SANDIA REPORT
SAND2010-1669
Unlimited Release
Printed May 2011

SAE2.py: A Python Script to Automate Parameter Studies using SCREAMER with Application to Magnetic Switching on Z

Franklin Orndorff-Plunkett

Sandia Summer Intern, 1653 Electromagnetic Effects

500 Memorial Drive: Massachusetts Institute of Technology, Physics Department

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

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2010-1669
Unlimited Release
Printed May 2011

SAE2.py: A Python Script to Automate Parameter Studies using SCREAMER with Application to Magnetic Switching on Z

Franklin Orndorff-Plunkett
[1]Sandia Summer Intern, Department 1653
[2] Permanent Address: Massachusetts Institute of Technology
500 Memorial Dr. Cambridge, M.A.

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-MS1653

Abstract

The SCREAMER simulation code is widely used at Sandia National Laboratories for designing and simulating pulsed power accelerator experiments on super power accelerators. A preliminary parameter study of Z with a magnetic switching retrofit illustrates the utility of the automating script for optimizing pulsed power designs.

ACKNOWLEDGEMENTS

I appreciate the advice and counsel of my mentor Pace VanDevender and of Ken Struve, who helped me with the intricacies of SCREAMER, and appreciate the managerial leadership of my managers Mark Kiefer in 2008 and Michele Caldwell in 2009 and of Larry Schneider, who provided encouragement and oversight of the Summer Intern Program for the Pulsed Power Science Center.

CONTENTS

1. Introduction……7
1.1. General Scripting Scheme……..7
1.2. Development of Magnetic Core Switching Technology……8

2. ScreamerAE2.py Guide…… 10
2.1. User Input……..10
2.1.1. Template Model File……10
2.1.2. Settings File……...11
2.2 Running SAE2.py……..14

3. Results of a Preliminary Optimum Search with SAE2……16
3.1. Current Performance of Z-Refurbishment……16
3.2. Optimization of ZR with Magnetic Switching Retrofit……17
3.2.1. Circuit Model and Consolidation into a Mega-Module……17
3.2.2 Optimization: A Safe and Better Performing Retrofit……..18

4. Conclusions……27
5. Works Consulted……28

Appendix A: Scripting Conventions……..29
Appendix B: Sample Template File……...30
Appendix C: Sample Settings File……43
Appendix D: SAE2.py Source Code……..47

FIGURES

Figures 1.1-1.6: Highest Yield Safe Double Switch Configuration Data…….....19-21
Figure 1.17: The output currents just after the second switch position for the single and double switch configurations…….....…….....23
Figure 1.8-1.13: Highest Yield Safe Single Switch Configuration…….....24-26

TABLES

Table 1: Effect of changing the Marx charge voltage……..……22

NOMENCLATURE

ASCII	American Standard Code for Information Exchange
COMET	A terawatt magnetic switching experiment for possible use of magnetic switching on PBFA II
DP	Dependent Variable
DV	Intermediate Dependent Variable that is used for calculating another quantity but is not itself passed into the SCREAMER run
FOM	Figure of Merit
IC	Input Constraint
ICF	Inertial Confinement Fusion
IF	Input related function
IFE	Inertial Fusion Energy
MSW(i)	(The i-th) Magnetic Switch
OC	Output Constraint
OF	Output related function
PBFA II	Particle Beam Fusion Accelerator II
PFL(i)	(The i-th) Pulse Forming Line
SAE2	ScreamerAE2.py
SCREAMER	A pulsed power design tool developed at Sandia National Laboratories
Z	The Z Machine at Sandia National Laboratories
ZR	The project for refurbishing Z and is often used to refer to Z in its refurbished state

1. INTRODUCTION

SCREAMER[1] is a circuit based code commonly used in pulsed-power design and requires numerous iterations to find optimal configurations. System optimization using simulations like SCREAMER is by nature inefficient and incomplete when done manually. This is especially the case when the system has many interactive elements whose emergent effects may be unforeseeable and complicated. For increased completeness, efficiency and robustness, investigators should probe a suitably confined parameter space using deterministic, genetic, cultural, ant-colony algorithms or other computational intelligence methods. I have developed SAE2--a user-friendly, deterministic script that automates the search for optima of pulsed-power designs with SCREAMER.

This manual demonstrates how to make input decks for SAE2 and optimize any pulsed-power design that can be modeled using SCREAMER. Application of SAE2 to magnetic switching on model of a potential Z refurbishment illustrates the power of SAE2. With respect to the manual optimization, the automated optimization resulted in 5% greater peak current (10% greater energy) and a 25% increase in safety factor for the most highly stressed element.

1.1. General Scripting Scheme

The procedure to automate SCREAMER requires the following tasks:

1. Define upper and lower bounds for each variable to define the parameter space to be probed.
2. Partition parameter ranges.
3. Establish a grid of points representing the configurations to test based on the partitioned parameter ranges.
4. For each point in the grid:
    - Represent the point as an input file.
    - Run the simulation with the input file.
    - Read the simulation output.
    - Evaluate constraints and execute any post processing necessary.

[1] Mark L. Kiefer, Kelley L. Shaw, Ken W. Struve, Mel M. Widner, "SCREAMER, A Pulsed Power Design Tool, User's Guide for Version 2.2" Sandia National Laboratories (updated 1999)

- Write in a log file the desired outputs and figures of merit.

1.2. Development of Magnetic Core Switching Technology

In 1951, W.S. Melville[2] spawned the idea of utilizing nonlinear, ferromagnetic inductors for pulse-compressing, self-closing switches. In his day, the materials engineers had not yet solved the issues of response time, manufacturing cost and short lifetimes of ferromagnetic cores. Therefore, approaches using spark-gap dielectric switches overshadowed the magnetic core approach until the early 1980s when the COMET I[3] project began. This project served as an unprecedented study of the applicability of the magnetic switching approach for a potential application in the Sandia Inertial Confinement Fusion (ICF) and future Inertial Fusion Energy (IFE) programs. A key development that made this project possible, and indeed that provided motivation, was the amorphous metallic-glass material, Metglas®. After a successful small-scale experiment[4] validated the design assumptions, Sandia bought an early production run of this material for use on the COMET I project as a full scale prototype of a PBFA II module. The success of the COMET I project—including validation of the SCEPTRE magnetic core code and characterization of Metglas® 2605SC material—led to COMET II[5] as an option for PBFAII. Magnetic switch hardware used in the COMET experiments failed at ~1000 shots because the manufacturing process had not yet matured and defects[6] on the Metglas® punctured the mylar wrapping, which eventually failed. New methods of manufacturing Metglas® result in fewer defects and a more robust switching system has already been deployed in commercial accelerators at comparable electrical stresses on the core. As a result of these improvements in Metglas® technology, a first principles analysis[7] showing that the transients in a wound core are

[2] W. S. Melville, IEEE Proc. **98** p.186-207 (1951)
[3] E. L. Neau, Proceedings of the 4th IEEE Pulsed Power Conference, 6-8 June 1983, Albuquerque, NM, USA; p.246-250 (1983)
[4] M. Stockton, E., L. Neau, and J. P. VanDevender, J. Appl. Phys. **53** (3) p2765-7 (1981)
[5] E. L. Neau, T. L. Woolston, and K. J. Penn, Proceedings of IEEE Power Modulator Symp; 1984; p.292-294 (1984)
[6] H. C. Harjes, K. J. Penn, G. A. Mann and E. L. Neau, Proc. 6th IEEE Pulsed Power Conference, Arlington, Virginia 1987, p.540-543 (1987)
[7] W. A. Johnson and L. K. Warne, Sandia National Laboratories SAND Report 90-1061 (1990); W. A. Johnson and L. K. Warne, Lasers and Particle Beams **10** (2) p. 375-385 (1992)

benign, and lower prices of materials, it is now reasonable to re-examine magnetic switching as an approach to pulse-compression for pulsed power applications.

2. SAE2 Guide

2.1. User Input

2.1.1. Template Model File

The template file must be an ASCII encoded .dat file that accurately reflects the device to be modeled. If the device is a radial orientation of identical lines, such as is the case with many ICF pulsed-power approaches, it must be represented as a single transmission line or one comprised of a set of branches (see SCREAMER manual for full documentation of use). In place of the original values for input variables, SAE2 uses substitution syntax to properly order values when making input files. For example, the following syntax declares a magnetic switch in the SCREAMER input file:

```
VAR L2 MSW
!PF Ri        Ro       W    H1 Hsat Hrev Bsat
0.5 .74188 0.8382 0.44 0.1 0.18 0.05 15.36
```

If Ri and W are the variables to be varied, then the user must type the following as substitution:

```
VAR L2 MSW
!PF   Ri      Ro        W       H1 Hsat Hrev Bsat
0.5 %m.nt 0.8382 %m.nt 0.1 0.18 0.05 15.36
```

where ‘%m.nt’s serve as placeholders for the numerical values that will be substituted during SAE2’s runtime. For this syntax, ‘m’ is the minimum number of digits for the value to be substituted and ‘n’ the number of digits to display after the decimal point. The ‘t’ is the value’s type and has several options—either ‘f’ for a float number, ‘i’ or ‘d’ for an integer number, ‘s’ for a string literal, ‘e’ for an exponential number (base 10) and g for the shorter of either the exponential or float representations. For SAE2 to properly order the variables, the user must type a parentheses enclosed, comma-separated list of the variable names—in the order they appear in the SCREAMER input file—at the end of the template file with a ‘%’ character just preceding it. For example:

‘%(varname_1,varname_2,...,varname_n)’

A complete example of the template file is at the end of this document. The following is a sample of the .dat template file:

```
Pulsed Power Design
! *Comment-suite*
TIME-STEP 2.E-12
RESOLUTION-TIME 1.E-10
END-TIME 2.2E-6
NUMBER-PRINTS 1
EXECUTE-CYCLES ALL
ECHO NO
MAX-POINTS 3000
BRANCH
RCG 1E12 390.0E-9
INITIAL VC1 %5.3e
[...Rest of SCREAMER input file with substitution syntax...]
%(VoltageInit,...)
```

2.1.2. Settings File

This file must have the same name as the .dat template file but an .asc extension (ASCII encoded) and must define all variable ranges, functions, constraints, time relation definitions, time-related calculations, working directories, environmental variables, and figure of merit functions. All comment lines must contain one or more '&' characters and any lines containing information the user wishes to exclude from the run at hand must have at their ends 'do not use'. For the sake of readability, the user can use more whitespace between words. Writing variable types in groups makes the file more readable (ranges, environmental variables etc.).

Three user-defined environmental variables are necessary for creating and working in the desired directories for the SCREAMER code and logging for Windows and Linux operating systems. If necessary, we can configure a version for the Macintosh OS:

- SCREAMERexe = "ScreamerXPU.exe" – name of SCREAMER executable file (must be in quotations)
- SimDir "Simulation_Directory"– directory where the SCREAMER executable is located (must be in quotations)
- CatFol = "Subfolder_Name"– directory where the log file will be written (must be a subfolder of SimDir and in quotations)

The user initializes the variables in two ways, either partition a range into values or specify only discrete values:

```
variable : [lower_bound, upper_bound] i
variable : (value0, value1, value2, … , valuen)
```

The first choice of syntax partitions the difference between upper and lower bounds into 'i' testable points for that variable. The second choice uses the values specified in the array `(value0, value1, value2, … , valuen)` as the testable points for that variable.

The expressions to be evaluated are entered as simple equations with two-character flags to indicate if the function is a dependent variable ('DP'), intermediate dependent variable ('DV' not to be substituted but used for calculating another quantity), output related ('OF'), or input related ('IF'). Variable enclosure and operation-order conventions are the same as those used in Matlab with the additional conventions 'var min' or 'var max' to ask a minimum or maximum of the variable 'var'. Those conventions along with all usable standard mathematical functions are outlined in the Appendix A. Some examples follow:

Tau2 = (1.4708-(W1+W2)-.2)/(300000000/sqrt(80)) DP

MSV = pi*((.8382)^2-(Ri1)^2)+pi*((.8382)^2-(Ri2)^2) IF

Vsafe = 1.08E5*(.8382*(1-exp(-sqrt(80)*(ZPFL2)*9/60)))*|Teff|^(-.33) OF

The user can declare constants by simply typing:

C1 = value

where the value is either an integer, floating point, or exponential number.

The user can enter constraints in plain English sentences. Flags after the end of each sentence indicate whether the value to be constrained is related to SCREAMER output ('OC') or to input ('IC').

- Vsafe must be greater than VPFL2 max. OC
- FOM2 must be less than 2. OC
- W1 must be greater than W2. IC

The user likely needs a standard way to calculate time differences between important events or the amount of time an element of the system is in a certain state. To accomplish this in the settings file for SAE2, the user must type "time points:" and then follow that with a comma-separated list of conditional phrases that indicate specific events or durations. The following example illustrates the three currently implementable user inputs:

- time points: Ti when Current exceeds 2.5E6, Ttop when Voltage1 reaches its maximum, Tdur while Voltage2 is above 0.7*(Voltage2 max)

To ask SAE2 to do further manipulations of SCREAMER output and/or time points, the user must type simple statements relating the necessary time points or variable names, each statement or name occurring on its own line within the settings file. If the user wants the result or variable name to print to the log file, he or she must also type 'output' after it on the respective line:

- Telapsed Tstart to Timp output
- Im Current max output
- Vm Voltage at Timp
- Voltage2 max output
- Price = Volume*PriceperVolume IF output

Figures of merit, which should simply be functions of input and/or output variables with or without a scale factor, must be entered as, 'FOM#' = scale_factor*fn(in and/or out vars)' where '#' can be any non-space string of characters. Here are some examples:

- FOM1 = 10^-14*Im/Telapsed
- FOM2 = (Voltage2 max)/Vsafe
- FOM3 = MSP min

Appendix C is a complete example of a settings file.

2.2. Running SAE2.py

To run SAE2 after having completed the template and settings files, go to the terminal, change directories to the location of the SAE2.py. On Linux machines, type in the command:

```
python3.0 SAE2.py
```

The script will ask for the name of the file. Enter the name without the extension, for example, the title is TSZ08 in our case (underlined):

```
Frankin Orndorff-Plunkett forndor@sandia.gov

This script package tests a user defined parameter space for SCREAMER models
using .csv outputs.

Name?       TSZ08
```

The script will search the home folder for any file that has that title. After it has done that, it will prompt you to choose a folder:

```
Automatically configured for POSIX filesystem.

#1     /home/HomeBase/D0x/Work/SCREAMER

#2     /home/HomeBase/D0x/Work/SandiaWork

      Which directory?

(enter an integer)      1
```

In this case, #1 is the desired folder. Once SAE2.py has run and has finished writing the customized script, which will be named *Name*script.py, will be in that folder. Then enter the following command to the shell:

`python3.0 *Name*script.py`

for Linux machines or:

`*Name*script.py`

for Windows machines.

Name is the title of the file, in our case it was 'TSZ08'. If any errors occur, check that all variables are properly instantiated in the settings file and check that the substitution syntax is in the correct order with the correct tuple of names at the end of the file.

3. RESULTS OF PRELIMINARY OPTIMUM SEARCH WITH SAE2

3.1. Current Performance of Z-Refurbishment

With a 90 KV charge on the Marx generator, the ZR device uses water-dielectric switching to deliver ~22MA of current in ~130 ns to a surrogate load of initial inductance 5 nH inside the post-hole convolute and an inductance change of 4.6 nH. Although this performance is sufficient for high-energy physics experiments and provides an extreme environment to test materials, the machine's performance could be improved if the following losses can be mitigated:

- Closing water switches deposit energy through Joule-heating of the water dielectric.
- Time-dependent inductive impedance limits water switch performance.
- The initial capacity across the water switch causes a prepulse that ramps to 220 KV in the forward-going voltage pulse just after the water switches.
- Going to higher voltages involves increasingly longer gaps at lower average electric field. This results in high resistances and inductances which limit the output power pulse duration to greater than 100ns for charge voltages of greater than 4MV, increasing switch jitter and degrading driver precision.
- Hardware damage from shock waves originating from the water switches limits the lifetime of major components.

