Model You an Energy System for Great Good

Ryan Goodfellow

May 26, 2017

Contents

Abstract

This work presents an exploratory modeling technique and accompanying set of simulation methods for developing energy system models. The technique and methods are actually applicable for developing any model described as a system of differential-algebraic equations e.g., dynamical systems whose structural model is a network and obey certain constrains such as conservation laws. The technique allows a developer to start with a DAE model of potentially high index, and then iteratively work toward a lower index model. Uncovering so-called hidden assumptions about the model along the way. The simulation methods are purpose built with external discrete control in mind. The underlying goal is to enable a work flow where one models a physical system that are explicitly designed to be controlled externally over a computer network. This allows the co-development

of actual distributed control systems alongside physical system. External discrete control is quite challenging in the DAE simulation setting, specifically when a control action arrives that changes the structure of the underlying system. The two prongs of this work eventually merge when the modeling diagnostics are used during simulation runtime to diagnose issues with models as they cause issues during simulation – instead of just simply failing to converge on the next time step value.

This work presents the modeling technique and simulation engine by using them to develop increasingly sophisticated energy system models - presenting the theoretical mathematical underpinnings along the way.

1 | Introduction

Differential-algebraic equations are often considered one of the most natural ways to model physical systems. We simply describe the behavior of a set of components in terms of a set of variables and time-derivatives of those variables and connect them through equations. In the linear setting the system looks like (1).

$$Ex' + Fx = f \tag{1}$$

If the matrix E is non-singular then we have a so-called implicit ordinary-differential equation (ODE) that looks like (2), and we can treat it just like any other ODE system.

$$x' = E^{-1}(f - Fx) \tag{2}$$

However, most DAE systems are not simply implicit ODEs. The coefficient matrix of the derivatives E commonly has very low rank, precluding the simple rearrangement of terms into an ODE. The distance between an ODE system and a DAE system is called the index. All simulation engines built specifically for DAEs are guaranteed to work for systems with an index of 1 or less [3, 1, 9, 7, 10]. A select few work on index-2 [4], and there are some very specialized techniques for particular domains that work on index-3. In the literature high-index systems are commonly referred to as *ill-posed* [8] systems. One commonly used index, the differentiation index gives a clear picture of the difficulties imposed by high-index. The differentiation index, defines the index of (1) as the number of times the system must be differentiated to arrive at an equation that uniquely defines the derivatives. However, the system we arrive at does not necessarily obey the constraints of the original. Many of the solution techniques referenced above can guarantee that at index-1 or -2, the solution computed will obey the constraints.

This situation then begs the question, when developing a DAE based model of a system, how can we check the model as we are developing it, if it is of index-1. This question is answerable in a number of ways and has been known for numerous index definitions for quite some time. The follow-on question is the focus of the current work. *Given that a model is of index greater than 1, how can we relate the high index to the specific points in our model that are causing difficulty and what can be done to fix it.* This work makes use of the *tractability index* for answering these questions. The theory underlying the tractability index offers broad insight into the structure of a DAE system - in the constant and time varying coefficient case as well as in the nonlinear setting.

The contribution of this work is two fold. First is the approach - how the tractability index and underlying algorithms are used to create a modeling workflow that guides the model developer towards sound *models and compositions of models that are guaranteed to work under simulation. The second are the tools and models themselves. The tools presented here take software form and are freely available as an open-source suite software called Cypress [6, 11] . The models presented are a part of the Cypress energy systems library.*

1.1 Outline

This paper starts with very simple circuits. The techniques and theory are presented with these simple models first because they are easy to reason and think about. Then the technique is used to grow increasingly sophisticated energy system models directly from the humble circuit beginnings.

1.2 Related Work

The work in [5] served in many ways as the jumping off point for this work. It is in this work that we really got a feel for the application of the tractability index to energy problems. We believe that the work in this paper compliments the work in [5] in the following way. In [5] the main idea was to develop locally-verifiable sufficient conditions for models of circuit elements and their compositions to be either of index-1 or -2. The authors provide a far reaching range of conditions that are indeed sufficient but not necessary. Other conditions may arise that allow for low index systems. The goal of the present work is taken from the opposite direction. Instead of starting with low index conditions and finding models that satisfy them, we start with higher-index models and find ways to lower the index.

2 | Circuit Beginnings

This section will first present the theory and mathematical machinery by example - using a simple model of a voltage source transmitting energy over a pi-model transmission line to a resistive load. This circuit is depicted in Figure 1. The first thing we need is a DAE formulation of this circuit. One way to get there is through the modified nodal analysis (MNA) of the circuit. MNA is described in the first subsection. Once we have the DAE in hand from MNA, the following subsection will introduce the tractability index and compute the index of the DAE for the circuit in Figure 1.

