SCREAMER: A Optimized Pulsed-Power Circuit- Analysis Tool\*

R. B. Spielman and Y. Gryazin

Idaho State University Pocatello, ID USA [spierick@isu.edu](mailto:spierick@isu.edu)

***Abstract*— SCREAMER was developed to solve a wide range circuits with a focus on pulsed-power systems. SCREAMER is a highly optimized code written in Fortran 77. We will describe the mathematical foundations of SCREAMER and show how SCREAMER uses a wide range of pulsed power circuit elements. SCREAMER incorporates many physics-based models such as lossy transmission lines, dynamic loads, gas switching, water switching, oil switching, magnetic switching, and magnetically insulated transmission lines, which are important to the high- voltage, pulsed-power community. Additional circuit models or modifications to existing models can be readily implemented in SCREAMER. We show an example of SCREAMER modeling a gas switch. SCREAMER is openly available to the community without restrictions. SCREAMER runs on the Macintosh, LINUX, and Windows platforms.**

***Keywords—circuit modeling; circuit simulations***

1. INTRODUCTION

SCREAMER is a special purpose circuit code originally developed in 1985 by Kiefer & Widner1 for pulsed power applications. Recently an improved gaussian matrix solver was added by Spielman & Gryazin.2 SCREAMER uses a very fast, accurate numerical differencing scheme with one time level; is fully implicit; and is second-order accurate. It is a flexible circuit code with extensive physics models built for pulsed- power applications. SCREAMER is written in Fortran 77 (with F90 extensions) and is presently compiled under the GNU framework (gcc and gFortran). SCREAMER runs in the MacOS, LINUX, and Windows environments. Screamer takes advantage of the modern computing capabilities found on desktop computers (DRAM > 8 GB and speeds > 0.1 TFlops). We describe here the mathematical foundations of SCREAMER, and provide an example of its use.

1. SCREAMER TOPOLOGY

SCREAMER is designed with a linear circuit topology in which limited circuit elements must be arranged in a linear fashion – in a branch. (See Fig. 1.) This may seem to be very restrictive in terms of allowable circuit elements (no series capacitors or parallel inductors) but that is not the case. Screamer allows secondary branches across both series elements and parallel elements of the primary or main branch. This effectively permits arbitrary inductors in parallel and capacitors in series to the main branch. In addition, branches

\*This work was supported by Sandia National Laboratories under contract #1518167.

allow for the modeling of multi-module systems in which many modules may tie together near the final load. Branches may leave from secondary (or higher) branches thereby allowing extremely complex circuits. It is important to note that a branch may never reconnect to any other branch. A detailed description of the derivation of the finite difference solution is available upon request. Additional details can be found in the SCREAMER User Manual.



Fig. 1. A linear circuit branch showing a progression of circuit elements added together to make up a complete circuit.

1. THE SCREAMER CIRCUIT MODEL

SCREAMER was designed to very rapidly solve a limited circuit architecture in which the primary circuit element is the PI block shown in **Fig. 2**. This was deliberately done to restrict the mathematics to a first-order differential equation.

The PI block is the fundamental element of transmission line segments. In this case, the resolution of the TL is determined by the values of the PI block making up the smallest length of the TL. Individual circuit components such as resistors, inductors, and capacitors are simply PI blocks with the unneeded values in the PI block set to 0.

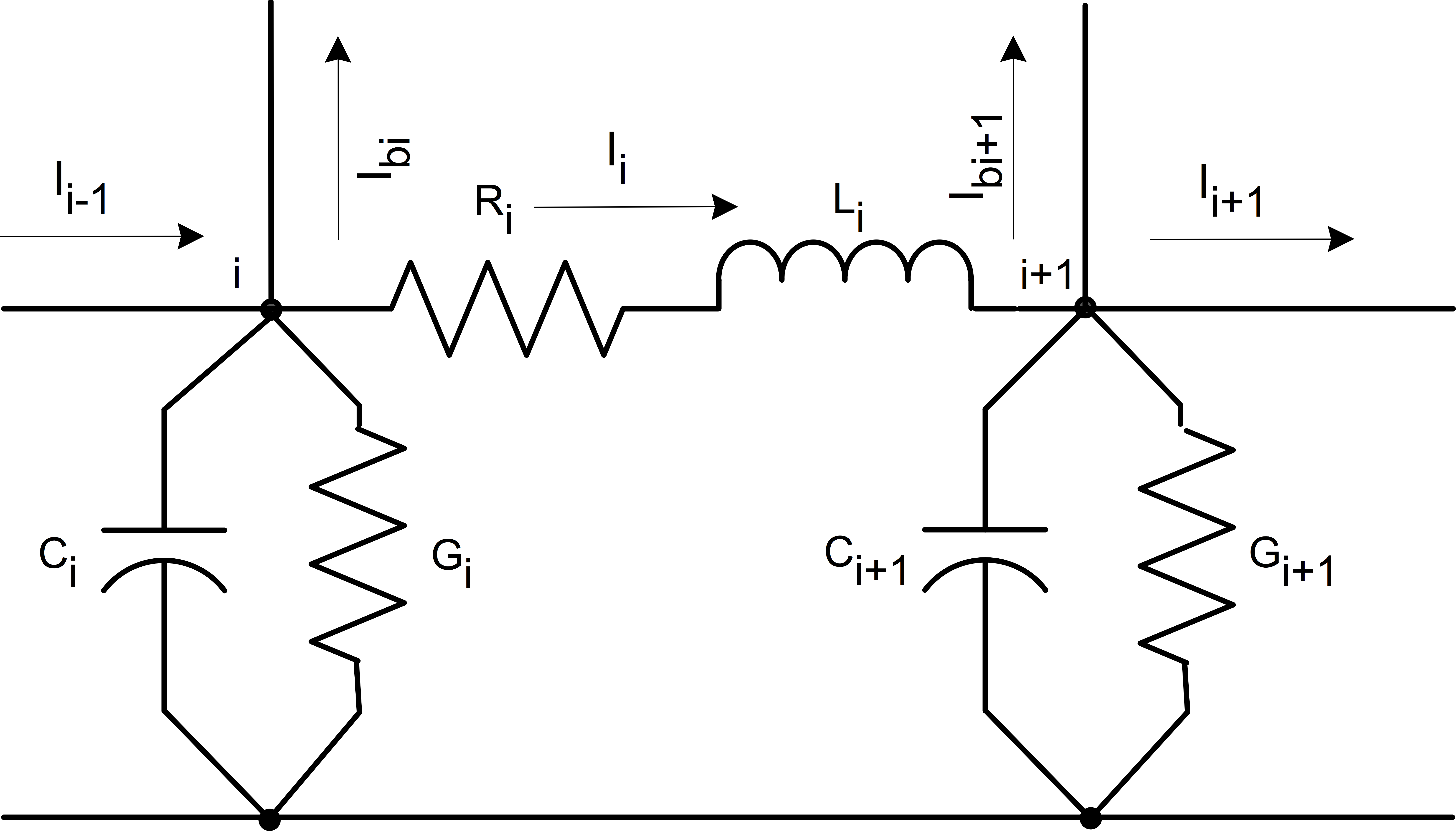


Fig. 2. A typical PI-block showing the fundamental circuit elements making up the block.