SCREAMER simulations indicate that magnetic switching has the potential to greatly reduce these losses.

Water-switching was chosen over magnetic switching in 1984 for PBFA II. In addition to having cheaply and easily manufactured components, water switching technology was more robust at the time and to this day is still more mature than magnetic switching. Although the COMET experiments validated magnetic switching at voltages and currents comparable to those produced in a module of Z and the problems with manufacturing the magnetic material have been solved, a single module testing program with realistic reflected energy is needed to explore the robustness of magnetic switching technology for future refurbishment of the Z-machine.

3.2. Optimization of ZR with Magnetic Switching Retrofit

3.2.1. Circuit Model and Consolidation into a Mega-Module

K. W. Struve developed a Z-Refurbishment circuit model for a GUI simulation code called BERTHA; J.P. VanDevender adapted it for the SCREAMER code and further developed it from the results of 3-D electromagnetic Emphasis simulations by M. F. Pasik and R. S. Coats. The magnetically switched option for potnetial Z refurbishment is documented in Appendix B and is modeled as the four levels (A through D) of Z, so each super-module represents N = 9 radially oriented modules. The parameters of the super-module's magnetic switch allow the voltage waveform V(t) before and after each switch to be the same as V(t) of the corresponding single modules. This means the switch time (duration to drive the switch through a flux change $\Delta\Phi$ to saturation) and the inductive rise-time (L_{sat}/Z) of the switch are the same as a single module. Therefore,

$$\int V\, dt = \Delta\Phi = w\,(R_o - R_i)\,\Delta B$$

and

$$L_{sat}/Z = \mu_o \ln(R_o/R_i)/(2\pi Z)$$

are independent of N, the number of modules represented. $Z_N = Z_1/N$ for N modules of impedance Z_1 in parallel and similarly, $L_{sat,N} = L_{sat,1}/N$. We meet these criteria by maintaining the outer radius R_o and the switch length w the same for the single module and super-module, choosing an effective inner radius $R_{i,N}$ and thus an effective ΔB_N for the super-module. Solving for $R_{i,1}$ as a function of $R_{i,N}$ and ΔB_N as a function of ΔB_1, gives

$$R_{i,1} = R_{i,N}^{\ N} / R_o^{N-1}$$

$$\Delta B_N = \Delta B_1\,(R_o - R_{i,1})/(R_o - R_{i,N}).$$

For the Metglas™ 2605SA, $\Delta B_1 = 2.8$ T. Therefore, for each choice of $R_{i,N}$, the corresponding effective ΔB_N is computed for the specified configuration's hysteresis profile. In addition, the value of the safe operating voltage $V_{safe} = 0.8\ V_{Breakdown}$ is computed by using the single module value of $R_{i,1}$.

The post-hole convolute then combines the four levels (A-D) into a single disk feed, which has an inductance of 5nH including 1nH in the return current can of the imploding plasma load. Since the investigators use the Z machine for a wide variety of experiments, we chose one

of the most challenging loads for these simulations. The surrogate load consists of a Z-pinch with a 1cm initial height, 3mm initial radius, mass of 200mg and a convergence ratio of 10. Thus the inductance changes 4.6nH based on the expression $\Delta L = 2*\text{length_in_cm}*\ln(R_i/R_f)$.

3.2.2. Optimization: A Safe and Better Performing Retrofit

A credible candidate for a Z retrofit with magnetic switching should satisfy two operations prerequisites: it must have a safe operating maximum voltage in the pulse forming line between the two switches and it must be able to serve as a *retrofit* for the current ZR device. These two criteria are built into the sample SAE2 settings file in Appendix C as a function that finds the electric length of the intervening PFL based on the widths of the magnetic switches and a plain-English inequality, "Vsafe must be greater than VPFL2 max." Since the optimum ZR model should also increase the implosion current I_{imp} and decrease the implosion time t_{imp}, the Figure of Merit (FOM) that I used for each run was FOM1 = $10^{-14}*I_{imp}/t_{imp}$. Analyzing the runs that satisfied the two operational constraints and maximized FOM1, I found new baseline configurations for the ZR model with magnetic switching and improved on the initial design provided by J.P. VanDevender.

In finding the new optimum from the output file, one must weigh the advantages and disadvantages of each configuration. We ran two cases: one with only the first water switch replaced by a magnetic switch to reduce energy losses and one with a second magnetic switch added for an additional stage of pulse compression.

The following are the output waveforms are from the single switch run that satisfied the two operational constraints and gave the highest FOM1 ratio. Implosion current and rise-time were 29.5MA and 89ns respectively. The optimum parameters of the magnetic switch for a single module follow: outer radius = 0.8382 m, inner radius = 0.229 m, width = 0.55 m (5 cores of 10 cm wide Metglas ribbon with 0.5 cm between cores), and ΔB=2.8 T. The packing fraction is 0.5. The water-oil barrier on each side is 4.2 cm thick and the lines are tapered on each side of the switch housing for field grading. The radial grading of the electric field by the magnetic switch adjacent to the barrier is very strong and must be included in the final design. The impedance downstream of the switch is 2.34 Ohms for each line for 21.5 ns to prepare for a possible second

magnetic switch option. The rest of the accelerator parameters (for N=9 supermodules) are given in the SCREAMER input file of Appendix B.

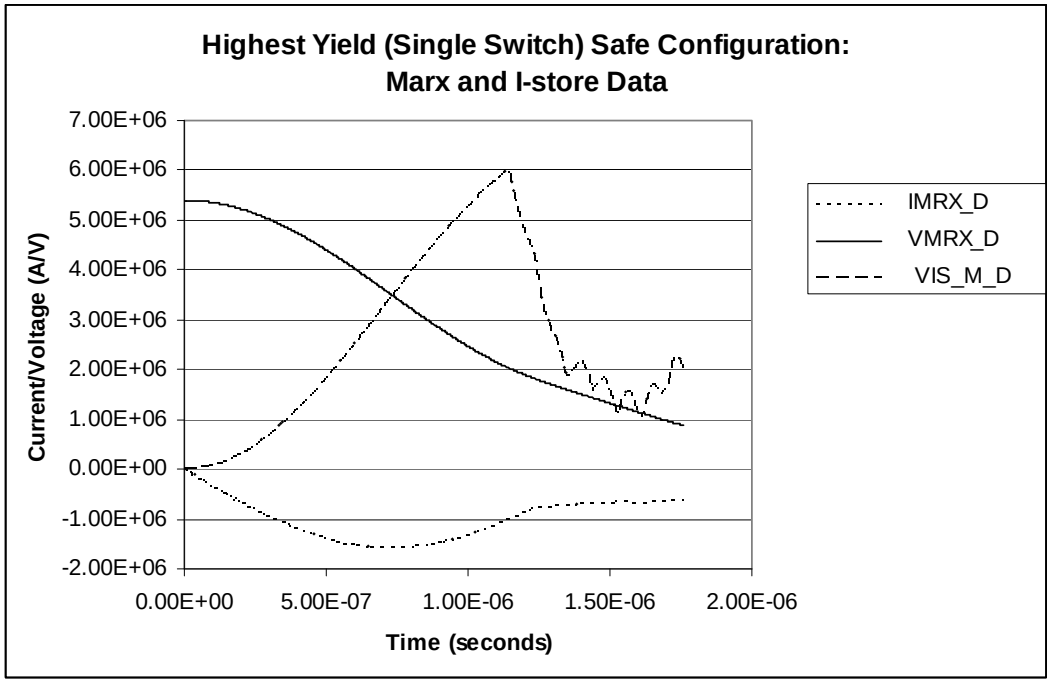

Figure 1.1 Marx Capacitor Bank and Intermediate Store.

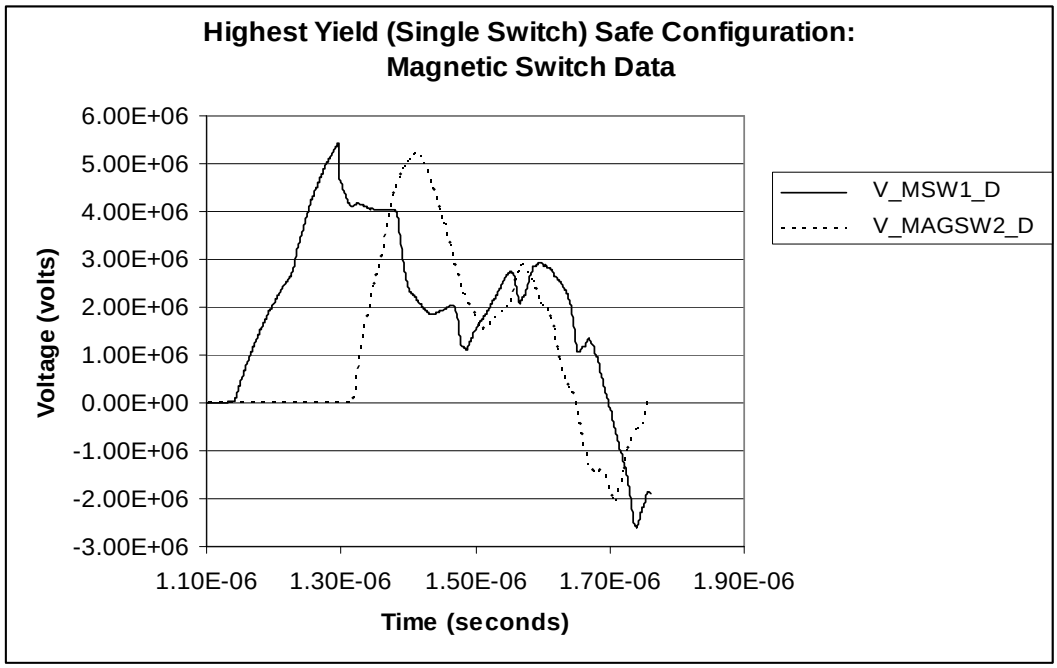

Figure 1.2 Magnetic Switches 1 & 2

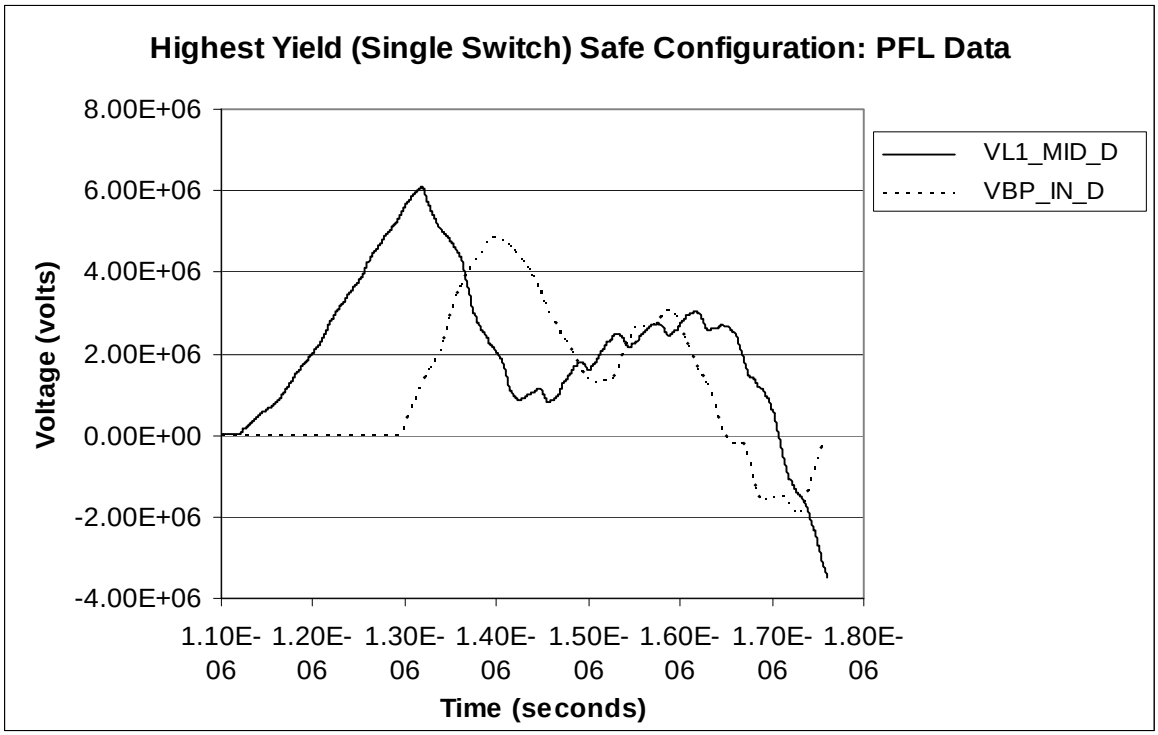

Figure 1.3 Pulse-Forming-Lines 1 & 2.

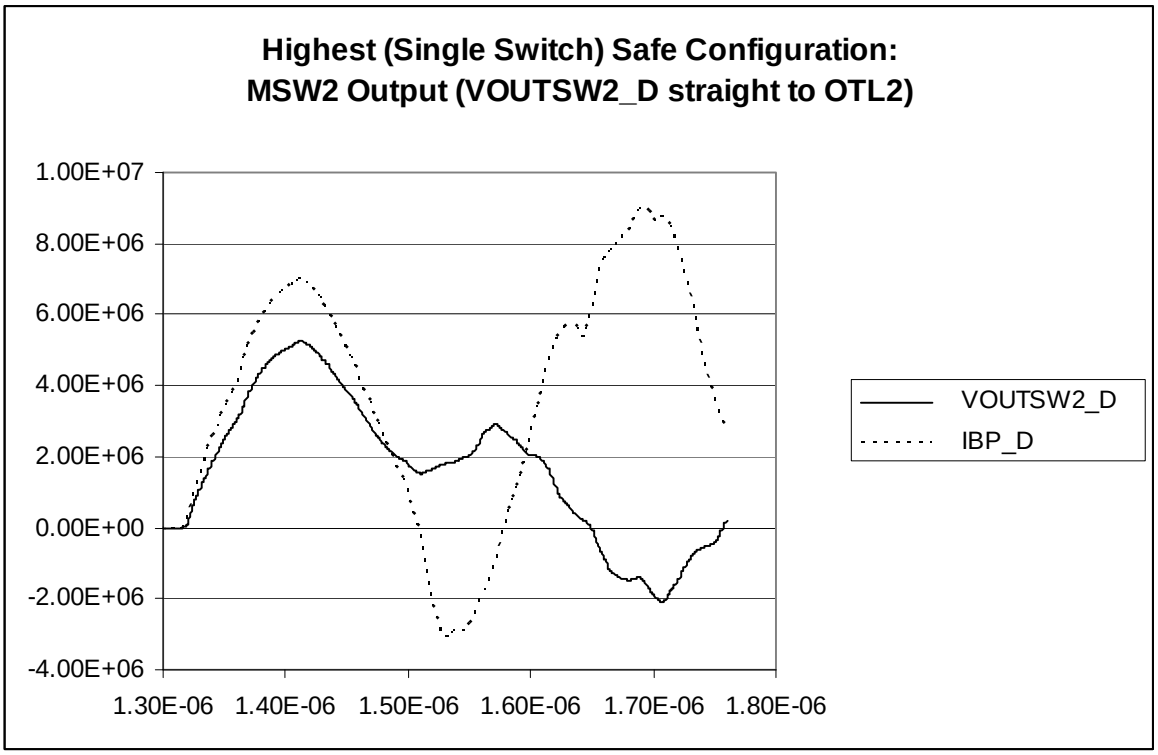

Figure 1.4 Output into the Bi-plate output transmission line.