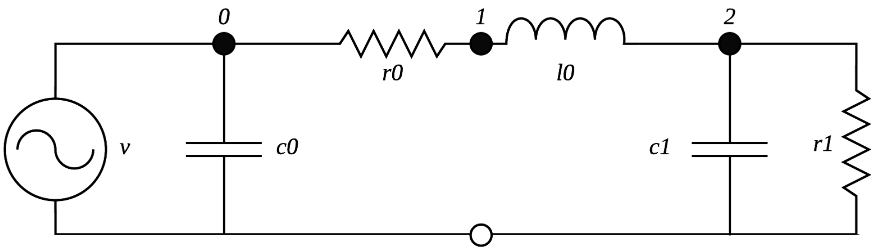

Figure 1: Circuit 0

It turns out that this circuit, as simple as it is, is actually of index 2. We will show how the mathematical machinery underlying the computation of the tractability index for this circuit can be used to point a spotlight on the troublesome parts of the model. This is the first step in finding a path to physically informed index reduction. Then we will show specific ways in which the index-2 property of this model causes problems namely external discrete control and structural dynamics. Then it will be shown that lower index model pointed to by the index reductions does not exhibit the same difficulties under simulation as the index-1 version and physical and a connection between the chosen index reduction and the underlying physics will be established.

The workflow just described shall become a theme for the entire rest of the paper as we build more and more sophisticated models and is captured concisely in Algorithm 1.

Algorithm 1 Workflow

```
Model
Project
Analyze
Refine
```

2.1 The Modified Nodal Analysis DAE

The MNA of a linear circuit produces a linear differential-algebraic system with time-varying coefficients of the form.

$$E(t)x' + F(t)x = f(t) \tag{3}$$

The MNA formulation of this equation is as follows. Note that we briefly walk through this here not because we feel that the MNA construction of (3) itself is noteworthy, but due to the fact that we will be using specific components from this construction in the mathematics that follows. In the notation that follows we drop the t parameter from all matrices to make reading easier.

The MNA-DAE for a linear circuit is constructed from two types of matrices, the *topological matrices* and *component value matrices*. The topological matrices are denoted as T_x for $x \in \{C, L, G, V, I\}$ (capacitance, inductance, conductance, voltage source, current source). For these matrices the columns represent branches and the rows represent nodes in the circuit and the total circuit's topology is represented by the matrix $T = (T_C, T_L, T_G, T_V, T_I)$. Each branch typically has 2 entries, 1 for each connection point. The nonzero entry for which current is entering the branch is equal to -1 and the other is equal to 1. The case where there is only one nonzero entry is when one of the connection points is the reference node. All columns in the complete topology matrix sum up to zero as per KCL. The *component value matrices* denoted as one of C, L, G for capacitance, inductance, and conductance respectively are diagonal matrices where each entry represents the capacitive, inductive or conductive value of the referenced component. We can now state the MNA-DAE formulation as the following.

$$\begin{pmatrix} -T_C C T_C^T & 0 & 0 \\ 0 & L & 0 \\ 0 & 0 & 0 \end{pmatrix} x' + \begin{pmatrix} -T_G G T_G^T & T_L & T_V \\ -T_L^T & 0 & 0 \\ T_V^T & 0 & 0 \end{pmatrix} x = \begin{pmatrix} -T_I i() \\ 0 \\ T_V^T v() \end{pmatrix} \tag{4}$$

2.2 Index Calculation

The first step in performing a simulation of a DAE system is determining the index. There are many definitions of index, here we are using the *tractability index* [8]. This index measures the number of steps that are required to project a DAE system into a subspace where the system becomes completely decoupled as an ODE system and a purely algebraic system. In this form, a simulation can take place by solving the ODE system and plugging the corresponding values into the algebraic system. If the index of the system turns out to be higher than 2, then it is not guaranteed that a simulation will be successful.

The first piece of mathematical machinery that must be presented is the projection algorithm itself [8], see Algorithm 2 below. The function `next_projector` is highly technical and not necessary at this point. It will be presented later in this paper.

Algorithm 2 Projection Algorithm

```
$G_0 := E$
$B_0 := F$
$Q_0 :=$ next_projector()                    ▷ a projector onto ker $G_0$
$P_0 := I - Q_0$
for $i \geq 1$ do
    $G_i := G_{i-1} + B_{i-1}Q_{i-1}$
    if $|G_i| \neq 0$ then stop
    $B_i := B_{i-1}P_{i-1}$
    $Q_i :=$ next_projector()                ▷ a projector onto ker $G_i$
    $P_i = I - Q_i$
```

This algorithm starts with an index of 0, and each iteration of the for loop increments the index. The algorithm terminates when G_i has full rank where i is simultaneously the current index in both the tractability sense and the for loop iteration sense. For a DAE (1) to be index-1, this algorithm dictates that $G_1 = E + FQ$ have full rank. Here Q is a projector onto the nullspace of E. Since we are concerned precisely with what the conditions are that are causing an index higher than 1, we begin our investigation with this matrix.

It turns out that the nullspace of G_1 will point directly at the problem places in the circuit given the context of the construction of (4). To compute a basis for this nullspace we employ the SVD. As a reminder the SVD is a matrix composition that produces a basis for the nullspace, rowspace, columnspace and left nullspace of a matrix. At the moment we are only concerned with the nullpsace. Given the SVD composition $A = U\Sigma V$ for $A \in \mathbb{R}^{m\times n}$ with $r = \texttt{rank}(A)$, a basis for the nullspace of A is given by $V[0{:}m,\ r{:}n]$. The columns of this nullspace point directly to linearly dependent sets in the matrix G_1 and thus give us our first pointer as to where problem areas are in the model. In fact, for the initial projector Q_0 from Algorithm 2, we simply use $V[0{:}m,\ r{:}n]V[0{:}m,\ r{:}n]^T$. Subsequent projectors have further requirements in terms of how they relate to previous projectors and will be discussed later.

