shown in Algorithm 2

Algorithm 2 Construct and Factor A_{int}^t, and construct $\bar{G}$

```
INPUTS: $\alpha^t$, $\beta^t$, $\alpha^{t-1}$, $\beta^{t-1}$, $A_2^t$, $\bar{G}$, $C_2$, $G_2$, $G_{2p}$, $G_{p2}$
if $(k > 0) \parallel ((t > 0)\&\&(\alpha^t = \alpha^{t-1})\&\&(\beta^t = \beta^{t-1}))$ then
    Reuse previously factored $A_2$
    $A_2^t \leftarrow A_2^t$
    $\bar{G} \leftarrow \bar{G}$
else
    Update and refactor $A_2$
    $A_2^t \leftarrow \alpha^t C_2 + \beta^t G_2$
    $A_2^t \leftarrow \mathrm{LU}(A_2^t)$
    $\bar{G} \leftarrow -G_{p2}A2^t\backslash(\beta G_{2p})$
end if
RETURN: $\bar{G}$, $A_2^t$
```

Lastly, computing the residual stamp requires computing the new internal state x_2^t. Since this depends on the port values x_p^t, it will change at every Newton iteration and thus we must recompute the residual stamp at every call to the device. This is shown in Algorithm 3.

Algorithm 3 Compute x_2^t and $\bar{f}$

```
**INPUTS:** $x_2^{t-1}$, $x_p^{t-1}$, $x_p^t$, $\alpha^t$, $\beta^t$, $A_2^t$, $C_2$, $G_2$, $G_{2p}$, $G_{p2}$
$x_2^t \leftarrow A_2^t \backslash (\beta^t G_{2p} x_p^t + (1-\beta^t) G_{2p} x_p^{t-1} + ((1-\beta^t) G_2 - \alpha^t C_2) x_2^t$
$\bar{f} \leftarrow G_{p2} x_2^t$
**RETURN:** $\bar{f}$
```

A few things of note:

- Matrix notation here (e.g. A_2, G_{p2}) is consistent with notation used within the Xyce code
- In the above algorithms, A_2 is used to represent both the original matrix A_2 and its inverse $(A_2)^{-1}$. This is because LAPACK does the factorization of A_2 in place, so the contents of A_2 depends on where you are in the algorithm.
- In the above algorithms, we use the Matlab notation backslash '\' to represent solving a system. Since A_2 contains the LU factors of A_2, writing $A_2 * B$ or $(A_2)^{-1}B$ would be incorrect. This also help remove some ambiguity as to whether a particular A_2 is the original A_2 or the factored A_2.

Chapter 5

Examples

The transformation based approaches from Section 3 and the two-level solver approach from Section 4 were tested on several examples.

5.1 RC Ladder

To illustrate the benefits of transforming the reduced model coordinate system, we first examine an RC line circuit. In these examples we only consider scaling the reduced states in order to reduce the number of time steps taken during transient simulation. At the present time, sparse reduced models are not supported in Xyce, so sparsifying the system yields no performance speedup.

To begin, we construct a reduced model of order $q = 20$ using FDSVD that matches the frequency-domain behavior of the original system very well. Next, we examine the transient response of the large and reduced models in response to a pulse input. Figure 5.1 shows that the reduced model, simulated in Xyce, actually simulates slower than the original system. This is due to the increased number of time steps required, as described previously.

To remedy this time step issue, we apply the scaling procedure described in Section 3.1 to the reduced model. The resulting reduced model is then simulated with the same setup, and as seen in Figure 5.2, the number of time steps taken is drastically reduced, and as a result the reduced model now simulates faster than the full system.

5.2 RLC Ladder

The next example considered is an RLC ladder. As in the previous example, we create a reduced model using FDSVD method that matches the frequency behavior of the original system. We then perform a transient simulation of the reduced model in response to a pulse input, the results of which are shown in Figure 5.3. Due to the poorly scaled reduced model states, transient simulation of the ROM takes ten times more steps, and simulates slower than the original system.

After applying the scaling procedure from Section 3.1 to the reduced model, the number of timesteps required by the reduced model is drastically reduced, and the simulation performs faster than the full system, as shown in Figure 5.4.