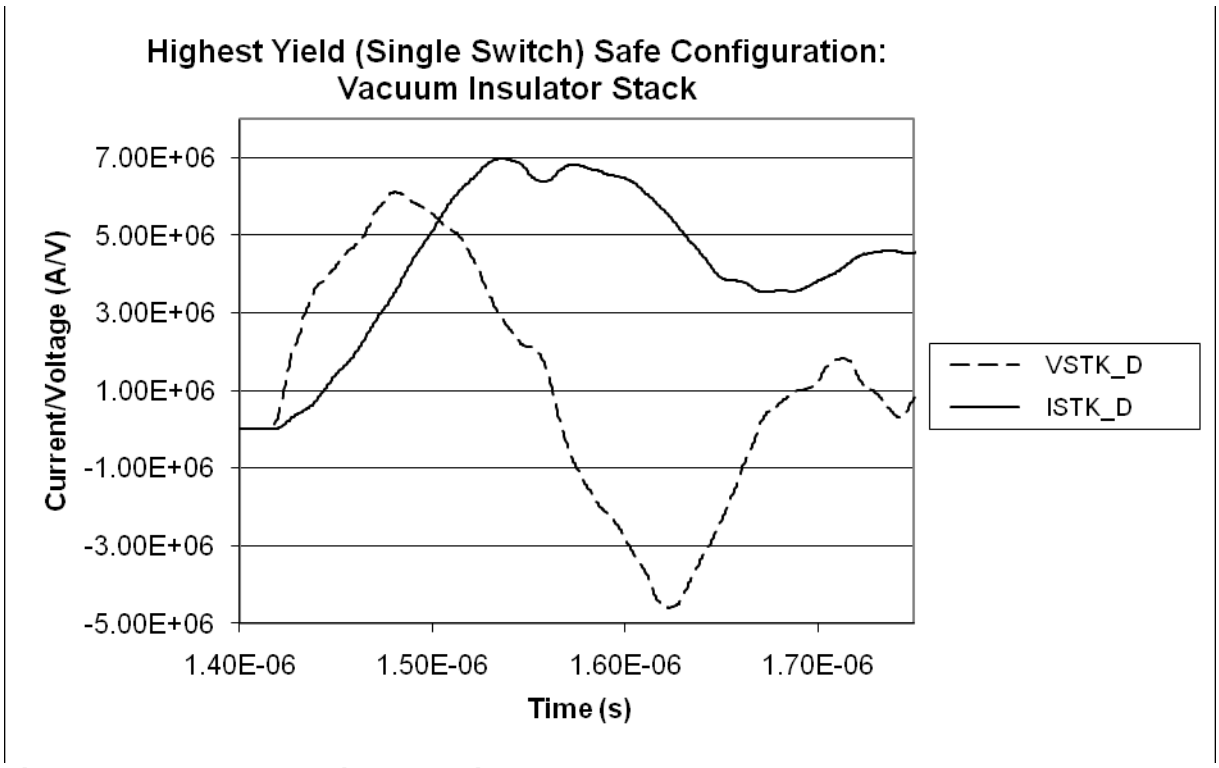

Figure 1.5. Vacuum Insulator Stack Data.

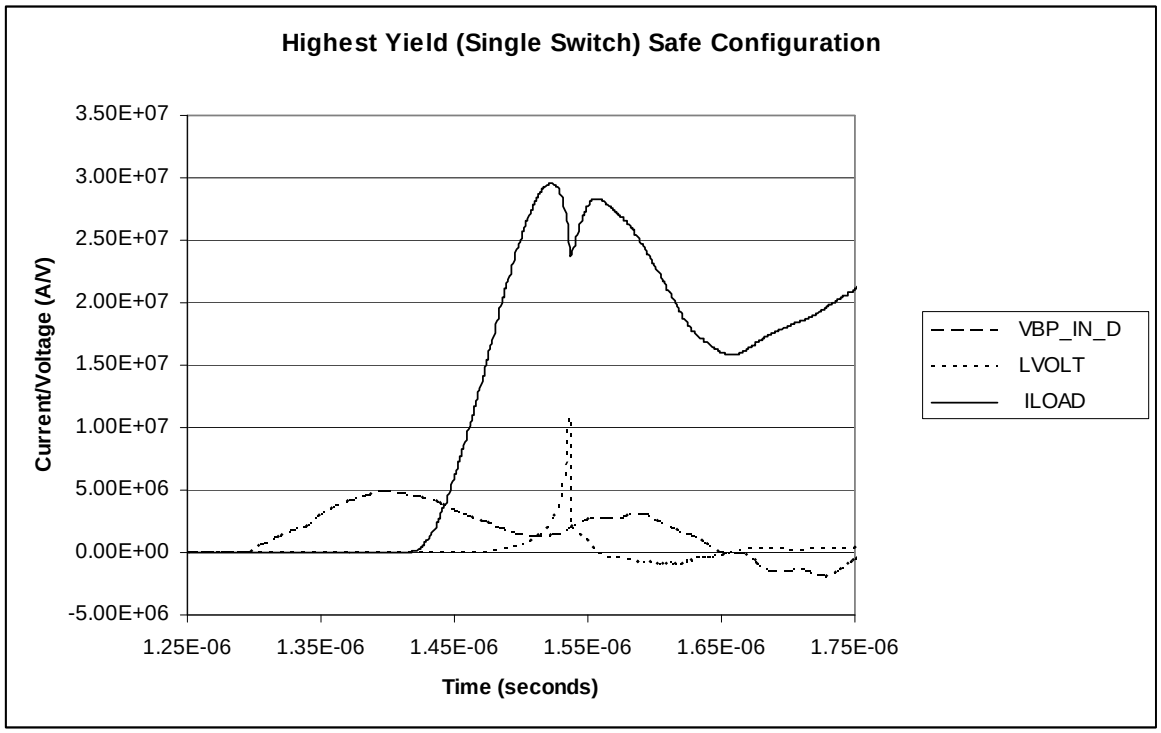

Figure 1.6 Load and PFL2 Data.

The best new baseline optimum model with a safe VPFL2 profile and a reasonable Vstk profile achieves a maximum current of 29.5MA with 89ns rise-time. This still is a marked improvement for the implosion performance, a 31% reduction of rise-time and a 34% increase in implosion current over the current Z-Refurb device. These improvements strongly warrant further study of magnetic switching pulse compression techniques for pulsed-power design.

This set of tests run by SAE2 served as 'over-tests', that is, it used 90kV in the initial Marx capacitor banks to stress the system beyond its normal running conditions. For the sake of completeness, I ran the new baseline case with 85, 80, 75, and 70kV to compare them with the 90kV case and with each other. The results from these runs are in the table below:

Marx Charge Voltage (KV)	Load Current (MA)	T_implosion (ns)	V_max/V_BD
90	29.5	103.1	0.832
85	28.5	105.2	0.803
80	27.4	108.0	0.773
75	26.2	111.1	0.748
70	24.8	115.3	0.712

Table 1. Effect of changing the Marx charge voltage. The load is the same surrogate load described above for all cases.

The single-switch design was optimized for 90 KV charge on the Marx capacitors. At 90 KV the voltage on PFL1 reaches 83% of the breakdown voltage, which is 3% larger than Stygar, et al's[8] criteria for a safe voltage based on point-plane breakdown. Therefore, 85 KV is the maximum charge voltage consistent with the constraints from Stygar, et al. The load current increases with--but is not quite proportional to—increasing charge voltage; therefore, performance decreases favorably with decreasing charge voltage. As expected, the safety factor on water breakdown improves with decreasing charge voltage. The implosion time is measured as the difference between the time at which the liner has converged a factor of 10 and the initiating time—as extrapolated from the currents at 5 MA and 20 MA. The implosion time decreases with increasing charge voltage.

[8] W. A. Stygar, M. E. Savage, T. C. Wagoner, L. F. Bennett, J. P. Corley, G. L. Donovan, D. L. Fehl, H. C. Ives, K. R LeChien, G. T. Leifeste, F.W. Long, R. G. McKee, J. A. Mills, J. K. Moore, J. J. Ramirez, B. S. Stoltzfus, K.W. Struve, and J. R. Woodworth PRST-Accelerators and Beams 12, 010402 (2009)

When the second magnetic switch was added for an additional stage of pulse compression, the implosion time was reduced to 82 ns and the load current was 26.2MA for the same surrogate load. The optimum parameters of the first magnetic switch for a single module follow: outer radius = 0.8382 m, inner radius = 0.3472 m, width = 0.33 m. The optimum parameters of the second magnetic switch for a single module follow: outer radius = 0.8382 m, inner radius = 0.3472 m, width = 0.28 m, and Z_{PFL2} = 2.34 Ohms. The rest of the accelerator parameters (for N=9 super modules) are given in the SCREAMER input file of Appendix B.

The following computed waveforms show that a second magnetic switch does increase the output power but the rest of Z is not able to support the increased power. The voltage on the vacuum insulator reaches 8.9 MV—well above its flashover voltage. A significant redesign of Z from the output of the first magnetic switch to the load is required to take advantage of benefits of a second magnetic switch.

The forward going current just after the second switch position (where the individual line impedance is 5.85 Ohms) for the single switch and double switch configurations are shown in Figure 1.7. The improved rise time and peak current with the double switch configuration is evident but the long and low current tails shows the faster dI/dt comes at a cost in energy efficiency.

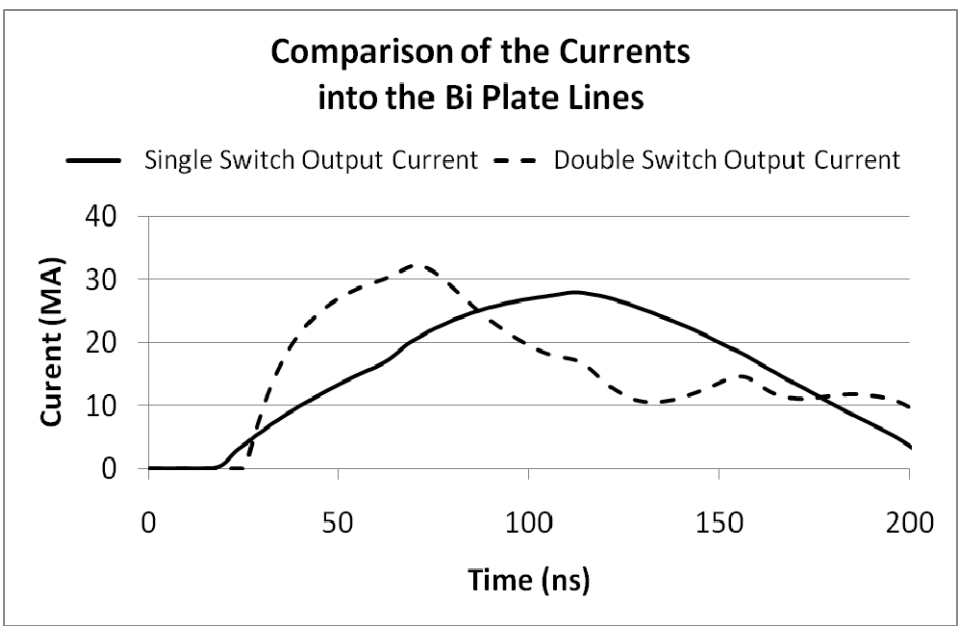

Figure 1.7 The output currents just after the second switch position for the single and double switch configurations

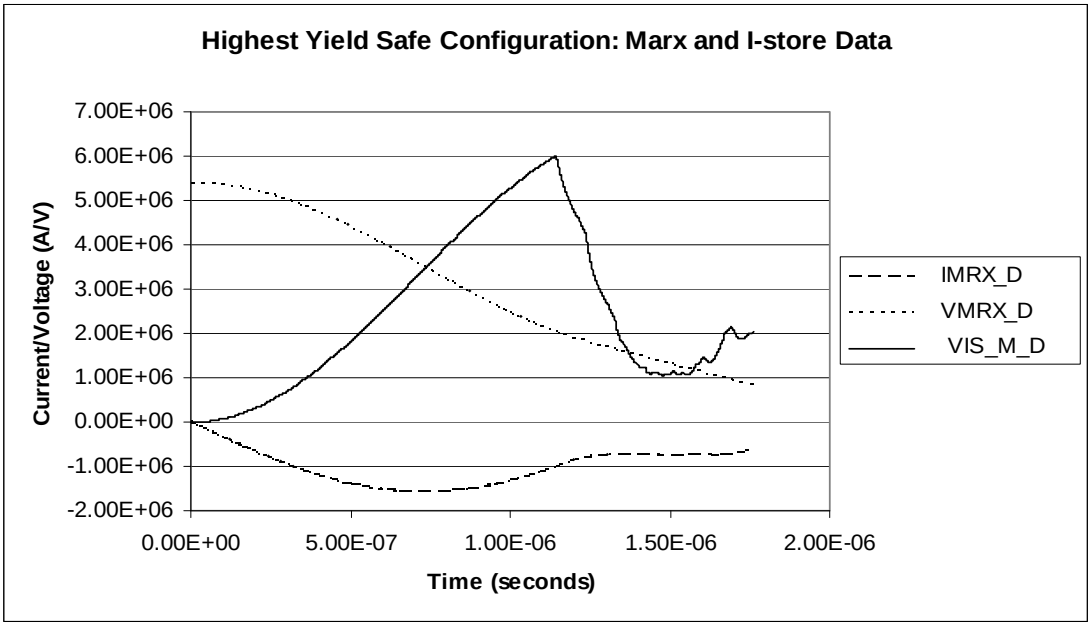

Figure 1.8 Marx Capacitor Bank.

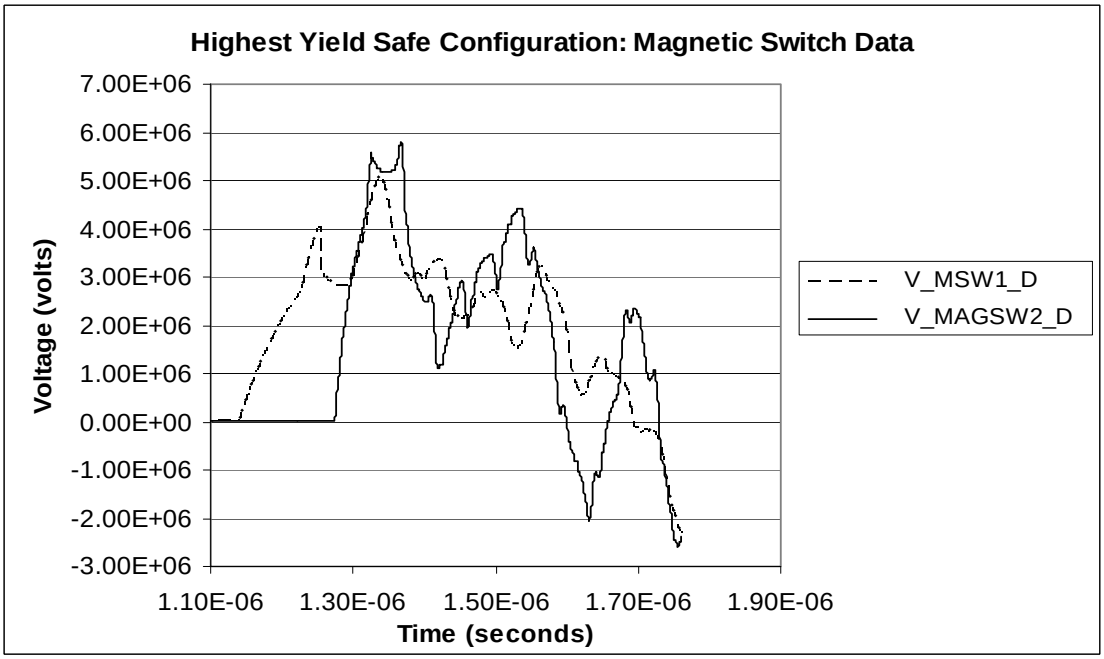

Figure 1.9 Magnetic Switches 1 & 2

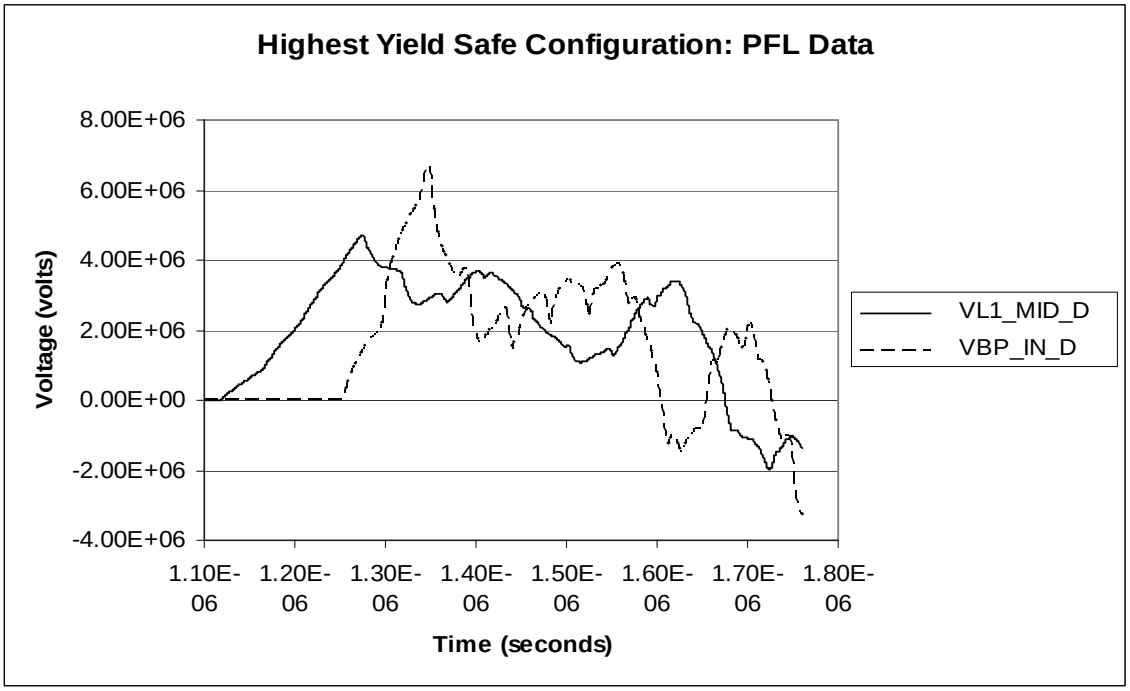

Figure 1.10 Pulse-Forming-Lines 1 & 2.