Example 2.1. Consider the circuit from Figure 1. For the parameters

c0	2.2e−3
c1	2.5e−3
rt	4
lt	3e−2
rl	7

this circuit has the following DAE-NMA (4) formulation.

$$\begin{bmatrix} -0.0022 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.0025 & 0 & 0 \\ 0 & 0 & 0 & 0.03 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} x' + \begin{bmatrix} -0.25 & 0.25 & 0 & 0 & 1.0 \\ 0.25 & -0.25 & 0 & -1.0 & 0 \\ 0 & 0 & -0.143 & 1.0 & 0 \\ 0 & 1.0 & -1.0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 \end{bmatrix} x \; = \; f()$$

Here the projector $Q = \mathtt{ker} G_0$ used in the initial projector iteration and the subsequent G_1 are the following.

$$Q \triangleq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 \end{bmatrix} \qquad G_1 = \begin{bmatrix} -0.0022 & 0.25 & 0 & 0 & 1.0 \\ 0 & -0.25 & 0 & 0 & 0 \\ 0 & 0 & -0.0025 & 0 & 0 \\ 0 & 1.0 & 0 & 03 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

which leads to a nullspace basis of G_1 by way of SVD as [1]

$$\begin{bmatrix} 0.999997580008784 \\ 0 \\ 0 \\ 0 \\ 0.0021999946760193 \end{bmatrix}$$

This tells us that in G_1 the 1st and 5th columns are linearly dependent, leading the rank deficient condition of G_1. Examining those two columns within the context of (4) we see that the contributing factors to column 1 are column 1 of the $-T_C C T_C^T$ submatrix of E and column 1 of the $-T_G G T_G^T$ submatrix of F. However the latter is nullified by Q. Likewise the contributing factors for column 5 come exclusively from the T_V submatrix of F. Thus the linear dependence has to do exclusively with the relationship between the first column of $-T_C C T_C^T$ e.g., the $\mathtt{c}_0$ branch of the circuit, and, the first column of $-T_V$ e.g., the voltage source branch of the circuit. ***The nullspace of G_1 has pointed us directly at the problem in the model.*** This agrees with the result from [5] that a CV-loop leads to an index-2 DAE from the MNA formulation. We have just come at it from a different angle. We should note that the conclusion we have drawn here is not nearly as strong as that in [5], in that work the authors characterized CV-loops as a general topological subcomponent that *always* leads to an index-2 MNA-DAE. In this case we have only shown a result for a particular circuit. But that is really what we are after here, tools and techniques that help us look

[1] values below machine precision set to zero

at particular models in the absence of broadly sweeping theory that can help. It is our hope that these investigative tools will help uncover such theory.

The takeaway from the nullspace of G_1 is that the relationship between the voltage source and the capacitor c_0 has a hidden assumption. In this case we can see that the relationship is overly ideal. In reality we have neither transmission mediums not capacitive elements that have zero underlying resistance. One way to establish a resistive link along the path in question is to augment the voltage source model with a resistor as depicted in Figure 2. Indeed, if we run this modified circuit back through the projection algorithm we find that the algorithm terminates after 1 iteration finding a G_1 of full rank.

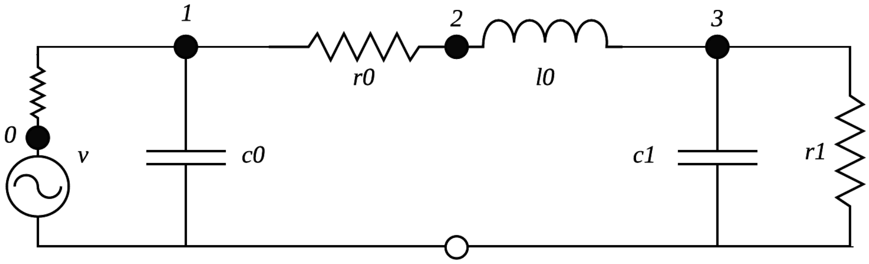

Figure 2: Single Phase Transmission Circuit - Index 1

The idea of making circuit elements less ideal is nothing new. Circuit simulation programs have been doing this for years to make solution progression smoother. We just take this opportunity to point out that our method in a very simple case leads to a well established practice. Now let us take a look at how this simple circuit model behaves under structurally dynamic simulation.

2.3 Simulation

In this section we simulate the index-1 version of the Pi-Line circuit using the projector based technique. First a mathematical characterization of the projected decoupled system will be given. Then we will discuss how the projector method can be used to identify places in the model that create an index-2 condition. Then we will show how to compute consistent initial conditions using the projector based method and convex optimization in a general way. Finally we will present the simulation technique of the projected subsystem using standard integrators on the underlying ODE.