*n o n o n o*

0.5 *Ii*1  *Ii*1  0.5 *Ii*  *Ii*  0.5 *Ibi* + *Ibi* =

# SCREAMER Differencing Scheme

From first principles, we derive the fundamental equations that are used in SCREAMER. We see the layout of two arbitrary

*n*

0.5 *I*

0.5*G V n*  *V o* + *Ci V n*  *V o i i i t i i*

*n n Ci n*

SCREAMER nodes in **Fig. 2**.

We can write the current equation of a single node. The current into the *ith* node (from the *ith* – 1 node) equals the

*i*1  *Ii*  *Ibi*

 0.5*Gi* + *t Vi* =

0.5 *Io o o* + 0.5*G*  *Ci V o*

* *I*  *I*

current out of the *ith* node, plus the current out of the *ith* node into the branch *b* at the *ith* node, and plus the current that

*n o n*

*i*1 *i bi*

*o*

*i t i*

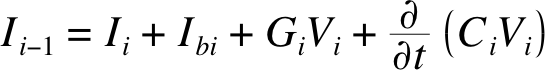
(6)

flows through the circuit elements to ground from the *ith* node. Conductance *G* (1/*R*) is the resistive path to ground and the capacitance *C* is the capacitance to ground. The currents are in Eq. 1 below.

0.5 *Vi*  *Vi*

 0.5 *Vi*+1  *Vi*+1 =

0.5*R I n*  *I o* + *Li I n* *Io i i i t i i*

*n n Li n*

(1)

We can then write the voltage equation between the *ith* and

0.5 *Vi*  *Vi*+1

 0.5*Ri* + *t Ii* =

0.5 *V o*  *V o* + 0.5*R*  *Li I o*

(7)

*ith* +1 nodes. The voltage drop between the *ith* node and the

*i*1 *i*

*i t i*

*ith* + 1 node is only the voltage drop across the *ith* resistor and *ith* inductor due to the *ith* current. These voltages are shown in Eq. 2 below.

(2)

The RHS of Eqs. 6 & 7 are constants are derived only from the circuit variables in prior time steps. The circuit values on the LHS of the equations in the parentheses are also constants. Let’s rewrite Eqs. 6 & 7.

0.5 -*I n*  *I n*  *I n*  0.5*G*  *Ci n*

We can then express both equations in terms of half time

*i*  1 *i bi*

*i t Vi* =

steps in which the values of the current and voltage from the

*o o o o*

## 0.5 *I*

*Ci*

*t*

old time step (*o*) are known and the values at the new time step (*n*) are to be calculated. The use of split time steps

*i*1  *Ii*  *Ibi* 

*n n n*

*Li*

*t*

* 0.5*Gi Vi*

(8)

increases the fundamental accuracy of the calculation. The voltage and current expressions for half time steps are below

in Eq. 3.

0.5 *Vi*

* *Vi*+1

 0.5*Ri* 

*o o*

*Ii* =

*o*

*Li*

*t*

0.5 *Vi*

 *Vi*+1 

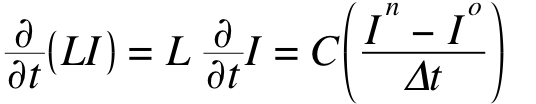
* + 0.5*Ri Ii*

(9)

(3)

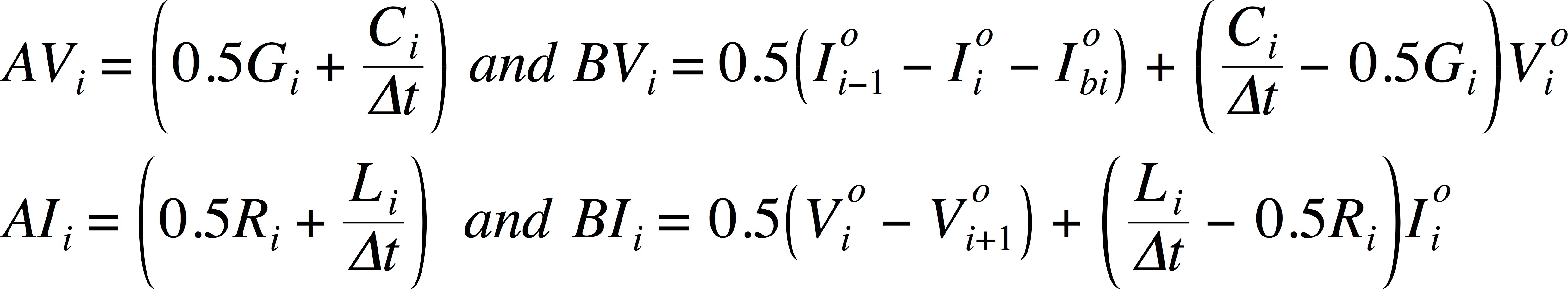
The partial derivative of (*LI*) and (*CV*) with time is simply expressed as a difference in the values of *I* and of *V* divided by the time step. This is shown in Eqs. 4 & 5. Note the use of the difference in current and voltage in the partial derivative with the assumption that *L* and *C* are not functions of time.

## (4)

 (5)

We can now express Eqs. 1 & 2 in terms of new and old variables. We see that the presence of a branch at the *ith* node only shows up in the *ith* current equation of the main branch. (This will lead directly to a sparse matrix when branches are used.) We collect the new variables on the left and the old variables on the right. The current equation is in Eq. 6 and the voltage equation is in Eq. 7.