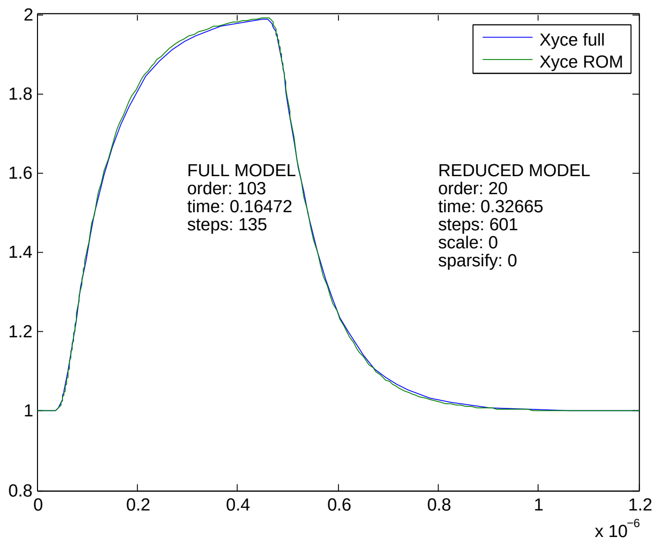

Figure 5.1. Transient response to pulse input for original RC circuit and reduced model. Note reduced model simulates slower than original system, and requires many more transient steps.

5.3 Voltage Regulator Circuit

This circuit contains parasitic cables connected to 18 nodes of a voltage regulator, corresponding to both measurement nodes and voltage supply nodes, resulting in a system with about $33,000$ unknowns. Using our previously developed techniques, we created ROMs for each of the cable models, resulting in a system with only $3,000$ unknowns. Simulating the new model using the scaling techniques described in 3.1, we were able to obtain a speedup of about $4x$. Using the two-level solver approach from Section 4 yields a further $4x$ speedup, for a total speedup of about $16x$. Figure 5.3 shows that the outputs of the circuit containing the ROM cable models (dotted lines) matches very accurately the output of the full circuit (solid lines).

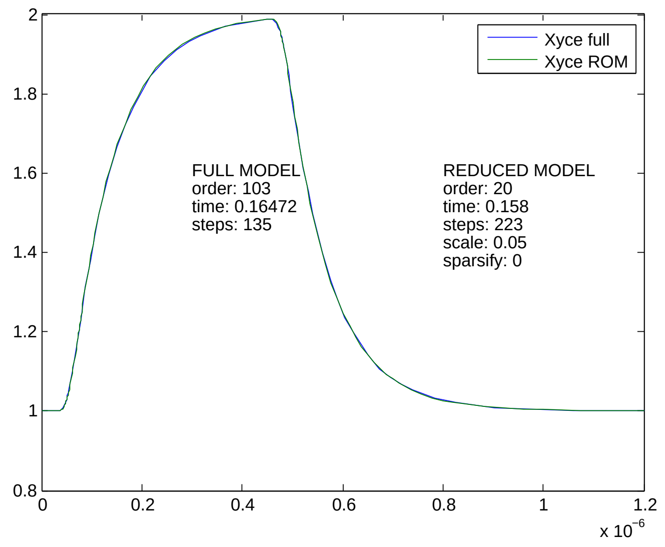

Figure 5.2. Order 20 reduced model

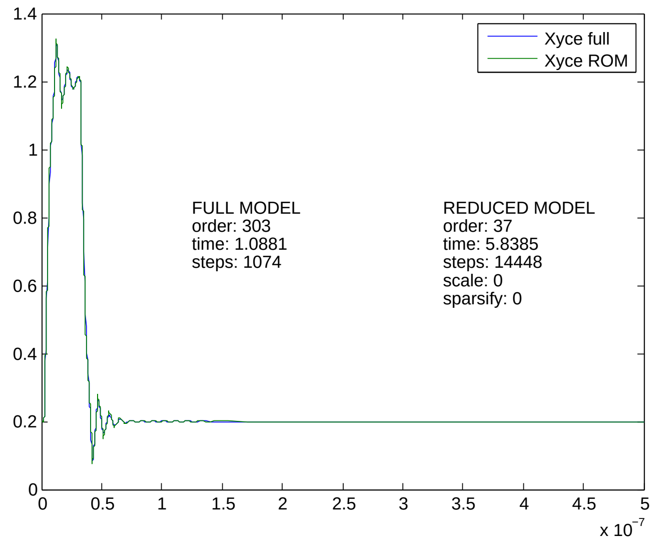

Figure 5.3. Transient response to pulse input for original RLC circuit and reduced model. Note reduced model simulates slower than original system, and requires many more transient steps.

*

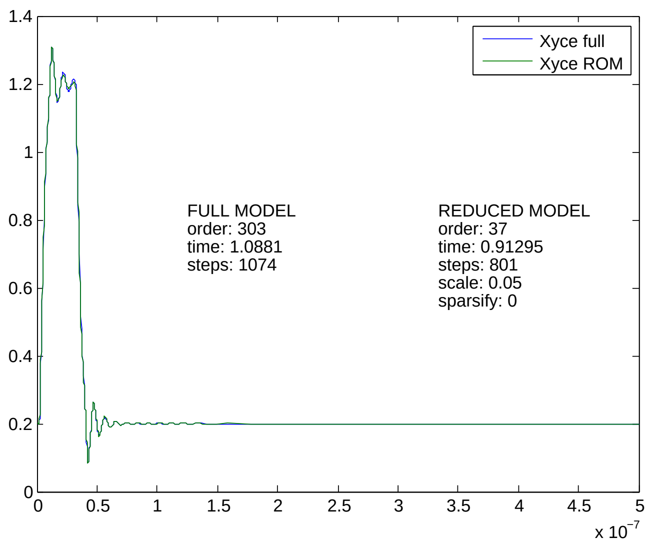

Figure 5.4. Reduced model with scaling.