2.3.1 Projected Systems

Before diving into simulation, let's take a look at the mathematical description of the projected systems created by Algorithm 2.[2] This is a projection of the original DAE onto a subspace where we wind up with an underlying ODE and a decoupled algebraic system.

$$\left(\begin{array}{c|cc} I & 0 & 0 \\ \hline 0 & 0 & \mathcal{N}_{01} \\ 0 & 0 & 0 \end{array}\right) \left(\begin{array}{c} u' \\ \hline 0 \\ v_1' \end{array}\right) + \left(\begin{array}{c|cc} \mathcal{W} & 0 & 0 \\ \hline \mathcal{H}_0 & I & 0 \\ \mathcal{H}_1 & 0 & I \end{array}\right) \left(\begin{array}{c} u \\ \hline v_0 \\ v_1 \end{array}\right) = \left(\begin{array}{c} \mathcal{L}_d \\ \hline \mathcal{L}_0 \\ \mathcal{L}_1 \end{array}\right) f \tag{5}$$

[2] We limit ourselves to systems of index 1 and 2 here, for fully general expressions of arbitrarily high index see [8]

Here the row above the horizontal line is the inherent underlying ODE, the first row below represents the constraints and the following rows represent index equations. The nonzero elements in the columns to the left of the horizontal line represent coupling coefficients linking the algebraic and index equations back to the underlying ODE.

From this system we derive the following index-1 and index-2 systems in a form amenable to simulation.

$$\begin{aligned} u' &= \mathcal{L}_d f - \mathcal{W} u \\ v_0 &= \mathcal{L}_0 f - \mathcal{H}_0 u \end{aligned} \tag{6}$$

$$\begin{aligned} u' &= \mathcal{L}_d f - \mathcal{W} u \\ v_0 &= \mathcal{L}_0 f - \mathcal{N}_{01} v_1' - \mathcal{H}_0 u \\ v_1 &= \mathcal{L}_1 f - \mathcal{H}_1 u \end{aligned} \tag{7}$$

Using these systems, we can see that to find a solution we can use an ODE solver to solve the inherent underlying ODE for some initial condition, plug and plug the resulting values into the constraint and index systems to solve the projected system.

The relationship between the values in the projected equations and the original are as follows.

	index-1	**index-2**
u	$\Pi_0 x$	$\Pi_1 x$
$v0$	$Q_0 x$	$Q_0 x$
$\mathcal{L}_d$	$\Pi_0 G_1^{-1}$	$\Pi_1 G_2^{-1}$
$\mathcal{L}_0$	$Q_0 G_1^{-1}$	$Q_0 P_1 G_2^{-1}$
$\mathcal{L}_1$		$\Pi_0 Q_1 G_2^{-1}$
$\mathcal{H}_0$	$\mathcal{L}_0 B_1$	$\mathcal{L}_0 B_2$
$\mathcal{H}_1$		$\mathcal{L}_1 B_2$
$\mathcal{W}$	$\mathcal{L}_d B_1$	$\mathcal{L}_d B_2$
$\mathcal{N}_{01}$		$-Q_0 Q_1$

The solution in terms of the projection is the following.

$$x = \sum_{i}^{\mu-1} v_i + u \tag{8}$$

Thus once we have the projected values, we can unproject back into the originating problem space to find our solution.

2.3.2 Consistent Initial Conditions

As straight forward as it sounds, computing a consistent initial condition is far from simple. This is the topic of our next section. The initial condition problem for (1) can be formulated as

$$(E + F) \begin{pmatrix} x \\ x' \end{pmatrix} = f \tag{9}$$

where $(E+F) \in \mathbb{R}^{n\times 2n}$, so (9) is underdetermined by a factor of 2. Taking a look at the projected system (6), the situation does not get much better. The initial condition problem for this system is

$$\begin{pmatrix} I & \mathcal{W} & 0 \\ 0 & \mathcal{H}_0 & I \end{pmatrix} \begin{pmatrix} u' \\ u \\ v_0 \end{pmatrix} = \begin{pmatrix} \mathcal{L}_d f \\ \mathcal{L}_0 f \end{pmatrix} \tag{10}$$

which is underdetermined by a factor of 1.5. The underdetermination factor of 2 or less leads us to believe that if we supply either x' or x completely, then we can compute the other. However, even just determining a consistent x' or x in a general way is nontrivial. For the differentiation index and corresponding derivative array methods, we have the approaches described in [2], but it is unclear how they map into the projected subspace setting.

In the projector based setting we can compute the initial conditions using a few additional tools. First we introduce a new matrix D. This comes from the factorization $E = AD$ and is known as the *properly stated leading term*. We will not get into the underlying theory here see [8], but suffice it to say that this factorization may be computed as the following

$$A := E \tag{11}$$

$$D := E^{-}E \tag{12}$$

where E^{-} is the pseudo-inverse of E. Furthermore the factorization has the property that

$$\text{im}\, D \oplus \ker A = \mathbb{R}^n \tag{13}$$

The partitioning (13) leads naturally to a *projector* R that has the properties

$$\text{im}\, R = \text{im}\, D$$
$$\ker R = \ker A$$

and may be computed as

$$R := DD^{-} \tag{14}$$

Given (12), and (14) we can now define the following equation system that allows us to compute a consistent set of initial conditions for (3).

$$\begin{aligned} Ey + Fx &= 0 && x \text{ and } y \text{ solve the system} \\ D(x-s) &= 0 && \text{constrain } x \text{ to the projection of } s \\ (I-R)y &= 0 && \text{constrain } y \text{ to be in the image space of } D \end{aligned}$$

Notice that the 3rd constraint forces y into the kernel of $(I - R) \equiv \ker A$ and $\ker A \oplus \text{im}\, D$, so y is thus constrained into the image of D. This equation system condenses into (15) [8].

$$\begin{pmatrix} E & F \\ I-R & D \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} f \\ Ds \end{pmatrix} \tag{15}$$

This is actually quite a wonderful little equation system. As a researcher working with a particular DAE system it allows one to directly tinker with differing initial values directly by manipulating

s. Looking back to the example problem of the index-1 Pi-Line transmission circuit we can see how this plays out.