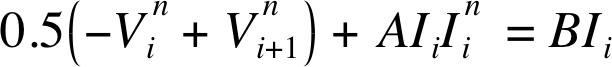
We can further simplify the equations by defining the four constants separately in Eq. 10. We use exactly these four constants in SCREAMER. The *Ai* constants depend only on circuit values. The *Bi* constants can be populated with circuit values and from the results of the prior time step.



(10)

By substituting these variables into Eqs. 8 & 9 we see that we have a simple set of 2•*N* equations and 2•*N* unknowns. Where *N* is the total number of nodes and *i* = 1, *N*. Thus, we finally have the simple representation of the circuit equations in Eqs. 11 & 12. Each of these equations is recalculated moving forward time step by time step and we leave out the notation indicating a specific time step for clarity.

## (11)

 (12)

We can take the coupled set of Eqs. 11 & 12 and rewrite them in terms of alternating current and voltage with increasing node index *i* in Eqs. 13 & 14. Finally, for simplification of the notation, we will drop the superscript *n* (which was indicative of the “new” value of voltage, which is now assumed to be calculated), add a superscript *k* indicative of the branch number, where *k* = 1, *nb* and *nb* is the total number of branches.

(13)



(14)

For only the case of *i* > 1 and *i* < *nk* (*i*), we multiply both Eqs. 13 & 14 by 2 and Eq. 14 by -1. While these choices seem arbitrary at this point, we have determined that this reduces number of calculations in the solver. The constant *nb* remains the total number of branches in the problem. Eq. 13 is the current equation for the *kth* branch and in the *ith* node of that *kth* branch, current leaves the *kth* branch into the first node a new *lth* branch. In SCREAMER, the branch index inequality *k* < *l* must hold.

(15)

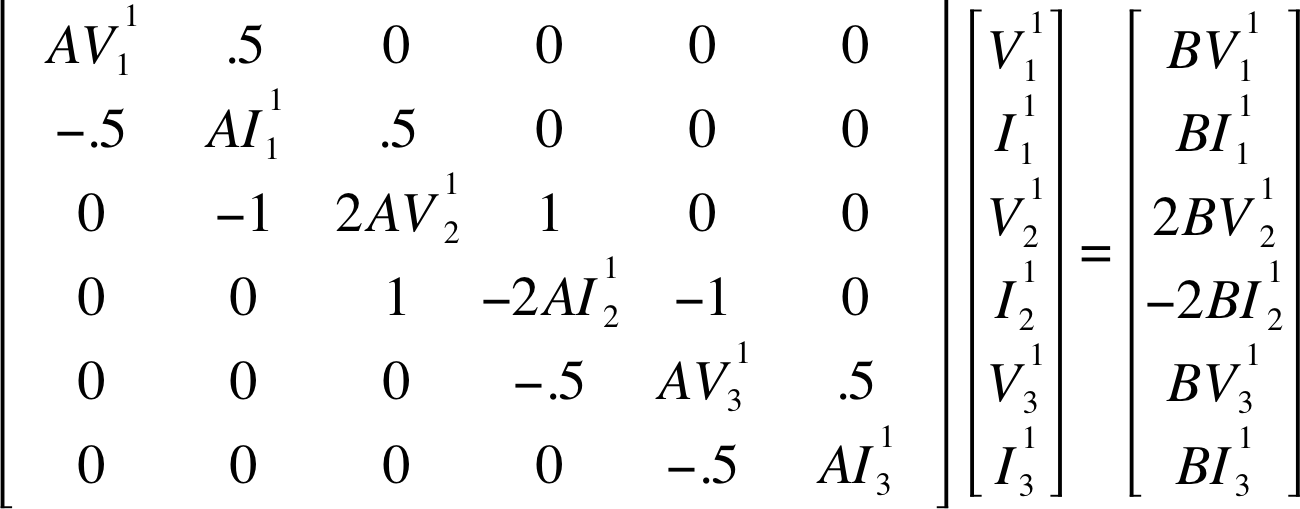


(16)

Eqs. 15 & 16 are the equations we will use to populate the full problem matrix and vector in the section below for *i* > 1 and *i* < *nk*(*i*).

# SCREAMER Matrix Population

Assume a simple example with no branches. We can express this simple three-node problem as a vector with 6 elements in the matrix formation shown in Eqs. 13 – 16. The *AVi* and *AIi* constants are used to populate the ***A*** matrix and the *BVi* and *BIi* constants are used to populate the ***b*** vector. The first row has no current from the prior node and the last row has no voltage in the following node. This is true even with branches. Be careful with the first and last nodes of all branches, they are based on Eqs. 13 & 14.



(17)

We can express this matrix and the two vectors in simple matrix notation. Where the ***x*** vector is the solution vector.

***A*** • ***x*** = ***b*** (18)

# How SCREAMER Actually Builds the **A** matrix and

***b*** *vector*

SCREAMER actually generates a slightly different node structure and different node counts than the simple discussion of circuit elements in Sect. III. B might suggest. Building a circuit from SCREAMER circuit blocks requires open nodes for connectivity to the next block. In practice, this means that we need *every* circuit block to end in a “dangling” resistor/inductor series block (RLseries). With this circuit configuration one can connect any type of circuit block in series. (Consider the trivial example if you tried to connect two RCgrounds one after the other.)

SCREAMER, as written, creates an extra *phantom* block and nodes that separate actual circuit elements. For example, a simple resistor & capacitor to ground block (RCground) mathematically requires only a single *ith* node and the *G* & *C* elements to ground but SCREAMER adds a phantom RLseries element following the *ith* node. In this case, the *R* & *L* in the *ith* phantom RLseries element are set to zero.

Similarly, a resistor and inductor in series block (RLseries) would naturally be placed between the *ith* and *ith* + 1 nodes. Phantom RCground elements are placed at the *ith* and *ith* + 1 nodes. A phantom RLseries element follows the *ith* + 1 node just as is done for the RCground block. In this case, the *G*’s (1/*R*) & *C*’s in the RCgrounds are set to zero and the *R* & *L* in the second, phantom, RLseries are set to zero. Only the first RLseries element has non-zero values.

Fig. 1 shows the nodes and elements of a simple example problem (without branches) that would be solved in SCREAMER with the phantom nodes.