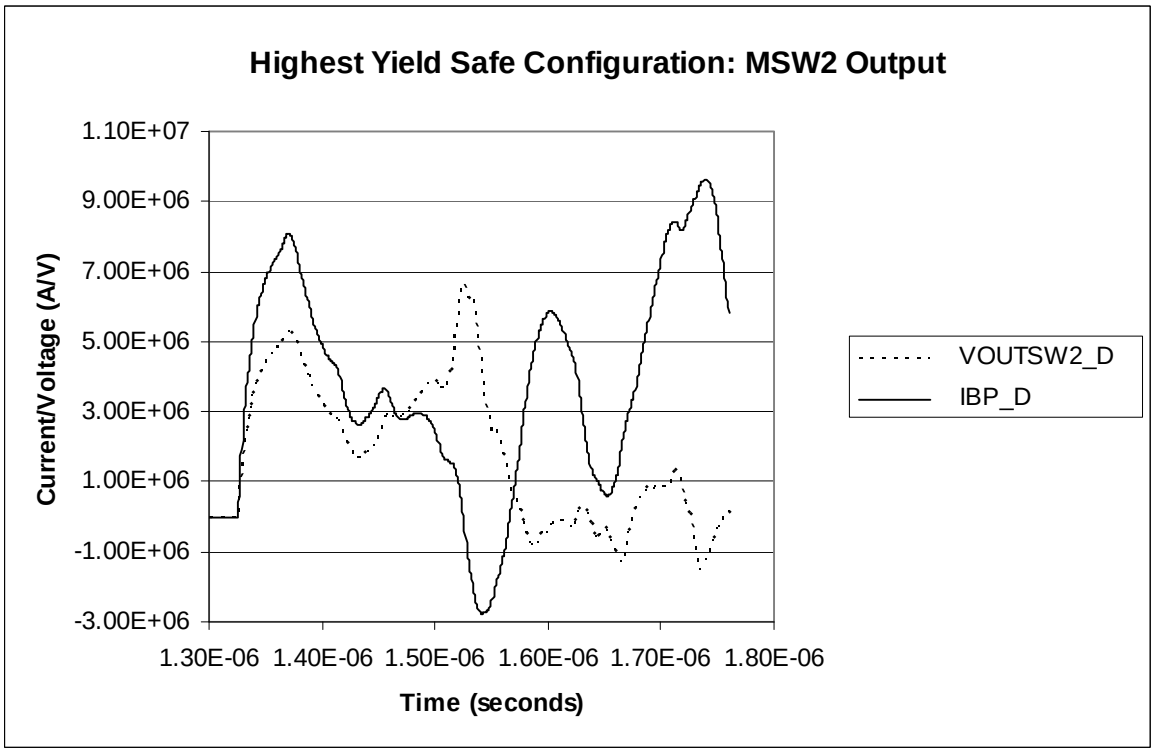

Figure 1.11 Output of 2nd Magnetic Switch.

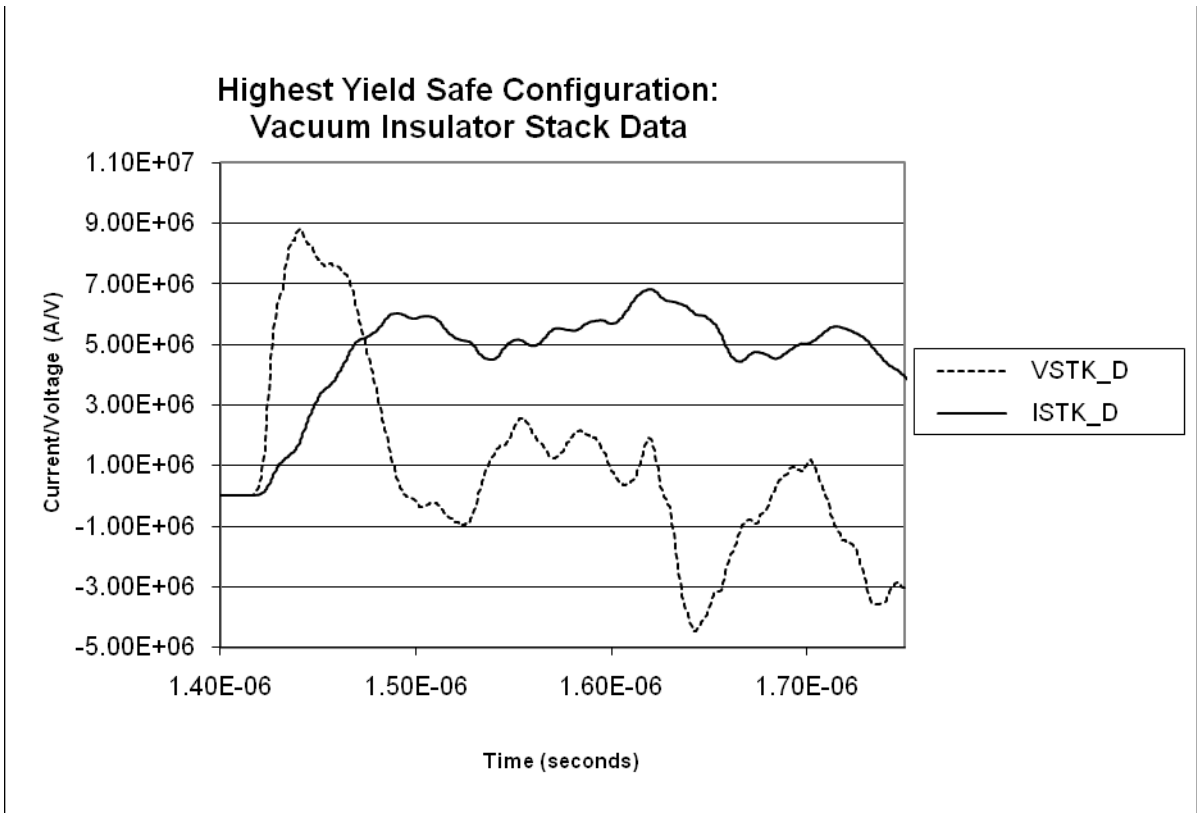

Figure 1.12 Vacuum Stack Data.

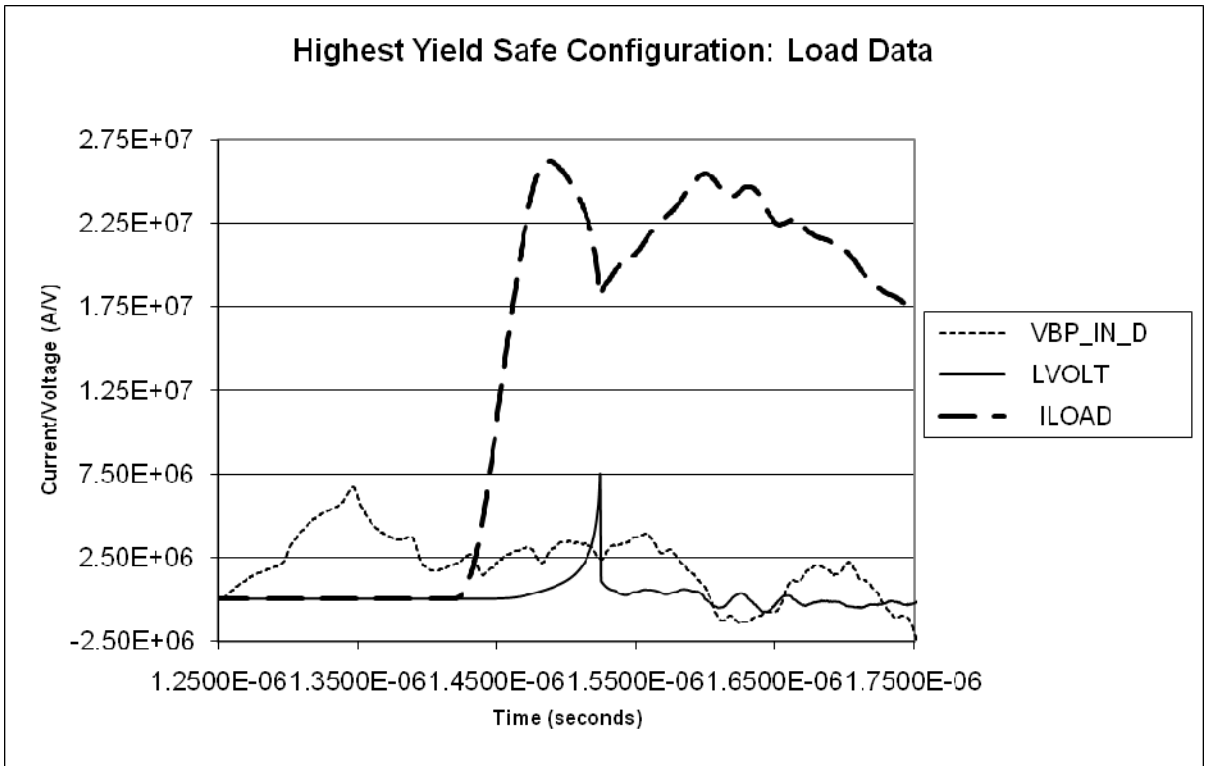

Figure 1.13 Load and PFL2 Data.

In the two-switch configuration, the first switch switches earlier than it does with a single switch and improves the safety factor for breakdown of the first switch; the maximum voltage at 90 KV charge is only 77.4% of the breakdown voltage. Breakdown in the second PFL becomes the limiting factor and the design optimizes with the maximum voltage = 80% of the breakdown voltage in accord with reference 8.

Although SAE2 can serve an investigator well as a preliminary study of a pulsed-power system simulated with SCREAMER, its search algorithm is not as robust as other more complicated approaches. The DAKOTA package can improve the robustness, but that is quite challenging software to learn. Therefore, I chose to develop SAE2 as a user friendly and easy to implement approach. If the Sandia National Laboratories pulsed-power community needs a more robust search algorithm for SCREAMER, SAE2 can be extended to a user-friendly particle-swarm-optimization script.

4. CONCLUSIONS

For investigators needing to execute a systematic, 1st order, preliminary study of a pulsed power system, SAE2is a user-friendly choice—such as Z-Refurbishment with Linear Transformer Drivers—in the SCREAMER simulation code, SAE2 is a user-friendly choice. SAE2 found the new baseline optimum with 29.5MA peak current, 89ns implosion time and a safe profile for PFL2 illustrates the power of simple gridded optimum search.

In addition, the pulsed-power community would benefit by further investigating magnetic switching as an approach to pulse compression. The processes and phenomena involved in magnetic switching are more adequately modeled and understood from first principles than those of water switching. Additionally, the mechanical shock—which weakens the structure of the system during each shot—would diminish greatly and the efficiency of power delivery to the load would be enhanced.

5. Works Consulted

Harjes, H.C., Penn, K.J., Mann, G.A. and Neau, E.L., "Investigations into the Design of Multi-terawatt Magnetic Switches" in *Proc. 6th IEEE Pulsed-Power Conf.* Arlington, VA, June 1981, pp.262-268

Johnson, W.A. and Warne, L.K., Simple Circuit Models for the Terminal Properties of a Nonlinear Closing Switch in *Laser and Particle Beams,* vol. 10, p375, 1992.

Neau, E.L., 1983, "COMET: A 6MV, 400kJ, Magnetically Switched Pulse-Power Module.", *Proceedings of the 4th IEEE Pulsed-Power Conference,* Albuquerque, NM, pp. 437-440.

Neau, E.L., Woolston, T.L. and Penn K.J., 1984, "COMET-II, A Two-Stage Magnetically Switched Pulsed-Power Module.", Conf. Record of the 1984 Sixteenth Power Modulator Symposium, June 18-20 1984, pp 292-294.