Starting with an initial guess of the zero vector (15) gives

$$x'(0) = \begin{pmatrix} 0.0 \\ 45.\overline{45}\texttt{e8} \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix} \quad x(0) = \begin{pmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ \texttt{1e7} \end{pmatrix}$$

Now while these values do indeed solve the system mathematically, having 10 million amps coming out of the voltage source is probably not a physically satisfying solution to the problem. A frequency domain analysis of the index-1 Pi-Line transmission circuit tells us that the initial values for the state variables should be the following, where the x' has been computed by plugging the frequency domain steady state value into (15)

$$x'(0) = \begin{pmatrix} 0.0 \\ 0.0028109 \\ 0.0 \\ 181.9415761 \\ 315.3438832 \\ 0.0 \end{pmatrix} \quad x(0) = \begin{pmatrix} 10.0 \\ 9.9999997 \\ 8.6459412 \\ -0.8143753 \\ 0.3385146 \\ 0.3385208 \end{pmatrix} \tag{16}$$

This is a much more appealing condition physically. It also suggests the notion that in many cases we will want to constrain the initial condition to be something physically reasonable. For systems with physical limitations, a good first shot at this may be an initial condition with a minimal least squares norm. This problem can be formulated as minimizing the objective function in (17)

$$\begin{aligned} \mathcal{A} &:= \begin{pmatrix} E & F \\ I - R & D \end{pmatrix}^{-1} \\ x &= \mathcal{A}b \\ &= \begin{pmatrix} \mathcal{A}_0 & \mathcal{A}_1 \end{pmatrix} \begin{pmatrix} b \\ Ds \end{pmatrix} \\ &= \mathcal{A}_0 b + \mathcal{A}_1 Ds \\ J &= ||\mathcal{A}_0 b + \mathcal{A}_1 Ds||^2 \end{aligned} \tag{17}$$

Minimizing this objective function using a convex optimizer will yield the minimal least squares initial condition s for (15). For the Pi-Line transmission example this is computed as

$$s = \begin{pmatrix} 0.0 \\ 9.99999909094478 \\ 0.0 \\ 6.36205080176246 \\ 0.909059625276132 \\ 0.0 \end{pmatrix}$$

Which yields the the following initial conditions via (15)

$$x(0)' = \begin{pmatrix} 0.0 \\ -0.00200065022165125 \\ 0.0 \\ 0.0780900097405279 \\ 0.0569929359261738 \\ 0.0 \end{pmatrix} \quad x(0) = \begin{pmatrix} 10.0 \\ 9.99999909094478 \\ 6.36376058984025 \\ 6.36205080176246 \\ 0.909059625276132 \\ 0.909055223985433 \end{pmatrix}$$

from here on we will refer to this simulation starting point as the ***minimum energy initial condition***.

2.3.3 Simulation

The simulation engine design for a projector based simulator is shown in figure 3

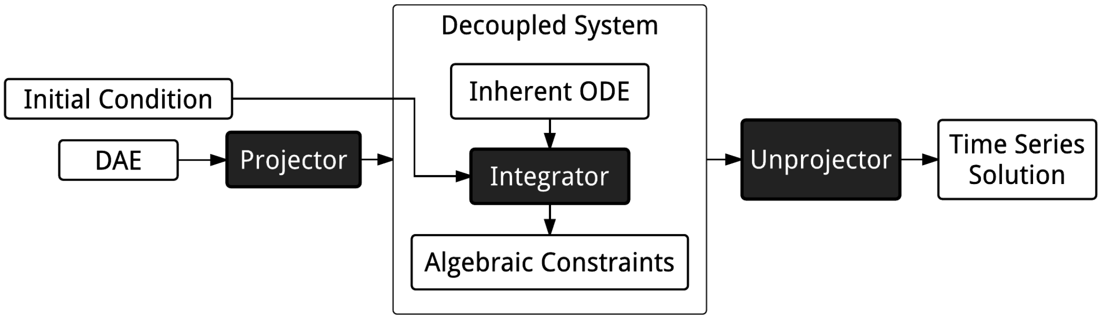

Figure 3: Projector Based Simulation Engine Design

The DAE and the initial condition (or approximation thereof) are fed to the simulation engine's interface. The simulation engine then computes the projection of the system. At this point let us restrict our attention to index-1 systems. The decoupling realized by the projector is composed of the inherent underlying ODE system and a system of algebraic constraints.

Once the decoupled system has been computed. The simulation engine integrates the inherent underlying ODE and feeds the results into the algebraic constraints. This results in time series values for u and v_0 as in (6). The unprojector is then simply a matter of evaluating (8). The result of the unprojector is a time series solution of the input DAE.

Experience with performing projection based simulations thus far shows that implicit, predictor-corrector style integrators seem to work best - particularly the trapezoidal integrator. Explicit integrators will fail in many cases, producing extremely unwieldy solutions or requiring extremely small time steps. The result of simulating the Pi-Line transmission model using a trapezoidal integrator with constant time step $1\mathtt{e} - 4$ starting from the frequency domain steady state condition is shown in figure 4

Figure 5 shows the same simulation starting from the automatically computed minimum energy initial condition.