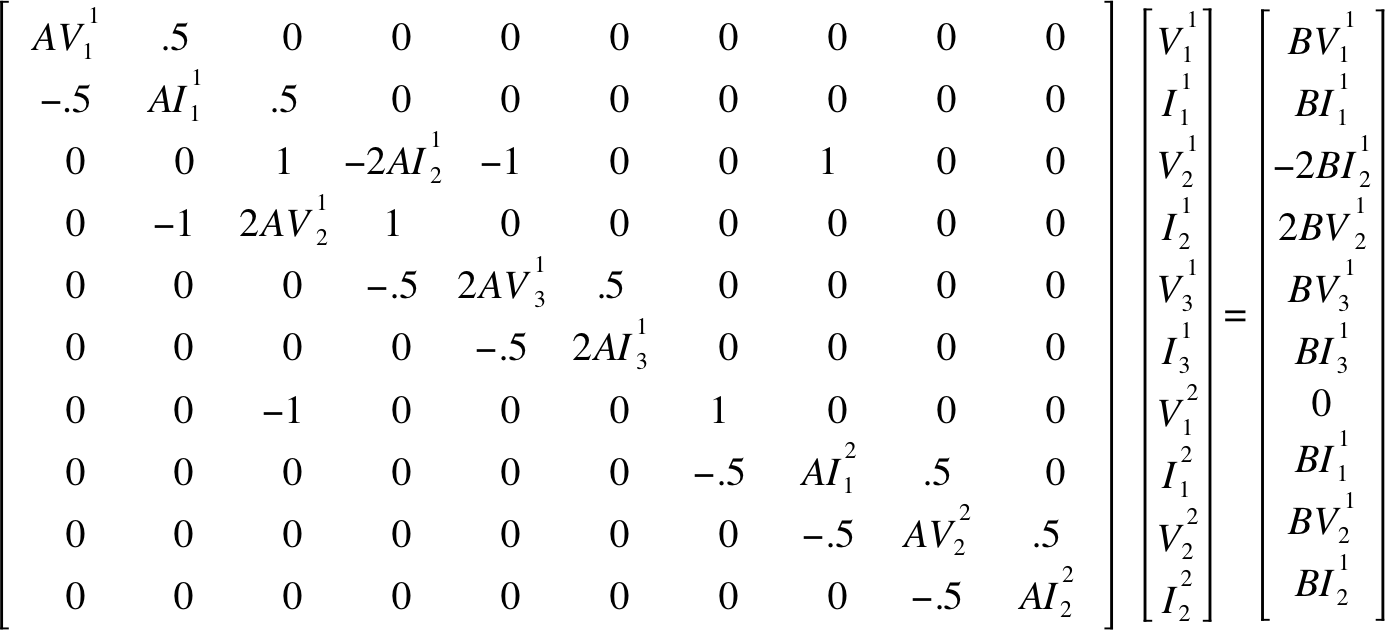
For example, with an RCground at node 1, there is no *actual* series element between nodes 1 and 2 to connect to the next block, so a *phantom* RLSeries element must be included, in which the parameters *L1* and *R1* are both set to 0. Screamer puts a connecting “wire” in the circuit. This means that *V(i)* = *V(i+1)* for all time steps or *V(i+1) - V(i)* = *0* for all time steps. Think of an RCground block as having a phantom RLseries element with the *Ri* & *Li* values zero.

There is an RLseries block between nodes 2 & 3 with two phantom RCground elements with their contents, *C* & *G*, set to zero and a dangling phantom RLseries block with all of its parameters zero following node 3. If one did not place a node immediately after the RLseries block after node 2 then connection to the next node would be difficult. The first RCground is at the 2*nd* node and the second RCground is at the 3*rd* node. Thus, an RLseries block actually populates ALL of the nodes that are available in that block and ends the block with a dangling phantom RLseries element.

In the end, all this does is greatly simplify the way that arbitrary circuit elements can be tied together without worrying about internal code quirks. This has the very unfortunate effect of increasing the number of nodes that a full matrix solver will have to handle AND, more unfortunately, forcing many of the *A* and *B* constants to have a value of zero.

# SCREAMER Matrix Solver

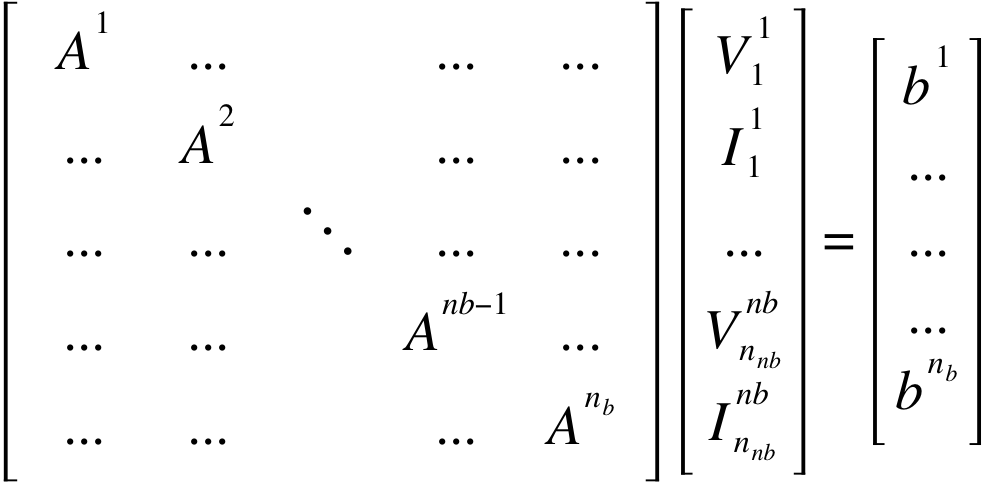
The matrix solution to the sparse tri-diagonal matrix described in Section C above has significant mathematical problems. Many of the diagonal elements will always have a value of zero due to the large number of phantom nodes. See Eqs. 13 – 16 and Eq. 17. A matrix having diagonal values that are zero cannot be solved with the usual methods due to divide by zero problems in the solver algorithms. Branches show up as off diagonal elements that limit sparseness. (See the matrix in Eq. 19 below.) We address this problem by converting he tri-diagonal matrix to a penta-diagonal matrix. Branches from branches only add to the sparseness problem.



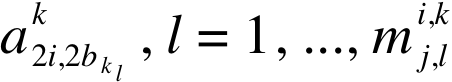
(19)

Our solver exhibits the efficiency of the standard sparse solvers, i.e. the number of multiplications in the proposed solver is *O(N)*, where *N* is the number of non-zero entries in the sparse matrix. It also has a very transparent programming implementation due to the efficient storage format and a simple, repeating sparse-matrix structure.