Power, A.J., Choi, P., Dolan, J.E., Parameter Studies for a Coaxial Saturable Inductor Magnetic Switch in *Journal of Applied Physics*, vol. 68, Issue 2, pp.840-844, 1990. [Published]

Appendix A: Scripting Conventions

Notational Conventions

- **To specify sequence of evaluation in a mathematical expression, use parentheses to contain the pieces that python3.0 will evaluate. The most deeply nested parenthetical will be evaluated first.**
- **Use the carat (^) or a double star (**) to denote raising a certain quantity to a power. For example Teff to the N-th power is written (Teff)^(N) or (Teff)**(N)**
- **Arithmetic operations are as follows: addition +, multiplication *, exponentiation **/^ (as above), division /.**

Usable Mathematical Functions and Constants

- **exp(arg) – natural base 'e' exponential**
- **sqrt(arg) – square root**
- **log(arg) – natural logarithm**
- **log10(arg) – logarithm base 10**
- **cos(arg) – cosine**
- **sin(arg) – sine**
- **tan(arg) – tangent**
- **asin(arg) – arcsine**
- **acos(arg) – arccosine**
- **atan(arg) – arctangent**
- **acosh(arg) – inverse hyperbolic cosine**
- **asinh(arg) – inverse hyperbolic sine**
- **atanh(arg) – inverse hyperbolic tangent**
- **cosh(arg) – hyperbolic cosine**
- **sinh(arg) – hyperbolic sine**
- **tanh(arg) – hyperbolic tangent**
- **eNB – User must enter this for natural base 'e'**
- **pi – User must enter this for the constant 'π'**

Appendix B: Sample Template File

```
Z REFURB -- ALL LEVELS -- High Inductance Load  10-21-09
! Water switch 1 with ZR disk MITLs, 4 nH feed inductance, 1 nH can
inductance.
!SCREAMER does not compute the voltage across capacitance across water
switches in end branches (for levels A, B, and C).
! High L-dot load of 1 cm long, 3mm radius cylinder.
TIME-STEP 0.03E-9
RESOLUTION-TIME 1.E-9
END-TIME 1.760E-6
NUMBER-PRINTS 1
EXECUTE-CYCLES ALL
ECHO NO
MAX-POINTS 3000
!
!
! Pace VanDevender
! This is based on Ken Struve's Bertha Model of 200811
!
! THE MARX GAPS HAVE 30 UNITS WITH A SPACING OF 2.54 CM
! PER GAP OR 91.44 CM OF SF6. 9 PARALLEL SWITCHES ARE USED FOR
! THE 9 PARALLEL MARXES
!
! THE MARX CAP STORE COMES FIRST WITH 60, 74, OR 85.0 KV CHARGE,
! AND 60 2.2 UFD CAPS IN EACH OF 9 MARXES.
!
! ======================================================================
! D L E V E L
! ======================================================================
!
BRANCH
!
RCG 1E12 390.0E-9
INITIAL VC1 5.4E6
! 6.00 MV for 100 kV
! 5.70 MV for 95 kV
! 5.40 MV for 90 kV
! 5.10 MV for 85 kV capacitor charge voltage
!     4.92 MV for 82 kV
!
CSV IC1
$IMRX_D
!
CSV VC1
$VMRX_D
CSV PC1
$P_MARX
!
! THE FIRST PART OF THE MARX INDUCTANCE
RLS 0.0 1.222E-6
CSV EOUT
$EMRX_D
!
! MARX RESISTANCE
```

```
!RLS 0.155 0.0 *****Bertha is a third as much****************
RLS 0.0555 0.0
!
! THE PARALLEL MARX RESISTANCE
RCG 63.0 0
!
! NEXT IS THE SECOND PART OF THE MARX INDUCTANCE-THE CONNECTIONS
!RLS 0 227.6E-9*********Bertha is 50 percent more inductance including
transition in IS********
RLS 0 340.6E-9
!
!
!
! **************** INTERMEDIATE STORE **************
!
CSV VOUT
$VIS_IN_D
!TRLINE LINEAR 45.0E-9 .4244*******Bertha has the taper modeled--approximated
here
TRLINE LINEAR 10.3E-9 0.55
TRLINE LINEAR 35.375E-9 0.422
!
! WATER RESISTANCE FOR THE TWO MIDDLE SECTIONS
!RCG 89 0 *********Bertha had slightly higher values*************
RCG 104. 0.0
!RCG  52    0.0
!
CSV VR1
$ VIS_M_D
!
!TRLINE LINEAR 45.0E-9 .4244*****Bertha has the taper modeled--approximated
here
TRLINE LINEAR 35.375E-9 0.4222
TRLINE LINEAR 10.3E-9 0.55
!
! **************** THE MAIN LASER TRIGGERED SF6 SWITCH AND ITS INDUCTANCE
!
! THE SF6 LASER SWITCH
CSV VOUT
$VIS_SW_D
!
RLS 0.0 42.7E-9
RLS 1E6 0.0
VARIABLE R2 EXP MODEL
! Ropen Rclose Tswitch Tau Zswitch
1.1E3 0.133 1098.E-9 1.2E-8 0.42
!
!
CSV VR2
$V_LTS
CSV Vout
$VL1_IN_D
CSV IOUT
$IL1_IN_D
! ********************** LINE 1 ********************
```

```
!
!RLS 0.0 8.23E-9*****************This is third part of LTS Inductance quoted
above.
RLS 0.0 26.4E-9 !Bertha value.
!TRLINE LINEAR 21.0E-9 .3
!Bertha has a taper included here at 0.3 Ohms.
TRLINE LINEAR 21.550E-9 .3
CSV VOUT
$VL1_MID_D
RCG 522. 0.0
!RCG 18.27 0.0
TRLINE LINEAR 21.550E-9 .3
!RCG 1.E9 0.0******Bertha has a more realistic parallel impedance.
!
! ****************** LINE 1 Water SWITCHES ****************
! ****************** LINE 1 Magnetic SWITCHES ****************
!THE FOLLOWING IS FOR a Magnetic Switch MS1 with 4 cores of Metglas 2605SC 10
cm wide each and 9 lines in parallel.
!Bsat effective is Bsat*(ro-ri,1)/(ro-ri,9)=15.36 T.
RLS 0.0 1.1E-9 !Inductance of taper
RLS 0.0 1.22E-9 !Inductance of barrier
CSV Vout
$V_MSW1_D
RLS 0.0 5.556E-9
VAR L2 MSW
!PF Ri Ro W H1 Hsat Hrev Bsat
0.5 %f 0.8382 %f 0.2 0.28 0.10 %f
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 1.1E-9 !Inductance of taper
!
! ********************* PFL2 **********************
!
CSV VOUT
$VBP_IN_D
TRLINE LINEAR %e %f
!
RCG 1.E6 0.0
CSV VR1
$V_MagSw2_D
! ****************** LINE 2 Magnetic SWITCHES ****************
!THE FOLLOWING IS FOR a Magnetic Switch MS1 with 2 cores of Metglas 2605SC
<=10 cm wide each and 9 lines in parallel.
!Bsat effective is Bsat*(ro-ri,1)/(ro-ri,9)=15.36 T.
RLS 0.0 1.1E-9 !Inductance of taper
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 5.556E-9
VAR L2 MSW
!PF Ri  Ro   W   H1 Hsat Hrev Bsat
0.5 %f .8382 %f 0.2 0.28 0.10 %f
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 1.1E-9 !Inductance of taper
!
!
! WATER RESISTANCE
RCG 1142.0 0
```

```
!
CSV VOUT
$VOutSW2_D
CSV IOUT
$IBP_D
CSV POUT
$P_BP
!
!
! ******************OTL2 Down******************************
TRL LIN 37.E-9 %f
!
CSV VOUT
$VOTL2_D
CSV IOUT
$IOTL2_D
!
RCG 867. 0
!RCG 30.9 0
TRL LIN 20.E-9 %f

TRL LIN 5.5E-9 %f
!
!**********************Mixer******************
! Note: Unlike Bertha, we cannot join the modules together at the mixer.
! The 36 K Ohms parallel resitor is not needed.
!
!*****************Extra inductance for Modules D and A***WHY?
!
!************************Water Convolute************************
TRL LIN 11.8E-9 0.847
RCG 3564.0 0
TRL LIN 11.2E-9 0.847 1.1076
!
!**********************The water/vacuum interface**************
RCG 1.77E3 0.0
TRL LIN 9.9E-9 1.1076 1.5204
CSV VOUT
$VSTK_D
CSV IOUT
$ISTK_D
CSV POUT
$P_STKD
!
!**************************Vacuum side of the plastic************
!
RLS 0.0 0.64E-9
!
! THE D MITL STARTS HERE
!
TRL LIN 0.365E-9 17.900 17.9 1.E-10
TRL LIN 0.476E-9 12.4 12.4 1.E-10
TRL LIN 0.471E-9 4.9 4.9 1.E-10
TRL LIN 3.56E-9 3.28 3.28 1.E-10
TRL LIN 1.0E-9 3.39 3.39 1.E-10
```

```
TRL LIN 0.508E-9 3.62 3.62 1.E-10
TRL LIN 0.504E-9 4.26 4.26 1.E-10
CSV IOUT
$IMITL_D
!
! The post-hole-convolute (PHC)
RLS 0.0 2.33E-9
RCG 1.0E6 0.0
!
ENDBRANCH ! Attach level C
RLS 0.0 1.12E-9
RCG 1.0E6 0.0
!
ENDBRANCH ! Attach level B
RLS 0.0 0.4E-9
RCG 1.0E6 0.0
!
ENDBRANCH ! Attach level A
!
!
RLS 0.0 0.79E-9
!
!
CSV EOUT
$E_PHC
!
CSV IOUT
$I_UP
CSV VOUT
$ V_PHC
RCG 0.0001 1.00E-10
VARIABLE R1 POS-MODEL
! TSW CURSW TOPEN ZFLOW GSWMIN GSWMAX CBFLAG
 2.0E-9 1.0E2 2.E-9 0.25 .0001 10000 0
!
CSV POUT
$P_CONV
!
RCG 1000. 0.0
!VARIABLE R1 DEC-MODEL **Changed on 112709
! Ropen Rclose Tswitch Tau for Chris Jennings Model of PHC Loss
!1000. 0.3 1.575E-6 2.E-9 **Changed on 112709
CSV R1
$R_PHC_LOSS
CSV IR1
$I_PHC_LOSS
!
! THE INNER MITL INDUCTANCE
RLS 0 4.E-9
CSV EOUT
$E_LOAD
!
! THE INITIAL PRS INDUCTANCE
RLS 0 1.0E-9
!
```

```
! ** ACTIVE FOIL LOAD, ro LENGTH mass rmin
CYLFOIL 0.30E-2 1.0E-2 200.E-6 0.3E-3
!
! FOIL PARAMETERS
!
CSV L2
$ FINDU
CSV EL2
$ LME
CSV VL2
$LVOLT
CSV IL2
$ ILOAD
CSV FKE
$ LKE
CSV FVEL
$LVEL
CSV FRAD
$ LRADIUS
CSV PIN
$P_LOAD
!
!
RCG 1E-12 0.

!
!
! END OF MAIN BRANCH AND LEVEL D
!
!
!
!BRANCH     ! ****************** LINE 1 WATER SWITCH Capacity FOR LEVEL D
****************
!
!RCG 1.E6 7.2E-9
!
!BRANCH     ! ****************** LINE 2 WATER SWITCHES FOR MODULE D
****************
!
!RCG 1.E6 3.6E-9
!
! ======================================================================
! C L E V E L
! ======================================================================
!
BRANCH
!
!
RLS 0.0 1.90E-9
!
CSV IOUT
$IMITL_C
!
TRL LIN 0.500E-9 4.37 4.37 1.E-10
TRL LIN 0.508E-9 3.73 3.73 1.E-10
```

```
TRL LIN 1.0E-9 3.45 3.45 1.E-10
TRL LIN 3.81E-9 3.30 3.3 1.E-10
TRL LIN 0.345E-9 6.93 6.93 1.E-10
TRL LIN 0.463E-9 8.92 8.92 1.E-10
TRL LIN 0.608E-9 9.76 9.76 1.E-10
TRL LIN 0.607E-9 14.53 14.53 1.E-10
!***************************** The C disk MITL
!
RLS 0.0 0.64E-9
!********************Vacuum side of the plastic**************
!
TRL LIN 9.90E-9 1.1076 1.5204
CSV VOUT
$VSTK_C
CSV IOUT
$ISTK_C
CSV POUT
$P_STKC
!
RCG 1.77E3 0.0
! The water/vacuum interface
!
TRL LIN 11.2E-9 0.847 1.1076
RCG 3564.0 0
TRL LIN 11.8E-9 0.847
!****************************Water Convolute*******************
!
!****************************The Water Mixer******************
!
TRL LIN 5.5E-9 %f
TRL LIN 20.E-9 %f
CSV VOUT
$VOTL2_C
CSV IOUT
$IOTL2_C
!
RCG 867. 0
!
TRL LIN 37.E-9 %f
! ******************OTL2 Down******************************
RCG 1142.0 0
!
CSV VR1
$VOutSW2_C
! ****************** LINE 2 Magnetic SWITCHES ****************
!THE FOLLOWING IS FOR a Magnetic Switch MS1 with 2 cores of Metglas 2605SC
<=10 cm wide each and 9 lines in parallel.
!Bsat effective is Bsat*(ro-ri,1)/(ro-ri,9)=15.36 T.
RLS 0.0 1.1E-9 !Inductance of taper
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 5.556E-9
VAR L2 MSW
!PF Ri Ro     W  H1  Hsat Hrev Bsat
0.5 %f 0.8382 %f 0.2 0.28 0.10 %f
RLS 0.0 1.22E-9 !Inductance of barrier
```

```
RLS 0.0 1.1E-9 !Inductance of taper
!
!
!RCG 1.E6 0.0
!CSV VR1
!$V_MagSw2_C
!
! ********************* PFL2 **********************
TRLINE LINEAR %e %f
!
! ****************** LINE 1 Magnetic SWITCHES ****************
!THE FOLLOWING IS FOR a Magnetic Switch MS1 with 4 cores of Metglas 2605SC 10
cm wide each and 9 lines in parallel.
!Bsat effective is Bsat*(ro-ri,1)/(ro-ri,9)=15.36 T.
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 5.556E-9
VAR L2 MSW
!PF Ri Ro W H1 Hsat Hrev Bsat
0.5 %f 0.8382 %f 0.2 0.28 0.10 %f
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 1.1E-9 !Inductance of taper
!
!
!
! ********************** LINE 1 Is BELOW *********************
!
$VL1_SW_C
TRLINE LINEAR 21.550E-9 .3
CSV VOUT
$VL1_MID_C
RCG 522. 0.0
!RCG 18.27 0.0
TRLINE LINEAR 21.550E-9 .3
$VL1_IN_C
!
RLS 0.0 26.4E-9
! ***** THE MAIN LASER TRIGGERED SF6 SWITCH AND ITS INDUCTANCE ******
!
RLS 1E6 0.0
VARIABLE R2 EXP MODEL
! Ropen Rclose Tswitch Tau Zswitch
!1.1E3 0.133 1108.E-9 1.2E-8 0.42
1.1E3 0.133 1087.E-9 1.2E-8 0.42
CSV VOUT
$VLSW_C
!
RLS 0.0 42.7E-9
! ******************* INTERMEDIATE STORE ******************
!
TRLINE LINEAR 35.375E-9 0.4222
TRLINE LINEAR 10.3E-9 0.55
CSV VOUT
$VIS_C
! WATER RESISTANCE
RCG 104 0.0
```

```
!RCG  52    0.0
TRLINE LINEAR 35.375E-9 0.422
TRLINE LINEAR 10.3E-9 0.55
!
!
! ******************** MARX AND OIL SECTION *******************
!
! THE 2ND PART OF THE MARX INDUCTANCE-THE CONNECTIONS TO THE INTERFACE
RLS 0 340.6E-9
!
!
! THE PARALLEL MARX RESISTANCE
RCG 63.0 0
!
! FIXED MARX RESISTANCE
RLS 0.0555 0.0
!
! MARX R FROM THE 36 SF6 GAPS OF 1 INCH EACH AND THE 1ST PART OF THE MARX L
RLS 0.0 1.222E-6
CSV EOUT
$EMRX_C
!
!
RCG 1E12 390.0E-9
INITIAL VC1 5.4E6
!
CSV IC1
$IMRX_C
!
CSV VC1
$VMRX_C
!
! END OF BRANCH C
!
! ======================================================================
! B L E V E L
! ======================================================================
!
BRANCH
!
!
RLS 0.0 1.17E-9
!
CSV IOUT
$IMITL_B
!
TRL LIN 0.506E-9 4.05 4.05 1.E-10
TRL LIN 0.502E-9 3.19 3.19 1.E-10
TRL LIN 1.00E-9 2.80 2.8 1.E-10
TRL LIN 1.00E-9 2.55 2.55 1.E-10
TRL LIN 1.1E-9 2.41 2.41 1.E-10
TRL LIN 0.834E-9 4.86 4.86 1.E-10
TRL LIN 0.850E-9 11.33 11.33 1.E-10
!************************************The B disk MITL*********
!
```

```
RLS 0.0 0.549E-9
! Vacuum side of the plastic
!
TRL LIN 9.9E-9 0.923 1.267
CSV VOUT
$VSTK_A
CSV IOUT
$ISTK_A
CSV POUT
$P_STKA
!
RCG 1500.0 0
!**********************StackMITL_A*******************
!
!
TRL LIN 11.2E-9 0.705 0.923
RCG 2853 0.0
TRL LIN 11.8E-9 0.705
!****************Water Convolute**********************
!****************************The Water Mixer*****************
!
TRL LIN 4.7E-9 %f
TRL LIN 16.E-9 %f
CSV VOUT
$VOTL2_B
CSV IOUT
$IOTL2_B
!
RCG 1078. 0
!
TRL LIN 24.E-9 %f
! ******************OTL2 UP******************************
!
RCG 1142.0 0
!
CSV VR1
$VOutSW2_B
! ****************** LINE 2 Magnetic SWITCHES ****************
!THE FOLLOWING IS FOR a Magnetic Switch MS1 with 2 cores of Metglas 2605SC
<=10 cm wide each and 9 lines in parallel.
!Bsat effective is Bsat*(ro-ri,1)/(ro-ri,9)=15.36 T.
RLS 0.0 1.1E-9 !Inductance of taper
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 5.556E-9
VAR L2 MSW
!PF Ri Ro     W  H1  Hsat Hrev Bsat
0.5 %f 0.8382 %f 0.2 0.28 0.10 %f
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 1.1E-9 !Inductance of taper
!
!
RCG 1.E6 0.0
CSV VR1
!$V_MagSw2_B
!
```

```
! ********************* PFL2 **********************
TRLINE LINEAR %e %f

!
! ****************** LINE 1 Magnetic SWITCHES ****************
!
!THE FOLLOWING IS FOR a Magnetic Switch MS1 with 4 cores of Metglas 2605SC 10
cm wide each and 9 lines in parallel.
!Bsat effective is Bsat*(ro-ri,1)/(ro-ri,9)=15.36 T.
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 5.556E-9
VAR L2 MSW
!PF Ri Ro W H1 Hsat Hrev Bsat
0.5 %f 0.8382 %f 0.2 0.28 0.10 %f
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 1.1E-9 !Inductance of taper
! ****************** LINE 1 Water SWITCH Is ABOVE****************
!
!
! ********************** LINE 1 IS Below *********************
!
$VL1_SW_B
TRLINE LINEAR 21.550E-9 .3
!CSV VOUT
!$VL1_MID_B
RCG 522. 0.0
!RCG 18.27 0.0
TRLINE LINEAR 21.550E-9 .3
CSV VOUT
$VL1_IN_B
!
RLS 0.0 26.4E-9
! ***** THE MAIN LASER TRIGGERED SF6 SWITCH AND ITS INDUCTANCE ******
!
RLS 1E6 0.0
VARIABLE R2 EXP MODEL
! Ropen Rclose Tswitch Tau Zswitch
!1.1E3 0.133 1112.E-9 1.2E-8 0.42
1.1E3 0.133 1122.E-9 1.2E-8 0.42
CSV VOUT
$VLSW_B
!
RLS 0.0 42.7E-9
! ******************* INTERMEDIATE STORE ******************
!
TRLINE LINEAR 35.375E-9 0.4222
TRLINE LINEAR 10.3E-9 0.55
!CSV VOUT
!$VIS_B
! WATER RESISTANCE
RCG 104 0.0
!RCG  52    0.0
TRLINE LINEAR 35.375E-9 0.422
TRLINE LINEAR 10.3E-9 0.55
!
```

```
!
! ********************* MARX AND OIL SECTION *******************
!
! THE 2ND PART OF THE MARX INDUCTANCE-THE CONNECTIONS TO THE INTERFACE
RLS 0 340.6E-9
!
!
! THE PARALLEL MARX RESISTANCE
RCG 63.0 0
!
! FIXED MARX RESISTANCE
RLS 0.0555 0.0
!
! MARX R FROM THE 36 SF6 GAPS OF 1 INCH EACH AND THE 1ST PART OF THE MARX L
RLS 0.0 1.222E-6
!CSV EOUT
!$EMRX_B
!
!
RCG 1E12 390.0E-9
INITIAL VC1 5.4E6
!
!CSV IC1
!$IMRX_B
!
!CSV VC1
!$VMRX_B
!
! END OF BRANCH B
!
! ======================================================================
! A L E V E L
! ======================================================================
!
BRANCH
!
!
!
RLS 0.0 1.615E-9
!
!
CSV IOUT
$IMITL_A
!
TRL LIN 0.504E-9 3.97 3.97 1.E-10
TRL LIN 0.498E-9 3.08 3.08 1.E-10
TRL LIN 1.01E-9 2.70 2.70 1.E-10
TRL LIN 0.998E-9 2.46 2.46 1.E-10
TRL LIN 1.38E-9 2.33 2.33 1.E-10
TRL LIN 0.484E-9 5.36 5.36 1.E-10
TRL LIN 0.518E-9 11.9 11.9 1.E-10
!************************************The A disk MITL*********
!
RLS 0.0 0.549E-9
! Vacuum side of the plastic
```

```
!
TRL LIN 9.9E-9 0.923 1.263
CSV VOUT
$VSTK_A
CSV IOUT
$ISTK_A
CSV POUT
$P_STKA
!
RCG 1500.0 0
!**********************StackMITL_A*******************
!
TRL LIN 11.2E-9 0.705 0.923
RCG 2853 0.0
TRL LIN 11.8E-9 0.705
!****************Water Convolute**********************
!
!
!****************************The Water Mixer*****************
!
TRL LIN 4.7E-9 %f
TRL LIN 16.E-9 %f
CSV VOUT
$VOTL2_A
CSV IOUT
$IOTL2_A
!
RCG 1078. 0
!
TRL LIN 24.E-9 %f
! ******************OTL2 UP******************************
RCG 1142.0 0
!
!
CSV VR1
$VOutSW2_A
! ****************** LINE 2 Magnetic SWITCHES ****************
!THE FOLLOWING IS FOR a Magnetic Switch MS1 with 2 cores of Metglas 2605SC
<=10 cm wide each and 9 lines in parallel.
!Bsat effective is Bsat*(ro-ri,1)/(ro-ri,9)=15.36 T.
RLS 0.0 1.1E-9 !Inductance of taper
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 5.556E-9
VAR L2 MSW
!PF Ri Ro     W   H1 Hsat Hrev Bsat
0.5 %f 0.8382 %f 0.2 0.28 0.10 %f
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 1.1E-9 !Inductance of taper
!
!
RCG 1.E6 0.0
CSV VR1
$V_MagSw2_C
!
! ********************* PFL2 **********************
```

```
TRLINE LINEAR %e %f
!
! ****************** LINE 1 Magnetic SWITCHES ****************
!
!THE FOLLOWING IS FOR a Magnetic Switch MS1 with 4 cores of Metglas 2605SC 10
cm wide each and 9 lines in parallel.
!Bsat effective is Bsat*(ro-ri,1)/(ro-ri,9)=15.36 T.
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 5.556E-9
VAR L2 MSW
!PF Ri Ro W H1 Hsat Hrev Bsat
0.5 %f 0.8382 %f 0.2 0.28 0.10 %f
RLS 0.0 1.22E-9 !Inductance of barrier
RLS 0.0 1.1E-9 !Inductance of taper
!
!
! ********************** LINE 1 IS BELOW *********************
!
TRLINE LINEAR 21.550E-9 .3
!CSV VOUT
!$VL1_MID_A
RCG 522. 0.0
!RCG 18.27 0.0
TRLINE LINEAR 21.550E-9 .3
CSV Vout
$VL1_IN_A
!
RLS 0.0 26.4E-9
! ***** THE MAIN LASER TRIGGERED SF6 SWITCH AND ITS INDUCTANCE ******
!
RLS 1E6 0.0
VARIABLE R2 EXP MODEL
! Ropen Rclose Tswitch Tau Zswitch
!1.1E3 0.133 1117.E-9 1.2E-8 0.42
1.1E3 0.133 1112.E-9 1.2E-8 0.42
CSV VOUT
$VLSW_A
!
RLS 0.0 42.7E-9
! ******************* INTERMEDIATE STORE ******************
!
TRLINE LINEAR 35.375E-9 0.4222
TRLINE LINEAR 10.3E-9 0.55
!CSV VOUT
!$VIS_A
! WATER RESISTANCE
RCG 104 0.0
!RCG  52    0.0
TRLINE LINEAR 35.375E-9 0.422
TRLINE LINEAR 10.3E-9 0.55
!
!
! ******************** MARX AND OIL SECTION ******************
!
! THE 2ND PART OF THE MARX INDUCTANCE-THE CONNECTIONS TO THE INTERFACE
```

```
RLS 0 340.6E-9
!
!
! THE PARALLEL MARX RESISTANCE
RCG 63.0 0
!
! FIXED MARX RESISTANCE
RLS 0.0555 0.0
!
! MARX R FROM THE 36 SF6 GAPS OF 1 INCH EACH AND THE 1ST PART OF THE MARX L
RLS 0.0 1.222E-6
!CSV EOUT
!$EMRX_A
!
!
RCG 1E12 390.0E-9
INITIAL VC1 5.4E6
!
!CSV IC1
!$IMRX_A
!
!CSV VC1
!$VMRX_A
!
! END OF BRANCH A!
!
! ================================================================
! E N D O F S I M U L A T I O N
! ================================================================
%(Ri1,W1,dB91,Tau2,ZPFL2,Ri2,W2,dB92,ZTRL,ZTRL,ZTRL,ZTRL,ZTRL,ZTRL,Ri2,W2,dB9
2,Tau2,ZPFL2,Ri1,W1,dB91,ZTRL,ZTRL,ZTRL,Ri2,W2,dB92,Tau2,ZPFL2,Ri1,W1,dB91,ZT
RL,ZTRL,ZTRL,Ri2,W2,dB92,Tau2,ZPFL2,Ri1,W1,dB91)
```