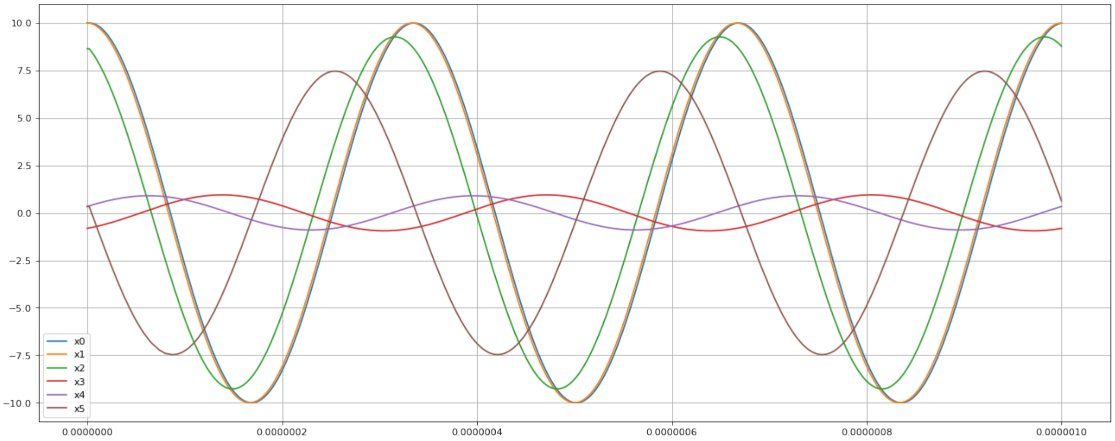

Figure 4: Projector Based Pi-Line Simulation Result Using Trapezoidal Integration

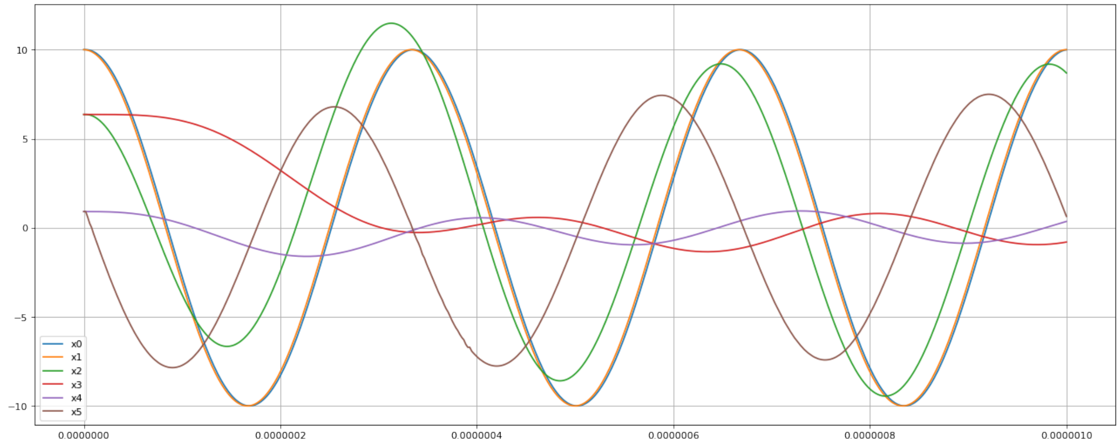

Figure 5: Projector Based Pi-Line Simulation Starting from Minimal Energy Condition

Notice here that the $x3$ value corresponding the the voltage on node 4 seems to be off. This is because in the convex optimizer setup that was initially used, a condition was present that forced all solution components into being positive. If this constraint is taken away, and instead a constraint is put in place that represents reasonable absolute value bounds, the initial condition conforms to a solution closer to the frequency domain steady state condition. Also note that over the course of the simulation, the solution converges to one that is equivalent to the one that stared from frequency domain steady state.

3 | Generic DAEs

While the MNA is wonderful for circuit analysis and simulation, it is not sufficient to model energy systems. The MNA presents an optimal systematic representation of Kirchoff's current and voltage laws. However, energy systems are comprised of more than electrical behavior. Electromechanics are present through the rotating action of synchronous machines, and electromagnetics come into play through magnetic coupling. Whats more, as new technologies come into the grid for energy production, storage and consumption – the physical phenomena that must be captured mathematically for the purposes of simulation will continue to become more heterogeneous. This motivates the use of a more general DAE formulation that can transcend physical domains of interest.

3.1 Flow Model

The generic DAE formulation that is most widely used for the purpose of inter-domain modeling is the flow model DAE. Indeed, this formulation may be viewed as a generalization of Kirchoff's voltage and current laws. The flow model views a DAE as a mathematical system whose structural model is a network. Within this network there are nodes and links. Each node encapsulates a group of variables. Links between nodes from a relationship between the variables each node encapsulates. The nature of this relationship depends on the type of the variable.

In the flow model there are two types of variables, *basic* and *flow*. When basic variables are linked, the relationship is defined as an equivalence relation. When flow variables are linked, the relationship is defined as a zero-sum. For example, in the electrical space, each node is comprised of 1 voltage and 1 current variable. Voltage variables are the basic type, a connection between two voltage nodes means the voltage at those nodes is the same. Current variables are the flow type, a connection between current nodes follows the zero-sum KCL law.