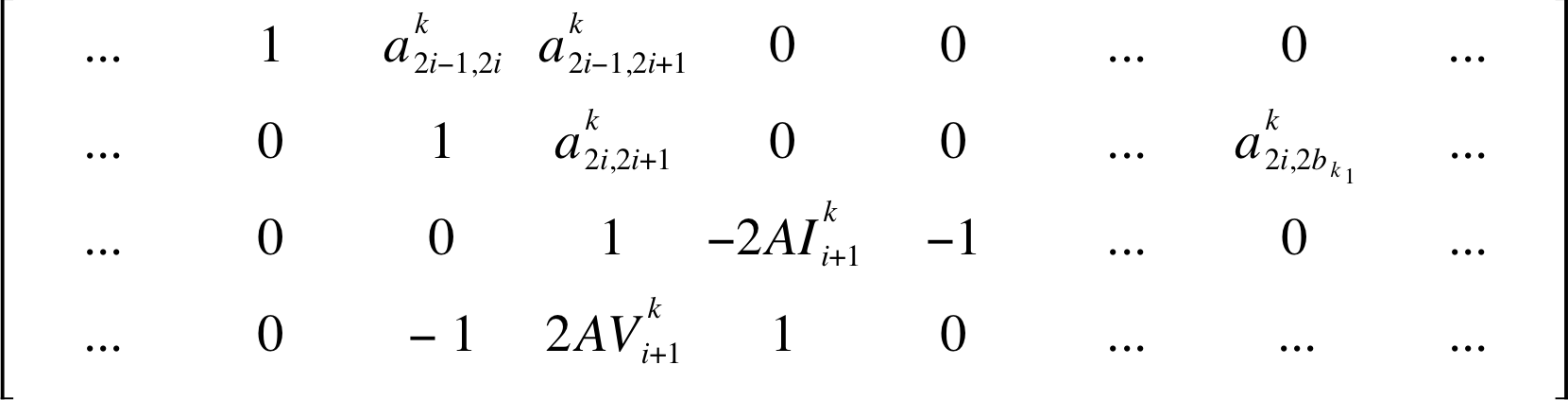
SCREAMER’s topology and restrictive circuit elements result in a linear system ***Ax*** = ***b*** that is shown in Eq. 20 in the following block form.

 (20)

In this system an *Ak* block corresponds to the *kth* branch. *Ak*, in turn, also has two rows of block structure corresponding to the nodes in the *kth* branch. In the system above, we skipped the indication of the branch connections in the off diagonal elements but we will consider them later. We also use the notation  for the first row in the *lth* block.

Then the *ith* and (*i* + 1)*st* 2-rows block of *Ak* can be presented in penta-diagonal form in Eq. 21. In Eq. 21 the *ith* block has already been transformed into row-reduced echelon form and we consider the details of the transformation of the (*i* + 1)*st* block. For convenience, in this matrix we indicate just one branch connection term where

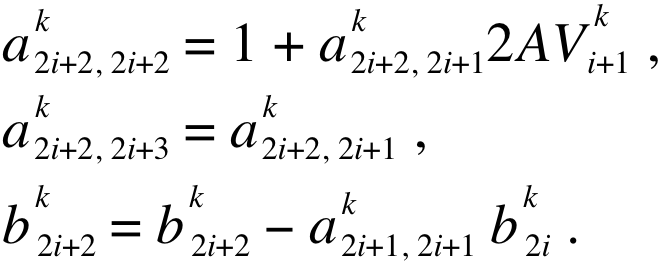
is the number of branches *kj* connected to the branch *kl* with *kj* < *k* < *kl*. It also includes the connection of the *lth* branch to the *kth* branch at the *vth* block, if *v* < *i*. It can be seen that the transformation of the (*i* + 1)*st* block of the *kth* branch to the reduced echelon form requires the elimination of two entries in the positions (2*i* + 2, 2*i*) and (2*i* + 2, 2*i* + 1) in the matrix block corresponding to the *kth* branch. The first step in the process requires only additions.



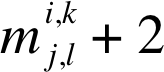
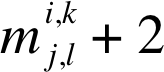
(21)

## (22)

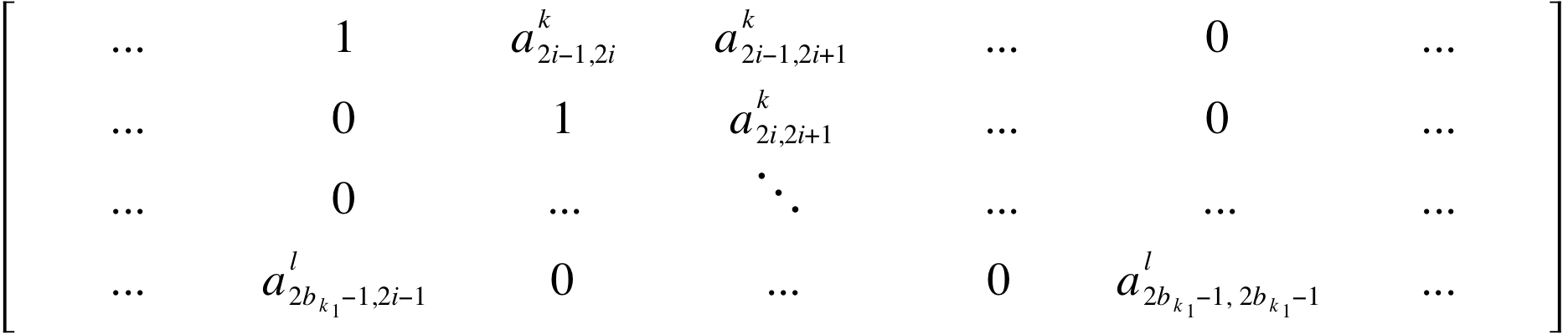
Note that the second term in the second equation of Eq. 22 is almost always zero.

The second elimination step, shown in Eq. 23, requires just two multiplications.