Appendix C: Sample Settings File

```
&&& Settings File (Partition Size, Parameter Ranges, Dependent
Parameters, Constraint Functions, Paths/Directories, Post-Processing,
Figures of Merit)

SCREAMERexe = ScreamerXPU.exe



W1: (.33, .44, .55)

W2: (.11,.22,.33)

Ri1: [.66,.76]  5

Ri2: [.66,.76]  5

ZPFL2:[.18,.26] 5

ZTRL:[.65,.73] 4



N = 9



&&  dependent variable functions must be entered in by dependency
tiers

INDUCTANCE = 2E-7*log(.8382/(1.4708-(W1)-.2)) IF do not use

Tau2 = (1.4708-(W1+W2)-.2)/(300000000/sqrt(80)) DP


RiS1 = (Ri1^N)/(.8382^(N-1)) DV

RiS2 = (Ri2^N)/(.8382^(N-1)) DV


dB91 = 2.8*(.8382-RiS1)/(.8382-Ri1) DP

dB92 = 2.8*(.8382-RiS2)/(.8382-Ri2) DP


Vsafe must be greater than VBP_IN_D max.  OC
FOM2 must be less than 2.  OC
&& W1 must be greater than W2. IC
```

```
SimDir = '/home/insignium/Documents10-21/Work/SCREAMER'

CatFol = 'LogFolder'

Name = 'TSZ08'


time points: Tstart when ILOAD exceeds 5E6, Tend when ILOAD falls
below 5E6, Timp when LVOLT reaches its maximum, Teff while VBP_IN_D is
above .63*(VBP_IN_D max)


MSPpV = 1.34

MSV = pi*((.8382)^2-(RiS1)^2+(.8382)^2-(RiS2)^2) IF output

Vsafe = 1.08E5*(.8382*(1-exp(-sqrt(80)*(ZPFL2)*9/60)))*|Teff|^(-.33)
OF output

t_imp Tstart to Timp UB 100E-9 output

Iimp ILOAD max LB 20.E6 output

Vimp LVOLT at Timp output

VBP_IN_D max  output

MSP = MSV*MSPpV IF output


FOM1 = 10^-14*Iimp/t_imp

FOM2 = (VBP_IN_D max)/Vsafe
```