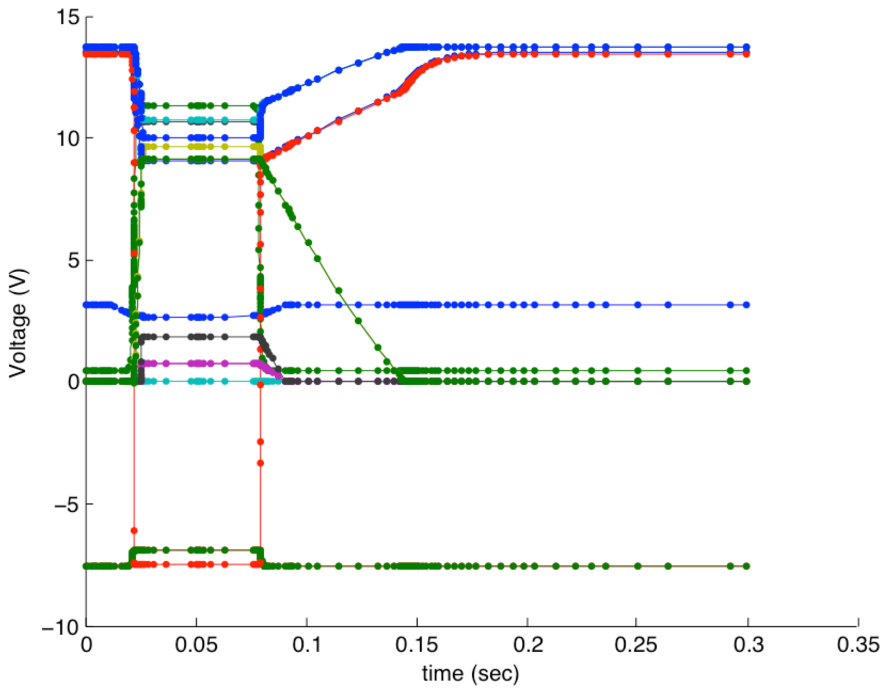

Figure 5.5. Simulation of voltage regulator circuit containing full cable models (solid lines) and the circuit prototype containing ROMs for the cables (dotted lines). The full circuit contained approximately $33,000$ unknowns, while the circuit using reduced models contained approximately $3,000$ unknowns, resulting in a transient simulation speedup of about $16x$.

Appendix A

Math Details: Preservation of the System under Transformation

In the scaling and sparisifcation methods presented above, we discussed both linear transformations on the state variables of the reduced model, and linear transformations on the projection matrices. In this section it will be shown that transforming the reduced system and the projection matrices are equivalent, that neither such transformation affects the accuracy of the resulting model, and that it is a trivial task to transform back to the original reduced state.

A.1 Transforming the Reduced Model

First we show that transforming the reduced system does not affect the input-output behavior of the system. Consider the reduced model with state $\hat{x}$

$$\hat{E}\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u, \quad y = \hat{C}^T\hat{x}.$$

The transformed system is created by changing coordinates to $\tilde{x}$, where $\hat{x} = P\tilde{x}$, and also left-multiplying the system by another transformation matrix Q (referred to by me as "shuffling" the equations), resulting in the following transformed system

$$Q\hat{E}P\dot{\tilde{x}} = Q\hat{A}P\tilde{x} + Q\hat{B}u \qquad\qquad y = \hat{C}^T P\tilde{x},$$

which we will write concisely as

$$\tilde{E}\dot{\tilde{x}} = \tilde{A}\tilde{x} + \tilde{B}u \qquad\qquad y = \tilde{C}^T\tilde{x},$$

where $\tilde{E} = Q\hat{E}P$, $\tilde{A} = Q\hat{A}P$, $\tilde{B} = Q\hat{B}$, and $\tilde{C} = P^T\hat{C}$. It can be shown that these two systems are identical in terms of input-output behavior by examining the transfer function. If the transfer function of the original system is

$$H(s) = L^T(sE - A)^{-1}B,$$

then the transfer function of the transformed system is

$$\begin{aligned}\tilde{H}(s) &= \tilde{L}^T(s\tilde{E} - \tilde{A})^{-1}\tilde{B} = \\ &= L^T P(sQEP - QAP)^{-1}QB = \\ &= L^T P\left(Q(sE - A)P\right)^{-1}QB \\ &= L^T PP^{-1}(sE - A)^{-1}Q^{-1}QB \\ &= L^T(sE - A)^{-1}B \\ &= H(s).\end{aligned}$$

Thus, any transformation to the reduced model (i.e. any choice of P, Q matrices) does not affect the input-output behavior of the model. Note: we are assuming P and Q are invertible, which is a reasonable assumption.