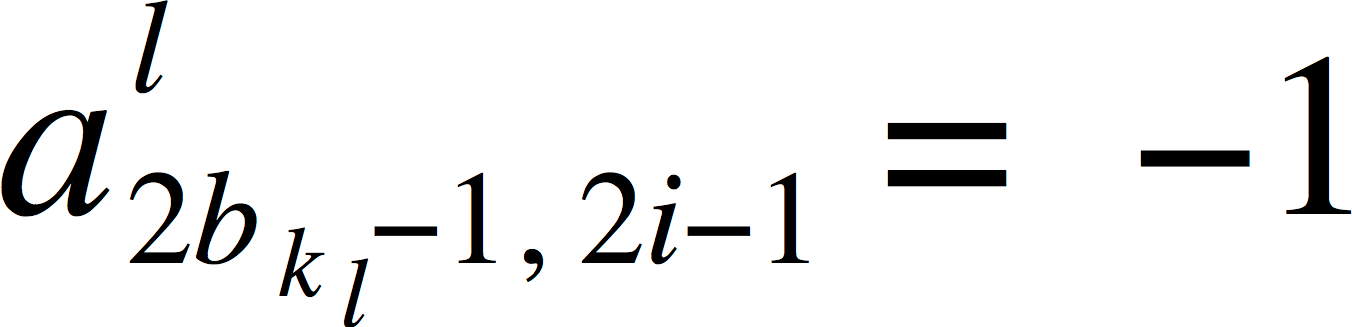
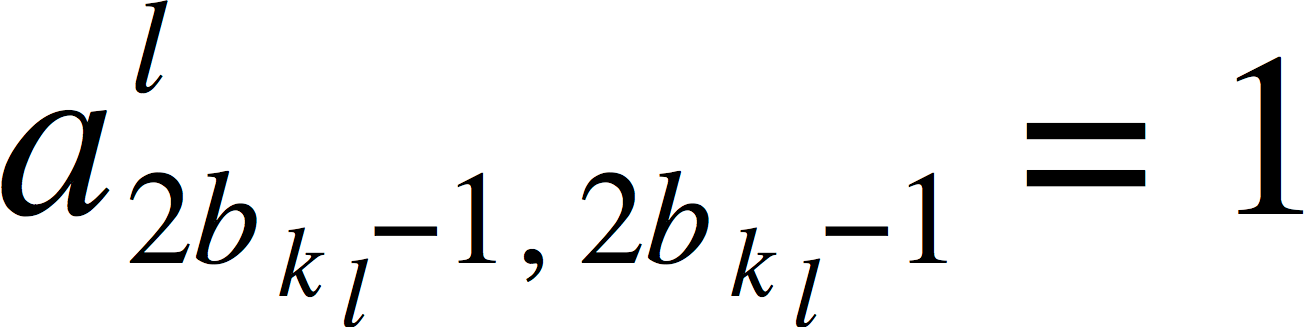
(23)

The scaling of the (2*i* + 2)*th* row results in additional  multiplications. This shows that the forward elimination step for the (*i* + 1)*st* block of the *kth* branch requires  multiplication operations.

Next, we consider the forward Gaussian elimination step at the first node of a branch *kl*. We assume that the branch *kl* is connected to the *ith* block of the branch *k* (*k* < *kl*). Also we consider that all rows above the first row in the block corresponding to the *lth* branch have already been transformed into row-reduced echelon form. Then we can present the relevant rows of the matrix in the form in Eq. 24.



(24)

In Eq. 24,  and  . If we denote the difference between these column positions by  , then we need ξ*l* iterations of a 2-step process to sequentially move the first non-zero entry to the right and eventually eliminate it. The first step in the process is:

*al* = *al ak* ,

! 80 nF version of the GA 35426M Capacitor 40 nF,

2*bk l*1, 2 *j*

2*bk l*1, 2 *j*1

2 *j*1, 2 *j*

! 15 nH, start with a 0.13 Ω ESR. L total - 121.5 nH

*al* = *al*  *al ak* , !

2*bk l*1, 2 *j*+1

2*bk l*1, 2 *j*+1

2*bk l*1, 2 *j*1

2 *j*1, 2 *j*+1

*bl* = *bl*  *al bk* .

! Use the Martin Switch model for the Kinetech Switch

2*bk l*1

2*bk l*1

2*bk l*1, 2 *j*1

2 *j*1

(25)

!

! Charge voltage - 100 kV

We also can note that in Eq. 25, the first term in the right hand side of the second equation is almost always zero. The second step can be written as

*al* = *al*  *al ak* ,

! BRANCH

!

! Capacitor capacitance and charge voltage

2*bk l*1, 2 *j*+1

2*bk l*1, 2 *j*+1

2*bk l*1, 2 *j*

2 *j*, 2 *j*+1 !

*al* = *al*  *al ak* , ** = 1, ..., *mi*,*k* ,

2*bk l*1, 2*bk *

2*bk l*1, 2*bk *

2*bk l*1, 2 *j*

2 *j*, 2*bk *

*j*,**

RCGround 1e+12 40e-9

*bl* = *bl*  *al k*

Initial VC1 200e3

2*bk l*1

*k l*

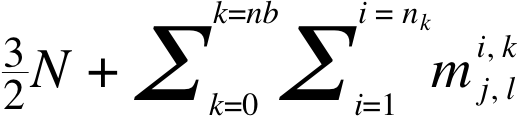
2*bk l*1

2*b* 1, 2 *j b* 2 *j* , *j* = 1, ..., *bk*  1 .

*l*

(26)

UFO VC1

Given *N* equations in the matrix, the number of multiplications required on the back substitution step is estimated by . We can see that if the number of branches remains fixed, both the forward and back substitution steps are approximately *O(N)*. So, the resulting algorithm presents a very efficient, direct linear solver.

The constructed direct linear solver has no limitation on the number of embedded branch levels. Essentially, there is only one restriction on the branch connections: If a branch *kl* connected to a branch *km* then the entry in the unknown vector corresponding to the first node of the branch *kl* must come after all entries in the unknown vector corresponding to nodes in the branch *km*.

1. USING SCREAMER TO MODEL GAS SWITCHES

One of the major advantages of SCREAMER is the extensive collection of validated physics models built into the code. In this Section we will model a simple capacitive discharge through a triggered gas switch into a fixed resistive load. SCREAMER contains a variable resistor model written by T. H. Martin.3 This model implements a Braginski model for the expansion of the arc channel in a switch. The model is applicable in various gases, water, and oil. We show a run deck in Sect. IV.A that describes the problem in SCREAMER form.

# SCREAMER Run Deck for a Switch Calculation

SNL Brick Model

!

! Rick Spielman 2016-05-13

!

Time-step 1e-12 Resolution-time 1e-9 End-time 1.5e-6 Number-prints 5 Execute-cycles all Grids no

Echo-setup no Max-points 20001

!

!Start circuit definition

!

! Start with a capacitor and switch

$V\_brick

! Cap inductance, case & parasitic inductance, and ESR

!