Appendix D: SAE2.py Source Code

```
#!usr/bin/env python

import math, sys, os, string, csv, re

from subprocess import Popen, PIPE

print('SCREAMER AUTO EXPLORATION\n\t\tFrankin Orndorff-Plunket
forndor@sandia.gov\nThis script package tests a user defined parameter space
for SCREAMER models using .csv outputs. \n\n')
```

```
# Settings and Template files

Name = str(input('Name?\t'))

System = os.name

TOP = '' # TOP OF THE OS.WALK FUNCTION to find the folder that the files
reside in

FSC = '' # FileSystemConvention

if System == 'nt':

    TOP = r'C:\\'

    FSC = r'\\'

    print("Automatically configured for NTFS filesystem.\n")

elif System =='posix':

    TOP = '/home'

    FSC = '/'

    print("Automatically configured for POSIX filesystem.\n")

else:

    print('This package has only been configured for Unix, Linux and
Windows.')

ToLook = []

CTPrompt = ''

for item in os.walk(TOP):

    for item2 in item[2]:

        if item2.find(Name+'.asc') !=-1:

            ToLook.append(item)

            CTPrompt = CTPrompt +'#'+str(len(ToLook))+'
'+str(item[0])+'\n'

        else:

            pass

ChTo = int(input(CTPrompt+'\n\tWhich directory?\n(enter an integer)\t'))


SimDir = ToLook[ChTo-1][0]

os.chdir(SimDir)

print('Working directory is:\t' +str(os.getcwd()))
```

```
if os.path.exists(ToLook[ChTo-1][0]+FSC+Name+'.asc') is True:

    SETTINGS = open(Name + '.asc', 'r+')

    TEMPLATE = open(Name + '.dat', 'r+')

else:

    print('Settings files not found. Please check the name and if the files
are in the correct folder.')

    pass

#Template Processing -------------------------------


TEMPLATESTR = TEMPLATE.read()

TwoPartsTS = TEMPLATESTR.rsplit('%',1)

TSL = SETTINGS.readlines()


TSLC = False

while TSLC == False:

    for i, item in enumerate(TSL):

        SPACE = item.isspace()

        OT = (item.find('&&'),item.rfind('do not use'))

        if SPACE is True or len(item)==0 or OT!=(-1,-1):

            TSL.remove(item)

            break

        elif i== (len(TSL)-1):

            TSLC = True

        else:

            pass

TSLL = False

if System == 'nt':

    while TSLL != True:

        for i, item in enumerate(TSL):
# These variables were going to be used to

            LinusExec = re.seach('Linux_Loader|pyVersion',item)
# automatically run the script that SAE2 writes.
```

```
            if i < (len(TSL)-1) and LinusExec is not None:
# It's easier just to type in the commands at

                TSL.remove(item)
# shell prompt.

                break

            elif i<(len(TSL)-1) and LinusExec is None:

                pass

            elif i==(len(TSL)-1):

                TSLL = True

                break

            else:

                pass

    LL = ''

else:

    pass




for i, item in enumerate(TSL):

    TSL[i] = item.strip()

for i, item in enumerate(TSL):

    H =
re.findall('(exp|sqrt|log|log10|asin|acos|atan|cos|sin|tan|acosh|asinh|atanh|c
osh|sinh|tanh|eNB|pi)|(\^)|(\|(.+)\|)',TSL[i])

    if H != []:


        for Groupset in H:

            if H.count(Groupset)>1:

                exec('H.remove(Groupset)\n'*(H.count(Groupset)-1))

            else:

                pass


            if Groupset[0] != '' and Groupset[0] !="eNB":

                TSL[i] = TSL[i].replace(Groupset[0],'math.'+Groupset[0])
```

```
            else:
                pass

            if Groupset[0] != ''and Groupset[0]=='eNB':
                TSL[i] = TSL[i].replace(Groupset[0],'math.e')
            else:
                pass

            if Groupset[1]!='':
                TSL[i] = TSL[i].replace(Groupset[1],'**')
            else:
                pass

            if Groupset[2] !='' and Groupset[3]!= '':
                TSL[i] = TSL[i].replace(Groupset[2],'abs('+Groupset[3]+')')
            else:
                pass

TSS = ''
for X in tuple(TSL):
    TSS = TSS + X + '\n'

PyLSL = [] # Linux System necessary variables
PSL = [] # Partitioning
PRVL = [] # Paramter Ranges or Values
PDL = []  # Paths/Directories
OSL = []  # OS Parameters (SCREAMER exe name, linux loader, python version)
FL = []   # Fuctions
DPL = []  #  Dependent variable functions
DPVL = [] #  Dependent variable intermediates
CFL = []  # Constraint Variable Functions
```

```
OF = []  # Output Functions
IF = []  # Input functions
CL = [] # Constants
CIL = []  # Constraint Inequality
PPL = []   # Post-Processing
FML = []   # Figures of Merit
DVL = []   # Dependent Variables
AVL = []   # All Variables

if System == 'nt':
    del PyLSL
else:
    pass

# This section parses the settings file
# with regular expressions.
for item in TSL:
    #Initialisation
    ps = []
    pr = []
    dp = []
    pd = []
    oS = []
    cf = []
    ci = []
    cl = []
    pp = []
    fm = []

    #Partition Specs
    PS = '(.+):\s*\[\s*.+\s*,\s*.+\s*\]\s+(\d+)'
```

```
    ps = ()
    if re.search(PS,item) is not None:
        ps = re.search(PS,item).group(1),re.search(PS,item).group(2)
    else:
        pass
    # OS specifics
    OS = '(pyVersion|Linux_Loader|SCREAMERexe)\s*=\s*(.+)'
    if re.search(OS,item) is not None:
        oS.append((re.search(OS,item).group(1),re.search(OS,item).group(2)))
        if re.search(OS,item).group(1) == 'SCREAMERexe':
            exec('EXE =\''+re.search(OS,item).group(2)+'\'')
        elif re.search(OS,item).group(1)=='pyVersion':
            exec('PYV = \''+re.search(OS,item).group(2)+'\'')
        elif re.search(OS,item).group(1) == 'Linux_Loader':
            if 'LL' in sorted(locals()):
                pass
            else:
                exec('LL = '+re.search(OS,item).group(2))
        else:
            print('Error %s was not fully specified in the settings file.
'%re.searc(OS,item).group(1))
            exit()


    else:
        pass


    # Parameter Ranging and Value Definition
    PR = '(.+\s*:\s*\[\s*.+\s*,\s*.+\s*\]|.+:\s*\(\s*.+\s*\))'
    pr = re.findall(PR,item)


    # Dependent Parameters
    F = '.+\s*=\s*.+'
```

```
    f = re.findall(F,item)


# Constraint Inequalities

    CI = '.+\s?must be.+'

    ci = re.findall(CI,item)


 # Paths/Directories

    PD = 'SimDir\s*=\s*.+|CatFol\s*=\s*.+|Name\s*=\s*.+'

    pd = re.findall(PD,item)


    #Post Processing

    PP = '.+\s.+ to .+|.+ at .+|.+\s+when\s+exceeds\s+.+|.+\s+when\s+falls
below\s+.+|.+\s+when\s+.+\s+reaches\s+its\s+.+|.+\s+while\s+.+\s+is
above\s+its\s+.+|.+\s+while\s+.+\s+is
below\s+.+|.+\s+.*\s*max\s*.*|.+\s+.*\s*min\s*.*'

    pp = re.findall(PP,item)


    #Figure of Merit

    FM = 'FOM\d*\s*=\s*.+'

    fm = re.findall(FM,item)


    if ps !=():

        PSL.append(ps)

    else:

        pass


    if pr != []:

        for item in pr:

            try:

                item = item.replace(':','=')

            except:

                pass

            print(item,'\going into PRVL')
```

```
        PRVL.append(item)
 else:
     pass

 if f != [] and f != ps and f != pd and f != fm:  # Functions
     for item in f:
         if item.find('exe') ==-1:
             FL.append(item)
         else:
             pass
 else:
     pass
# Constraint Inequalities
 if ci != []:
     for itema in ci:
         CIL.append(itema)
 else:
     pass

 #Paths/Directories
 if pd !=[] :
     for item in pd:
         Repl = re.search('=\s*\'',item)
         DasRepl = Repl.group(0)
         item = item.replace(DasRepl,'= r\'')
         PDL.append(item)
 else:
     pass

 if fm !=[]:
     for item in fm:
```

```
            FML.append(item)
    else:
        pass


    if pp != [] and pp !=ps and pp != pr and pp != dp and pp!= ci and pp!=fm
and pp!=pd and pp!=f and item != '':
        for i,item in enumerate(pp):
            if item.find(',') != -1:
                pp[i] = item.split(',')
                for item2 in pp[i]:
                    if item2.rfind('time points:') != -1:
                        item2 = item2.replace('time points:','',1)
                    else:
                        pass
                    if item2 in PPL:
                        pass
                    else:
                        PPL.append(item2)
            else:
                PPL.append(item)
    else:
         pass



#  From here SAE2 translates the parsing from the
#  settings file into python strings to be written
#  the script file.

# Independent and Dependent Parameters Distinction
IP = re.findall('[^,\s\(\)]{2,}',TwoPartsTS[1])
IPL = []
print(IP,'\tIP')
```

```
for item in IP:

    for item2 in PRVL:

        if item not in IPL:

            IPL.append(item)

        else:

            pass

IPT = tuple(IPL)




#Constraint and Dependent Parameter function parsing

for item in FL:

    IsDPorCF = []

    for item2 in IPL:

        IsDPorCF.append(item.find(item2))

    if IsDPorCF != [-1, -1, -1, -1, -1, -1] and
re.match('.+\s*=\s*.+\s+DP\s*',item) is not None or
re.match('.+\s*=\s*.+\s+DV\s*',item)is not None:

        try:

            DPL.append(item.rstrip(re.search('(\s+DP\s*)',item).group(1)))

        except:

            DPVL.append(item.rstrip(re.search('(\s+DV\s*)',item).group(1)))

    elif re.match('\s*([\w]+)\s*=\s*[-]?\d+(\.\d+)?[eE]?[\-
]?(\d+(\.\d*)?)?\s*$',item) is not None:

        CL.append(item)

    else:

        CFL.append(item)




for i,item in enumerate(PRVL):

    PRVL[i] = item.replace(':','=')




# Partition Size Variable
```

```
String1 ="IPT = %s\n%s\n%s\n%s\n%s\nTSS =
%s\n"%(IPT,PDL[0],PDL[1],PDL[2],'TwoPartsTS ='
+str(TwoPartsTS),'\"\"\"'+TSS+'\"\"\"')

for item in PSL:

    String1 = 'PS'+item[0]+' = ' + item[1]+' \n' +String1

String1 = "\nimport math,sys,os,string, csv,re\nfrom subprocess import Popen,
PIPE\n"+ String1

Order = []


print(IPL,'\T IPL')

for item in IPL:

    Order.append(IPL.index(item))

print(Order,'\t Order')

OrderT = tuple(Order)

OPC = ''


#Auto-partitioning Section

String2 = ''

for item in PRVL:

    String2 = String2 + item +'\n'

print(PRVL,'\t PRVL')

# String 4 (and the last bit of String3): Translating Math in to Python Math

DPFunctions = []

DPVFunctions = []


for item in DPL:

    DPFunctions.append(item)


for item in DPVL:

    DPVFunctions.append(item)
```

```
for i,item in enumerate(DPFunctions):

    DPFunctions[i] = item.split('=',1)

    DVL.append(DPFunctions[i][0])

for i,item in enumerate(DPVFunctions):

    DPVFunctions[i] = item.split('=',1)

    DVL.append(DPVFunctions[i][0])

print(DPFunctions,'\tDPFunctions\n',DPVFunctions,'\tDPVFunctions\n',DVL,'\tDVL
\n')


ConstantsS = ''

for item in CL:

    ConstantsS = ConstantsS + item + '\n'

String3 =r"""

#Array Manipulation Functions and Constants

"""+ConstantsS+r"""

def CombSVar(nA,A):

    "Takes a list of lists A and the corresponding name list (item in A is a
vector) and combines the items while also keeping track of the variables'
order."

    C = []

    B = []

    NewVector = []

    OrderList = []

    Ag = ''

    for i, item in enumerate(A):

        Ag = Ag + str(nA[i]).strip('\[\'\]') +'='+ str(item) +'\n'

    for i, item in enumerate (nA):

        OrderList.append(nA[i])

        NewVector.append('curr' + str(nA[i]).strip('[\']'))

        C = B.append(('\t'*i+'for curr%s in %s:\n'%(str(nA[i]).strip('[]'),
str(nA[i]).strip('[]'))).replace('\'',''))

        D = ''.join(B) + str('\t'*len(nA)) +'NewVector =
%s\n'%(str(NewVector).replace('\'',''))
```

```
        E = D + '\n' + '\t'*len(nA) + 'NVA.append(NewVector)\n'
    ExecString =  '\nNVA = []\nInputArray = []\n'+Ag+ E
    OrderTuple = tuple(OrderList)
    return ExecString, OrderTuple


# Partition
supv= []
nsubv = []
ParamC = 0
VarDict = vars()
RangesDict = {}
VDC = VarDict.copy().items()
for k, v in VDC:
    if k in IPT and type(v)== type([]):
        param = k
        nsubv.append(param)
        ParamC = ParamC + 1
        RangesDict[k] = v
    elif k in IPT and type(v)==type(()) and k not in ('vIntv','supv','nsubv'):
        param = k
        nsubv.append(param)
        ParamC = ParamC + 1
        RangesDict[k] = v
    else:
        pass
CSP = ''


for eachName in nsubv:
```

```
    if type(VarDict[eachName])==type([]):

        vIntv = RangesDict[eachName]

        r = vIntv[1]-vIntv[0]

        CSP = 'p'+eachName + ' = ' +str(r)+'/(PS'+eachName +'-1)\n'
+eachName+'= []\n'+CSP +"\nsubv%s = []\nfor eachNum in range(PS%s):\n
subv%s.append(%f+p%s*eachNum)\nsupv.append(subv%s)"%(eachName,eachName,eachNam
e,vIntv[0],eachName,eachName)



    elif type(VarDict[eachName])==type(()):

        CSP = eachName +' = '+ str(tuple(VarDict[eachName])) +'\n'+CSP+
'\nsupv.append('+str(list(VarDict[eachName]))+')\n'

    else:

        pass

CSP = 'supv = []\n'+CSP

exec(compile(CSP,'CombinerPrep','exec'))

exec(compile(CombSVar(nsubv,supv)[0],'Combiner','exec'))

iAVL = list(CombSVar(nsubv,supv)[1])

for item in """+str(DPFunctions)+""":

    iAVL.append(item[0].strip(' '))


iAVT = tuple(iAVL)

TPT = tuple(TwoPartsTS[1].strip('()').split(','))

IPS = '('

for item in TPT:

    for item2 in iAVT:

        if re.search(item2,item) is not None or re.search(item,item2) is not
None:

            IPS = IPS+'item[%i]'%iAVT.index(item2)+','

        else:

            pass


IPS = IPS.rstrip(',')+')'

print(IPS)

TPT = tuple(TwoPartsTS[1].strip('()').split(','))
```

```
"""

# Variable Lists to keep track
#AVL = IPL+DVL

String4 = "for item in NVA:\n    if item not in InputArray:"
String4a = ''
String4b = '\n        dongxi = item'
String4c = '\n        InputArray.append(dongxi)'
String4d = '\n    else:\n        pass'

for item in DPFunctions:
    for item2 in IPT:
        if re.search(item2,item[1]) is not None:
            item[1] =
item[1].replace(re.search(item2,item[1]).group(),'item[iAVT.index(\''+item2+'\
')]')
        else:
            pass
    String4b = String4b + '+[%s]'%str(item[1])

for item in DPVFunctions:
    for item2 in IPT:
        if re.search(item2,item[1]) is not None:
            item[1] =
item[1].replace(re.search(item2,item[1]).group(),'item[iAVT.index(\''+item2+'\
')]')
        else:
            pass
    String4a =  String4a + '\n        '+item[0]+'='+item[1]
String4 = String4+String4a+String4b+String4c+String4d
print('\n\n',String4,'\nString4')
```

```
# String 5 Input File Creation and Passing to SCREAMER
String5a =  r"""
if os.path.exists(SimDir+'%s'+CatFol) is True:
    pass
else:
    os.mkdir(SimDir+'%s'+CatFol)
os.chdir(SimDir+'%s'+CatFol)
Count = 0
if os.path.exists(SimDir+'%s'+CatFol+'%s'+Name+'_Catalog.txt') is True:
    Test = open(Name+'_Catalog.txt','r').read()
    Count = Test.count(Name)
    open(Name+'_Catalog.txt','r').close()
else:
    pass
os.chdir(SimDir)



for j,item in enumerate(InputArray):
    EnumInt = str(j)"""%(FSC,FSC,FSC,FSC,FSC)
String5b = ''
for item in CFL:
    for item2 in IPT:
        if item.find(item2) !=-1:
             String5b = String5b + '\n
round(item[iAVT.index(\'%s\')],4)'%str(item2)
        else:
             pass
    if item.find('OF') !=-1:
        OF.append(item.strip(' '))
    elif item.find('IF'):
        IF.append(item.strip(' '))
```

```
    else:

        print(str(item)+'\n\tThis function was not categorized as either one
involving input or output variables' )

del CFL

print(OF,IF)

String5c = r"""

    MakeInputFile = open(SimDir + '%s'+ Name+'Cyc'+str(Count)+'No'+EnumInt
+'.dat', 'w+')"""%FSC

String5d = r"""

    TPTS = TwoPartsTS[0]

    exec('Sub = ' +IPS)

    MakeInputFile.write(TPTS%Sub)

    MakeInputFile.close()

"""

String5e = ''

if System == 'posix':

    String5e = r"""

FinalRun = len(InputArray)