What gives the flow model it's generality is it's 3rd and final abstraction the *component*. A component is a sort of super-node. It encapsulates nodes and defines a mathematical relationship between them through equations. Users of Modelica will be familiar with this concept as a Modelica class. Modelers can define virtually any behavior[3] within a component, including nonlinear differential-algebraic equations.

The flow model is formulated as follows

$$f(x', x, t) = \begin{pmatrix} f_{\text{component}}(x', x, t) \\ f_{\text{basic}}(x', x, t) \\ f_{\text{flow}}(x', x, t) \end{pmatrix} = 0 \tag{18}$$

where

$$\begin{array}{ll} f_{\text{component}} & \in \mathbb{R}^n \times \mathcal{D}_f \times \mathcal{I} \to \mathbb{R}^m \\ f_{\text{basic}} & \in \mathbb{R}^n \times \mathcal{D}_f \times \mathcal{I} \to \mathbb{R}^s \\ f_{\text{flow}} & \in \mathbb{R}^n \times \mathcal{D}_f \times \mathcal{I} \to \mathbb{R}^l \end{array}$$

n : total number of variables
m : number of component equations
l : number of links connecting flow variables
s : number of link segments connecting basic variables

A link is a substrate that connects any number of nodes, and a segment is a connection from a node to a link. For example, consider the network depicted in Figure 6. This network has 1 link and 3 segments. Each link naturally imposes s basic equations as that is the number of equivalences between basic variables, but only a single flow constraint equation, as it is the zero sum from all the segments.

The flow model results in a larger number of variables and equations than a carefully constructed formulation such as the MNA where a minimization of the number of variables and equations is a principle objective. The principle objective of the flow model is generality.

[3] with the notable exception of partial differential equation based behavior

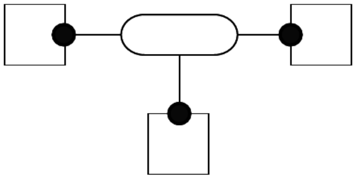

Figure 6: A Simple DAE Flow Model Network

In order to begin working with (18) in the projector setting for purposes of simulation, we first need a linearization. The linearization of (18) is formulated as

$$F_{x'} \Delta x'_* + F_x \Delta x_* = -f(x'_*, x_*, t) \tag{19}$$

where x_* and x'_* are working guesses at a solution and the Δ quantities are corrections to the working guesses. $\nabla_{x'}$ and ∇_x are the Jacobians defined by

$$F_{x'} = \nabla_{x'} f(x'_*, x_*, t) = \begin{pmatrix} \frac{\partial f_0}{\partial x'_0} & \cdots & \frac{\partial f_0}{\partial x'_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x'_0} & \cdots & \frac{\partial f_m}{\partial x'_n} \end{pmatrix} \quad F_x = \nabla_x f(x'_*, x_*, t) = \begin{pmatrix} \frac{\partial f_0}{\partial x_0} & \cdots & \frac{\partial f_0}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_0} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

A solution to (19) can readily be found using iterative approximation techniques such as Newton-Krylov or Newton-Raphson. We will come back to performing simulations on (18) using (19) in § 3.2.

3.2 Nonlinear Projection

The nonlinear projection simulation process is a combination of all of the mathematics presented thus far. The best way to describe this is through a mathematical diagram. Consider Figure 7.

The nonlinear projection process starts with a guess (x'_{*0}, x_{*0}). This guess is then fed into the residual function f at time $t = 0$. If the residual does not evaluate within an acceptable tolerance of 0 then we must compute a corrector for our guess. That corrector is determined by first projecting into the decoupled subspace. Then we formulate the corrector problem in the same way we formulated the consistent initial condition problem earlier – completing the under-determined system with subspace constraints (15), formulating an objective function (17) for the desired condition and computing the optimal value for the stated objective. In this case the output of the optimization is a corrector four our guess. This process repeats until the residual function is within an acceptable tolerance of zero. In the linear setting we introduced the minimal energy condition to help with staying inside physical limits. That goal is shared here too, however, now we would also like to add more information to the objective function so that we can optimize nonlinear approximation convergence time as well.

When the residual has reached a value that is within an acceptable tolerance of zero, we are ready to integrate to the next time point. This integration procedure is the same one that was carried out