RLSeries 0.24 50e-9

!

! Martin Model, L – measured 0.375” gap and 242 psia air

!

RLSeries 10e9 35e-9 Var R2 Switch

!dielectric switchtime gap pressure nswitch nchannels AIR 0.0 0.009525 16.46 1 1

UFO R2

$Sw\_res

!

! Output TL header inducatance

!

RLSeries 0.00 36.5e-9

!

! Load resistance = Rmatched

!

RCGround 2.0 0.0 UFO IR1

$I\_brick UFO QR1

$Q\_brick UFO VR1

$V\_out UFO PR1

$P\_brick UFO ER1

$E\_brick

!

! End circuit

# Simulation Results Compared to Data

The total inductance of the simulation was adjusted slightly to fit the measured time to peak current. (The inductance of the circuit is only known to ~ 10%.) We also adjusted the total resistance of the circuit by a few percent to match the measured peak current. Fig. 3 shows the brick current in the SCREAMER simulation compared to the data. The simulation assumes a total brick inductance of 121.5 nH, a capacitor ESR of 0.24 Ω, and a load resistance of 2 Ω. The T. H. Martin model uses an air dielectric at 16.5 bar, a switch gap of 0.95 cm, and single

arc channel. The initial switch resistance was chosen to be 10 GΩ. (The model is insensitive to the choice of the initial resistance.) The simulation runs with a 2-ps time step to accurately model the changing resistance of the switch. (Recent analysis shows that time steps of ~ 10-3 of the problem duration provide adequate accuracy for the T. H. Martin variable resistance.) The capacitor delivered a peak current of 48 kA in 100 ns 0-100% at a charge voltage of +/-100 kV. The circuit delivered a total of ~ 10 mC to the load. An analysis of the calculation shows that the SCREAMER circuit model matches the data nearly perfectly for a full cycle of the current. The simulation only deviates in the start of the second cycle. Note, these data were taken using a liquid (salt) resistor and its resistance may vary as energy is deposited into the liquid. The electrical characteristics of a gas-switch/brick cannot be modeled accurately using a fixed value of resistance.

hohlraum loads, magnetic-flashover inhibition model, and magnetic switching.

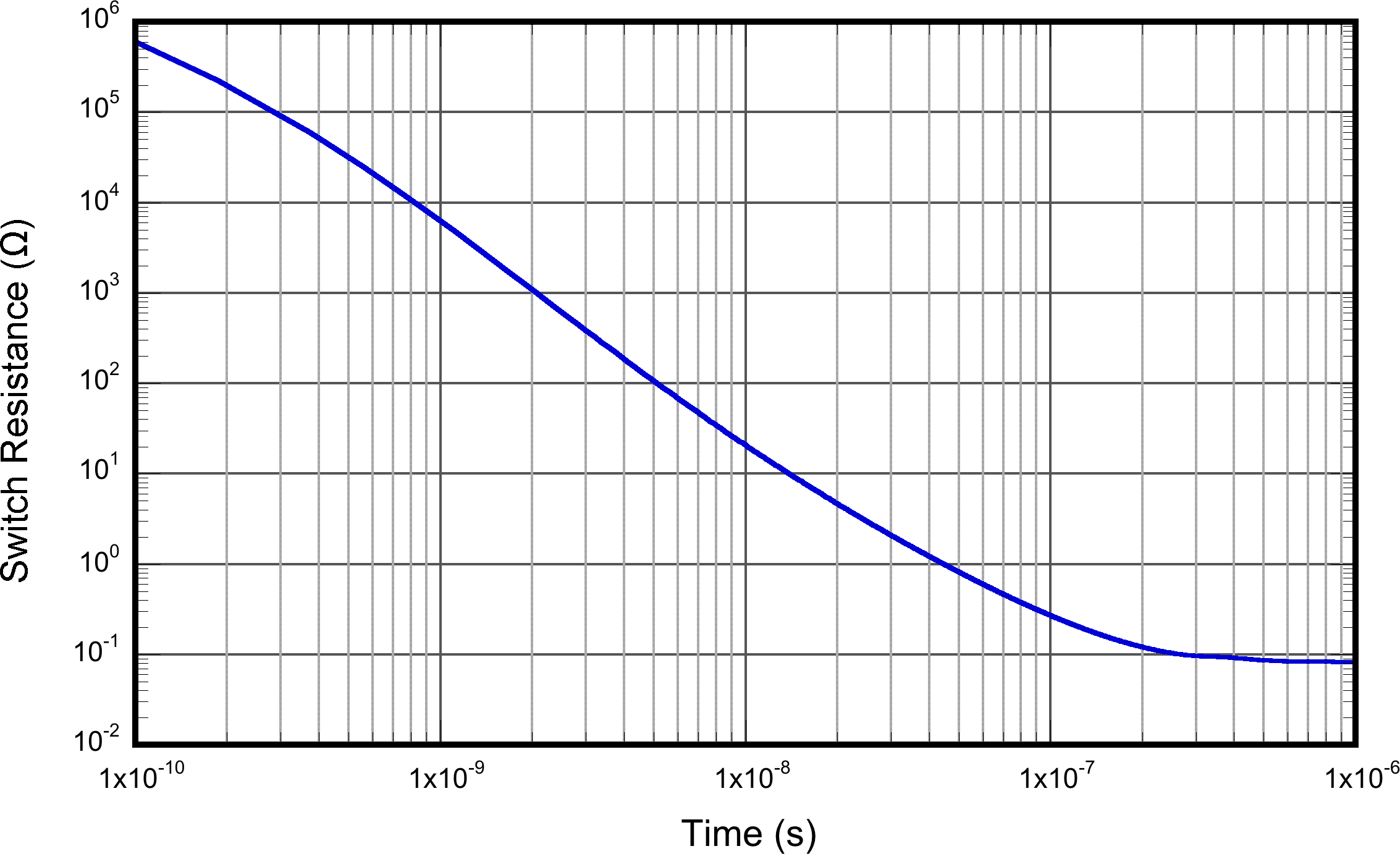
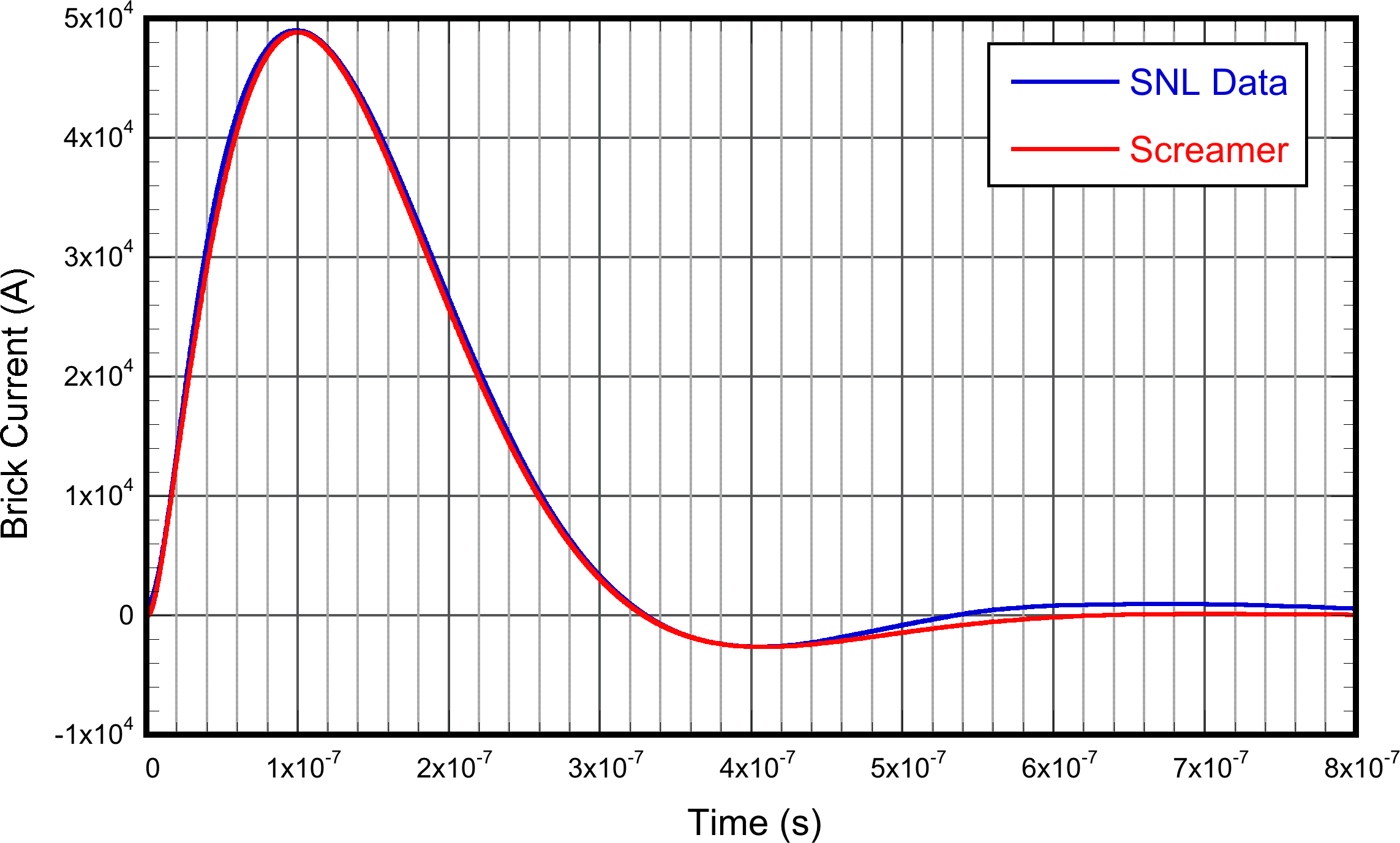