print(FinalRun)

IAF= open(Name+'_Test_Points.txt','a+')

IAF.write(str(iAVT)+'\n')

for item in InputArray:

    IAF.write(str(item)+'\t'+str(InputArray.index(item))+'\n')

IAF.close()

Sim = '%s'

for j in range(0,FinalRun):

    p = Popen('%s '+Sim, shell=True, stdin=PIPE, stdout=PIPE, stderr=PIPE)

    sss= Name+'Cyc'+str(Count)+'No'+ str(j) + '.dat\n\n'

    out, err = p.communicate(sss.encode())

    print('\nFinished Run No.:\t'+str(j))

    """%(EXE,LL)

elif SYSTEM =='nt':
```

```
    String5e = r"""

FinalRun = len(InputArray)

IAF= open('InputArray.txt','a+')e to GNU Emacs, one component of the GNU/Linux
operating system.

To quit a partially entered command, type Control-g.


for item in InputArray:

    IAF.write(str(item)+'\t'+str(InputArray.index(item))+'\n')

IAF.close()

Sim = '%s'

for j in range(0,FinalRun):

    p = Popen(Sim, shell=True, stdin=PIPE, stdout=PIPE, stderr=PIPE)

    sss= Name+'Cyc'+str(Count)+'No'+str(j) +'.dat\r\n\r\n'

    out, err = p.communicate(sss.encode())

    print('\nFinished Run No.:\t'+str(j))

"""%(EXE)

else:

    pass

String5 = String5a+String5b+String5c+String5d+String5e


# String 6 CSV Reading Method

String6 = """

OPC = \'\'

if os.path.exists(SimDir+\'%s\'+Name+\'Cyc\'+str(Count)+\'No\'+str(0)
+\'.dat.csv\') is True:

    OPC = \'.dat.csv\'

elif os.path.exists(SimDir+\'%s\'+Name+\'Cyc\'+str(Count)+\'No\'+str(0)+
\'.csv\') is True:

    OPC = \'.csv\'

else:

    pass

csvrd = {}

csvrs = ''
```

```
for k in range(0,FinalRun):

    K = str(k)

    SOP = open(Name +\'Cyc\' +str(Count)+\'No\' +str(k)+OPC ,'r+')

    SOP1 = next(SOP).split(',')

    CSVR  = csv.reader(SOP)

    if K =='0':

        for i,item in enumerate(SOP1):

            SOP1[i] = item.strip()

        for i,item in enumerate(SOP1):  # Finding all Post-Processing variable
names and

            if re.search(item,TSS) is not None and len(item)!= 0:  # reading
the corresponding lists

                csvrd[SOP1[i]] = SOP1.index(item)

            else:

                pass

        for k,v in csvrd.copy().items():

            csvrs = csvrs + 'header =
next(CSVR)\\nheader['+str(v)+'].strip()\\n' + k + '= [float(line['+str(v)+'])
for line in CSVR]\\n\\nSOP.seek(0)\\n\\n'

    else:

        pass

    csvrs = csvrs.rstrip('\\n\\nSOP.seek(0)\\n\\n')

    exec(csvrs)

"""%(FSC,FSC)

String7 = ''

# String 7: Post-Processing and FOMs

# String7 First Part Zeroing & Defining minimals and maximals

for item in PPL:

    if
re.search('^([\w]+){1}\s*(max|Max|min|Min|maximum|Maximum|minimum|Minimum)',it
em) is not None:

        pass

    else:

        String7 = String7 + '    ' + re.search('([\w]+){1}',item).group(1) + '
= 0.0\n'
```

```
for item in FML:

    String7 = String7 + '    ' + re.search('([\w]+){1}',item).group(1) + ' =
0.0\n'

for item in OF:

    String7 = String7 + '    ' + re.search('([\w]+){1}',item).group(1) + ' =
0.0\n'


AS = ''

for item in PPL:

    A = re.search('([\w]+)\s+(reaches|is
at)\s+(its)?\s+(max|min)|([\w]+)\s+(min|max)',item)

    A1 = '    '

    if A is not None:

        if A.group(1) is not None and A.group(2) is not None and A.group(3) is
not None and A.group(4) is not None:

            A1 = A1 + 'Max'+ A.group(1)+ ' = ' + A.group(4) + '(' + A.group(1)
+ ')\n'

        elif A.group(5) is not None and A.group(6) is not None :

            A1 = A1 + 'Max' + A.group(5) + ' = ' + A.group(6) + '(' +
A.group(5) + ')\n'

        else:

            pass

        if AS.find(A1) == -1:

            AS = AS + A1

        else:

            pass


String7 = String7 + AS


AS = ''

for item in PPL:

    B = re.search('([\w]+)\s+(when|while)\s+([\w]+)\s+(exceeds|reaches|falls
below|falls
beneath|over|under|above|below|beneath|is\s+above|is\s+below|is\sbeneath|is\s+
over|is\s+under)\s+(.+)?',item)
```

```
    if B is not None:


        if B.group(2) == 'when':

            if B.group(4) == 'reaches':

                if B.group(5).find('max') != -1:

                    String7 = String7 + '    ' + B.group(1)+ ' = time[' +
B.group(3)+'.index(Max'+B.group(3)+')]\n'

                elif B.group(5).find('min') != -1:

                    String7 = String7 + '    ' + B.group(1)+ ' = time[' +
B.group(3)+'.index(Min' +B.group(3)+')]\n'

                else:

                    pass


            elif B.group(4) == 'exceeds':



                if re.search('max',B.group(5)) is not None:

                    String7 = String7 + '    for i, item in
enumerate('+B.group(3)+'):\n        if '+B.group(3)+'[i] >= ' +
B.group(5).replace(re.search('(([\w]+){1}\s*(max|Max|maximum|Maximum))',item),
'max('+re.search('(([\w]+){1}\s*(max|Max|maximum|Maximum))',item).group(1)+')'
) + ' and ' + B.group(3) + '[i-1] < '
+B.group(5).replace(re.search('(([\w]+){1}\s*(max|Max|maximum|Maximum))',item)
,'Max'+re.search('(([\w]+){1}\s*(max|Max|maximum|Maximum))',item).group(1))+':
\n' + B.group(1) +' = time[+' +B.group(3)+'.index(item)]\n        else:\n
pass\n'

                elif re.search('min',B.group(5)) is not None:

                    String7 = String7 + '    for i, item in
enumerate('+B.group(3)+'):\n        if '+B.group(3)+'[i] >= ' +
B.group(5).replace(re.search('(([\w]+){1}\s*(min|Min|minimum|Minumum))',item),
'min('+re.search('(([\w]+){1}\s*(min|Min|minimum|Minumum))',item).group(1)+')'
) + ' and ' + B.group(3) + '[i-1] < '
+B.group(5).replace(re.search('(([\w]+){1}\s*(min|Min|minimum|Minumum))',item)
,'Min'+re.search('(([\w]+){1}\s*(min|Min|minimum|Minumum))',item).group(1))+':
\n' + B.group(1) + ' = time[+' + B.group(3)+'.index(item)]\n        else:\n
pass\n'

                elif re.search('max',B.group(5)) is None and
re.search('min',B.group(5)) is None:

                    String7 = String7 + '    for i, item in
enumerate('+B.group(3)+'):\n        if '+B.group(3)+'[i] >= ' + B.group(5) + '
and ' + B.group(3) + '[i-1] < ' + B.group(5) + ':\n            '+B.group(1)+'
= time[' +B.group(3) + '.index(item)]\n        else:\n            pass\n'

                else:
```

```
                    pass


            elif B.group(4) == 'falls below' or B.group(4) == 'falls beneath':


                if re.search('max',B.group(5)) is not None:

                    String7 = String7 + '    for i, item in
enumerate('+B.group(3)+'):\n        if '+B.group(3)+'[i] <= ' +
B.group(5).replace(re.search('(([\w]+){1}\s*(max|Max|maximum|Maximum))',item),
'max('+re.search('(([\w]+){1}\s*(max|Max|maximum|Maximum))',item).group(1)+')'
) + ' and ' + B.group(3) + '[i-1] > '
+B.group(5).replace(re.search('(([\w]+){1}\s*(max|Max|maximum|Maximum))',item)
,'Max'+re.search('(([\w]+){1}\s*(max|Max|maximum|Maximum))',item).group(1))+':
\n' + 'T2 = time[' +B.group(3)+'.index(item)]\n        else:\n
pass\n'

                elif re.search('min',B.group(5)) is not None:

                    String7 = String7 + '    for i, item in
enumerate('+B.group(3)+'):\n        if '+B.group(3)+'[i] <= ' +
B.group(5).replace(re.search('(([\w]+){1}\s*(min|Min|minimum|Minumum))',item),
'min('+re.search('(([\w]+){1}\s*(min|Min|minimum|Minumum))',item).group(1)+')'
) + ' and ' + B.group(3) + '[i-1] > '
+B.group(5).replace(re.search('(([\w]+){1}\s*(min|Min|minimum|Minumum))',item)
,'Min'+re.search('(([\w]+){1}\s*(min|Min|minimum|Minumum))',item).group(1))+':
\n' + 'T1 = time[' + B.group(3)+'.index(item)]\n        else:\n
pass\n'

                elif re.search('max',B.group(5)) is None and
re.search('min',B.group(5)) is None:

                    String7 = String7 + '    for i, item in
enumerate('+B.group(3)+'):\n        if '+B.group(3)+'[i] <= ' + B.group(5) + '
and ' + B.group(3) + '[i-1] > ' + B.group(5) + ':\n            '+B.group(1)+'
= time[' +B.group(3) + '.index(item)]\n        else:\n            pass\n'

                else:

                    pass

            else:

                pass

        if B.group(2) == 'while':

            if B.group(4) == 'exceeds' or B.group(4) == 'over' or B.group(4)
== 'is over' or B.group(4).find('above' ) != -1:

                if B.groups() is not None:

                    IN = re.search('([\-]?[\d]*[\.]?[\d]+[eE]?[\-
]?[\d]*[\.]?[\d]+)?([+\-
*/])?[\(]?([\w]+)[\)]?\s+(max|Max|maximum|Maximum)',B.group(5))

                    if IN is not None:

                       # print(IN.groups())
```

```
                        # print(item,B.groups(),IN.groups())

                         String7 = String7 +'\n    \n    qa = []\n    for n,
item in enumerate(' + B.group(3)+ '):\n        q = int('+B.group(3)+'[n]/(
'+IN.group(1)+IN.group(2)+'Max'+IN.group(3) + '))\n'

                    else:

                        pass



                    String7 = String7 + "        if q > 0:\n
qa.append(n)\n        else:\n            pass\n    "+B.group(1)+"1 =
time[qa[0]]\n    "+B.group(1)+"2 = time[qa[len(qa)-1]]\n    "+B.group(1)+" =
abs("+B.group(1)+'2 - '+ B.group(1)+'1)\n\n'



                else:

                    pass




# iAVL for post processing ordering considerations

String7 = String7 + r"""    for n,NAME in enumerate(iAVL):

        exec( NAME +' = InputArray['+K+']['+str(n)+']')

"""



for item in OF:

    if re.search('\A\s*([\w]+)',item) is not None and
re.search('(\s*output\s*)',item) is not None:

        String7 = String7 + '\n    try:\n        '
+item.replace(re.search('(\s*output\s*)',item).group(1),'').replace('OF','') +
'\n    except:\n        print(\'Unknown error in .txt settings file.\tPlease
check that file (%s)\'+K )\n        pass\n'%item

    else:

        pass

for item in IF:

    if re.search('\A\s*([\w]+)',item) is not None and
re.search('(\s*output\s*)',item) is not None:

        String7 = String7 + '\n    try:\n        '
+item.replace(re.search('(\s*output\s*)',item).group(1),'').replace('IF','') +
'\n    except:\n        print(\'Unknown error in .txt settings file.\tPlease
check that file (%s)\'+K)\n        pass\n'%item

    else:
```

```
        pass


for item in PPL:

    D  = re.search('([\w]+)\s+([\w]+)\s+at\s+([\w]+)',item)

    if D is not None:

        String7 = String7+'\n    '+D.group(1)+' =
'+D.group(2)+'[time.index('+D.group(3)+')]\n'

    else:

        pass

for item in PPL:

    Da =
re.search('([\w]+)\s+([\w]+)\s+to\s+([\w]+)\s+(UB|LB)\s+(\d*.\d+[eE]?[-
]?[\d]*)',item)

    if Da is not None:

        String7 = String7 + '    ' + Da.group(1)+ ' = ' + Da.group(3) + ' - '
+Da.group(2)+'\n'

    else:

        pass




    T = re.search('([\w]+)\s+([\w]+)',item)

    for itemum in FML:

        if itemum.find(T.group(1)) !=-1:

             if T is not None:

                if item.find('max') != -1 and T.group(2)!='max':

                    String7 = String7 + '    ' + T.group(1)+ ' =
Max'+T.group(2) + '\n'

                elif item.find('min') != -1 and T.group(2)!='min':

                    String7 = String7 + '    ' + T.group(1)+ ' =
Min'+T.group(2) + '\n'

                else:

                    pass

             else:

                 pass
```

```
        else:

            pass




for i,item in enumerate(FML):

    Addendum = ''

    if re.search('([\w]+)\s*(max|min)',item) is not None:

        Addendum ='    try:\n         '+
item.replace(re.search('([\w]+)\s*(max|min)',item).group(),re.search('([\w]+)\
s*(max|min)',item).group(2).capitalize()+re.search('([\w]+)\s*(max|min)',item)
.group(1))+'\n    except:\n        print(\'Unknown error.  Please check the
figure of merit functions in the settings file.\')\n'

    else:

        Addendum ='    try:\n        '+item+'\n    except:\n
print(\'Unknown error.  Please check the figure of merit functions in the
settings file.\')\n'

    String7 = String7 + Addendum

IVS  = '    ImplosionVector = []\n    try:\n        ImplosionVector = ['

for item in TSL:

    if re.search('\s+output\s*',item) is not None and  re.search('FOM',item)
is None:

        if re.search('\A([\w]+)\s+(min|max)?',item).group(2)is not None:

            IVS = IVS +
re.search('\A([\w]+)\s+(min|max)',item).group(2).capitalize()+re.search('\A([\
w]+)\s+(min|max)',item).group(1) +', '

        elif re.search('\A([\w]+)\s+(min|max)?',item).group(2) is None:

            IVS = IVS + re.search('\A([\w]+)\s+(min|max)?',item).group(1)+', '

        else:

            pass

    else:

        pass

for item in FML:

        IVS = IVS + re.search('^\s*(FOM\d*)',item).group()+ ', '

IVS = IVS.rstrip(', ')+']'
```

```
String7 = String7 + IVS + '\n    except:\n        print(\'Unknown error.\')\n
pass\n'

#

FOMFC = 0

for item in FML:

    if re.search('FOM',item) is not None:

        FOMFC = FOMFC+1

    else:

        pass

String8 = """

    os.chdir(CatFol)

    W = open(Name+'_Catalog.txt','a+')

    if int(K)==0:

        W.write(Name+ '\\tLog File:  Test No. \'+str(Count)+'\\Order of
Outputs:  %s'+ 'Order of Variables: '+str(iAVT)  + '\\n\\n')

    else:

        pass

    W.write(str(ImplosionVector)+' Run No.: ' +K+'\\n'+'Configuration:
'+str(InputArray[int(K)])+ '\\n')


    """%re.search('(\[.+\])',IVS).group(1)

String9A = ''

# Strings 9: Constraints Satisfied?


for ren in CIL:

    OC = re.search('([\w]+)\s+must
be\s+(less\s+than|greater\s+than|lower\s+than|higher\s+than)\s+([\w]+)\s*(max|
min)?\.',ren)

    if re.search("(OC|IC)",ren) is not None:


        if OC.group(2).find('less') !=-1 or OC.group(2).find('lower')!=-1:


            if OC.group(4) is not None:
```

```
                String9A = String9A + '\n    if '+  OC.group(1) + ' < ' +
OC.group(4).capitalize() + OC.group(3)+':\n        W.write(\''+OC.group(1)+':
constraint is satisfied  \')\n    else:\n        W.write(\''+OC.group(1)+':
constraint is NOT satisfied  \')'

            else:

                String9A = String9A + '\n    if '+ OC.group(1) + ' < '
+OC.group(3)+':\n        W.write(\''+OC.group(1)+': constraint is satisfied
\')\n    else:\n        W.write(\''+OC.group(1)+': constraint is NOT satisfied
\')'

        elif OC.group(2).find('greater')!=-1 or OC.group(2).find('higher')!=-
1:



            if OC.group(4) is not None:

                String9A = String9A + '\n    if ' + OC.group(1) + ' > '+
OC.group(4).capitalize() + OC.group(3)+':\n        W.write(\''+OC.group(1)+':
constraint is satisfied  \')\n    else:\n        W.write(\''+OC.group(1)+':
constraint is NOT satisfied  \')'

            else:

                String9A = String9A + '\n    if '+ OC.group(1) + ' < '
+OC.group(3)+':\n        W.write(\''+OC.group(1)+': constraint is satisfied
\')\n    else:\n        W.write(\''+OC.group(1)+': constraint is NOT satisfied
\')'



String9 = String9A + '\n    W.write(\'\\n\\n\')\n    W.close()\n
os.chdir(SimDir)'

pySTRING = String1+String2+String3+String4+String5+String6 + String7 + String8
+ String9



Runtime = open(Name+'script.py','w+')

Runtime.write(pySTRING)  # writing the script

Runtime.close()

TEMPLATE.close()

SETTINGS.close()



exit()
```

Distribution

5 Franklin Orndorff-Plunkett
Physics Department 4-304
Massachusetts Institute of Technology
500 Memorial Drive
Cambridge, MA 02139-4307

2 E. L. Neau
L-Cubed
919 Sandia Rd NW
Albuquerque, NM 87107

4	MS0125	J. Pace VanDevender	12100
2	MS1152	Michelle Caldwell	1653
2	MS1152	Mark L. Kiefer	1652
1	MS1181	Larry X Schneider	1650
1	MS1181	Dillon H. McDaniel	1602
1	MS1186	Mark Herrmann	1680
1	MS1190	M. Keith Matzen	1600
1	MS1191	John Porter	1670
1	MS1193	Mike Cuneo	1683
1	MS1193	Daniel B. Sinars	1683
1	MS1194	Kenneth Struve	1679
1	MS1194	Mark E. Savage	1671
1	MS1196	William Stygar	1671
1	MS0899	Technical Library	9536 (electronic copy)