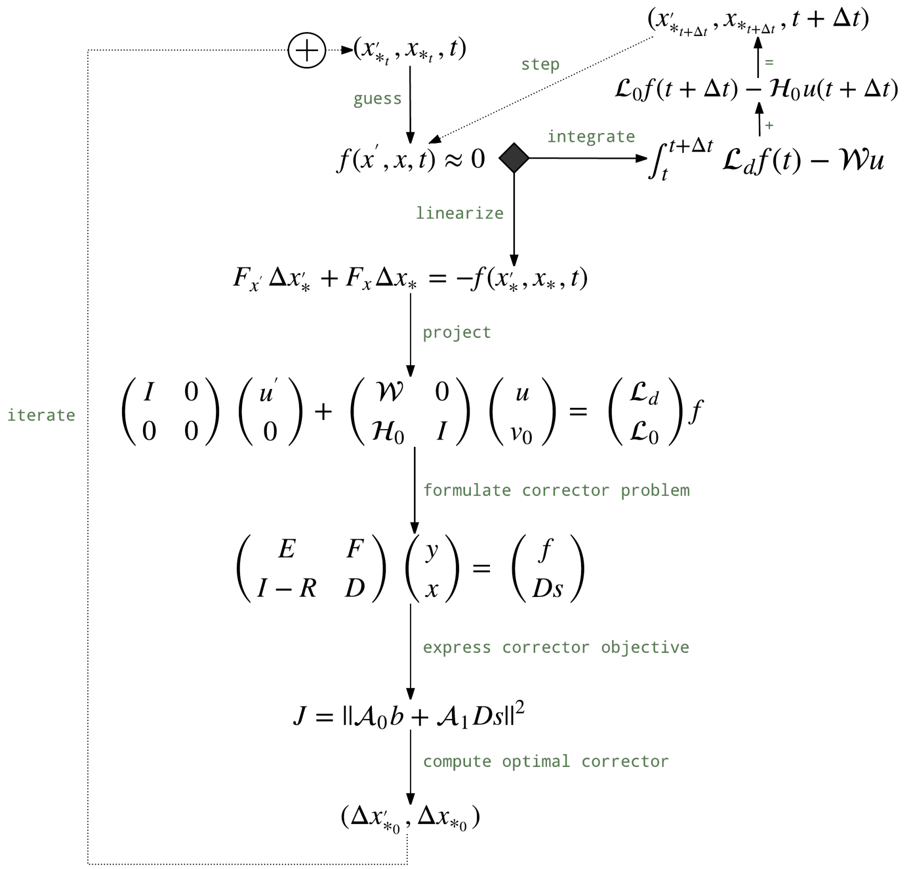
$(x'_{*_{t+\Delta t}}, x_{*_{t+\Delta t}}, t+\Delta t)$
(x'_{*_t}, x_{*_t}, t)
step
=
$\mathcal{L}_0 f(t+\Delta t) - \mathcal{H}_0 u(t+\Delta t)$
guess
+
integrate
$f(x', x, t) \approx 0$
$\int_t^{t+\Delta t} \mathcal{L}_d f(t) - \mathcal{W}u$
linearize
$F_{x'} \Delta x'_* + F_x \Delta x_* = -f(x'_*, x_*, t)$
project
iterate
$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u' \\ 0 \end{pmatrix} + \begin{pmatrix} \mathcal{W} & 0 \\ \mathcal{H}_0 & I \end{pmatrix} \begin{pmatrix} u \\ v_0 \end{pmatrix} = \begin{pmatrix} \mathcal{L}_d \\ \mathcal{L}_0 \end{pmatrix} f$
formulate corrector problem
$\begin{pmatrix} E & F \\ I-R & D \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} f \\ Ds \end{pmatrix}$
express corrector objective
$J = \|\mathcal{A}_0 b + \mathcal{A}_1 Ds\|^2$
compute optimal corrector
$(\Delta x'_{*_0}, \Delta x_{*_0})$

Figure 7: Nonlinear Projection Process

in the linear case, the only difference being we are integrating a linearization of the original DAE about time point t.

One critical piece to note here is that the structure of the DAE can now vary with time e.g., the $F_{x'}$ and F_x matrices vary with time. This is what we call a *structurally dynamic system*. Structural dynamics will be covered in detail in § 3.4

3.3 DyPy Framework

3.4 Structural Dynamics

References

[1] Ascher, U. M., and Petzold, L. R. Projected Implicit Runge–Kutta Methods for Differential-Algebraic Equations.

[2] Brenan, K., Campbell, S., and Petzold, L. *Numerical solution of initial-value problems in differential-algebraic equations*. 1995.

[3] Brown, P. N., Hindmarsh, A. C., and Petzold, L. R. Using Krylov Methods in the Solution of Large-Scale Differential-Algebraic Systems. *SIAM Journal on Scientific Computing 15*, 6 (11 1994), 1467–1488.

[4] Estévez Schwarz, D., and Lamour, R. Projector based integration of DAEs with the Taylor series method using automatic differentiation. *Journal of Computational and Applied Mathematics 262* (5 2014), 62–72.

[5] Estévez Schwarz, D., and Tischendorf, C. Structural analysis of electric circuits and consequences for MNA. *International Journal of Circuit Theory and Applications 28*, 2 (3 2000), 131–162.

[6] Goodfellow, R., and Kline, E. Cypress: A Testbed for Research in Networked Cyber-Physical Systems. In *10th International Conference on Testbeds and Research Infrastructures for the Developemnt of Networks and Communicites (TRIDENTCOM)* (Vancouver, Canada, 2015).

[7] Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., and Woodward, C. S. SUNDIALS. *ACM Transactions on Mathematical Software 31*, 3 (9 2005), 363–396.

[8] Lamour, R., März, R., and Tischendorf, C. *Differential-Algebraic Equations: A Projector Based Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[9] Mattsson, S. E., and Söderlind, G. Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives. *SIAM Journal on Scientific Computing 14*, 3 (5 1993), 677–692.

[10] Petzold, L. A description of DASSL: A differential/algebraic system solver. *Scientific computing* (1982).

[11] Ryan Goodfellow. Cypress Software Repository. https://github.com/rcgoodfellow/cypress, 2016.