Fig. 4. A plot of the switch resistance calculated by SCREAMER as a function of time.

Fig. 3. We compare the current calculated by SCREAMER (red line) and the measured current (blue line).

It is instructive to plot the calculated resistance of the high- voltage gas switch as a function of time. (See Fig. 4.) Early in time, the resistance of the switch is > 500 kΩ. This resistance rapidly falls as the driving current increases (depositing energy in the arc channel) and the arc channel expands. The switch resistance falls monotonically in time as the pulse progresses. Even at peak current (~ 100 ns) the resistance of the switch is still ~ 0.3 Ω. This is a large fraction of the 2-Ω matched impedance of the circuit. After peak current, the arc channel continues to expand and the resistance of the arc channel continues to fall.

We conclude that SCREAMER is capable of accurately modeling the resistive behavior of high-voltage, high-current gas switches in the parameter ranges of interest. We have predictive capability in the design of improved switches.

1. EMBEDDED PHYSICS MODELS IN SCREAMER

One of the unique features of SCREAMER is the presence of a large suite of validated physics models. We have implemented the following features that are very useful for users: z-pinch loads, gas-puff loads, spherical implosions, multi-shell loads, decaying resistors, exponential resistors, variable inductors, DPF loads, ion-diode loads, lossy transmission lines, magnetically insulated transmission lines, Z-flow electron-loss model, electron-beam diodes, dynamic-

1. CONCLUSION

SCREAMER can address most circuits of interest to pulsed- power systems with the application of extensive branches in branches. The solver mathematics has been implemented in a clear and transparent fashion. The Gaussian, sparse-matrix solver is heavily documented. This solver allows for a simple diagnostic analysis of the problem matrix and a direct comparison of the circuit with the problem mathematics.

SCREAMER has a large suite of embedded physics models that are of interest to the pulsed-power community. The presence of these physics models makes SCREAMER unique in the circuit-modeling world.

SCREAMER is available for use without restriction. Copies of binaries and the source code are available upon request. We request that all copies of SCREAMER be obtained directly from ISU so that we can maintain a list of users for informational purposes.

SCREAMER is fully operational on Macintosh, LINUX, UNIX, and Windows platforms. SCREAMER binaries and source code are available upon request.

ACKNOWLEDGMENT

We thank M. Wisher of Sandia National Laboratories for providing us the capacitor-switch data.

REFERENCES

1. M. L. Kiefer & M. M. Widner, “SCREAMER – A Single-Line Pulsed- Power Design Tool”, in Proc. of the 5th IEEE Pulsed Power Conference, Arlington, VA, June 10-12, 1985, p. 685.
2. R. B. Spielman & Y. Gryazin, “SCREAMER V4.0 – A Powerful Circuit Analysis Code,” Proc. of the 20th IEEE Pulsed Power Conf., Austin, TX (2015), p. 637.
3. T. H. Martin, et al., in Proc. of the 9th IEEE Pulsed Power Conf., Albuquerque, NM (1993), p. 463.