A.2 Transforming the Projection Matrices

Next, we will show that the system transformations discussed on the previous section, which was an operation directly on the reduced model, can instead be viewed as a transformation to the projection matrices, and thus applied to the projection matrices before the reduced model is created.

Consider the transformed reduced linear system

$$\tilde{E}\dot{\tilde{x}} = \tilde{A}\tilde{x} + \tilde{B}u$$

and expand out the matrices $\tilde{E}, \tilde{A}, \tilde{B}$, resulting in

$$Q\hat{E}P\dot{\tilde{x}} = Q\hat{A}P\tilde{x} + Q\hat{B}u.$$

Now expand $\hat{E}, \hat{A}, \hat{B}$, resulting in

$$Q(W^TEV)P\dot{\tilde{x}} = Q(W^TAV)P\tilde{x} + Q(W^TB)u.$$

Now regroup the terms

$$(WQ)^TE(VP)\dot{\tilde{x}} = (WQ)^TA(VP)\tilde{x} + (WQ)^TBu$$

and define new projection matrices $\tilde{W} = WQ$ and $\tilde{V} = VP$ such that the transformed linear system can be viewed as a projection of the original system with transformed projection matrices

$$\tilde{W}^TE\tilde{V}\dot{\tilde{x}} = \tilde{W}^TA\tilde{V}\tilde{x} + \tilde{W}^TBu.$$

As a result of the equivalence of these two approaches, we have the option of either integrating the transformations into the MOR procedure, so they are performed on the projection matrices, or applying them to the resulting ROM as a post-processing step. The latter is particularly useful as we may not always have access to the MOR procedure itself. For example, we may be given a reduced model created from some black box software.

References

[1] Eric R. Keiter, Thomas V. Russo, Eric L. Rankin, Richard L. Schiek, Keith R. Santarelli, Heidi K. Thornquist, , Deborah A. Fixel, Todd S. Coffey, and Roger P. Pawlowski. Xyce parallel electronic simulator: User's guide, version 5.1. Technical Report SAND2009-7572, Sandia National Laboratories, Albuquerque, NM, 2009.

[2] Eric R. Keiter, Thomas V. Russo, Eric L. Rankin, and Roger P. Pawlowski. Xyce parallel electronic simulator: Radiation models reference guide, version 5.1. Technical Report SAND2009-7324, Sandia National Laboratories, Albuquerque, NM, 2009.

[3] C. W. Ho A. E. Ruehli and P. A. Brennan. The modified nodal approach to network analysis. *IEEE Trans. Circuits Systems*, 22:505–509, 1988.

[4] Kartikeya Mayaram and Donald O. Pederson. Coupling algorithms for mixed-level circuit and device simulation. *IEEE Transactions on Computer Aided Design*, II(8):1003–1012, 1992.

DISTRIBUTION:

1 Al Lehnen
Mathematics Department
Madison Area Technical College
3550 Anderson Street
Madison, WI 53704

4 Brad Bond
2711 Blenheim Ave, Apt 3
Redwood City, Ca, 94063

1 Jaijeet Roychowdhury
Electrical Engineering and Computer Sciences Department
545E Cory Hall
University of California-Berkeley
Berkeley, CA 94720

1	MS 1318	Robert J. Hoekstra, 01426
1	MS 1320	Sivasankaran Rajamanickam, 01426
1	MS 1320	Mike A. Heroux, 01426
1	MS 1318	Bruce A. Hendrickson, 01440
1	MS 1323	Scott A. Hutchinson, 01445
4	MS 1323	Eric R. Keiter, 01445
1	MS 0316	Joseph P. Castro, 01445
1	MS 1323	Richard L. Schiek, 01445
1	MS 1323	Heidi K. Thornquist, 01445
1	MS 1323	Thomas V. Russo, 01445
1	MS 1318	Russell Hooper, 01445
1	MS 1322	Ting Mei, 01445
1	MS 1323	Eric Rankin, 01445
1	MS 1322	John B. Aidun, 01425
1	MS 1322	Richard P. Muller, 01425
1	MS 1320	Pavel B. Bochev, 01442
1	MS 1320	David M. Day, 01442
1	MS 0352	Charles E. Hembree, 01344
1	MS 0359	Donna L. Chavez, 01911
1	MS 0899	Technical Library (electronic copy), 9536
1	MS 0612	Review & Approval Desk, for DOE/OSTI, 9612

Sandia National Laboratories