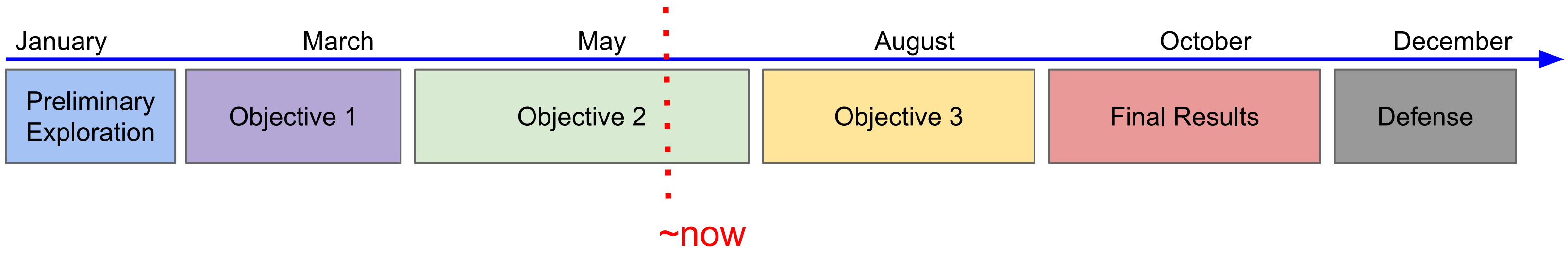
Project Update / Synopsis

Tom Looby
Nuclear Engineering Grad Student @
University of Tennessee-Knoxville
05/21/2018

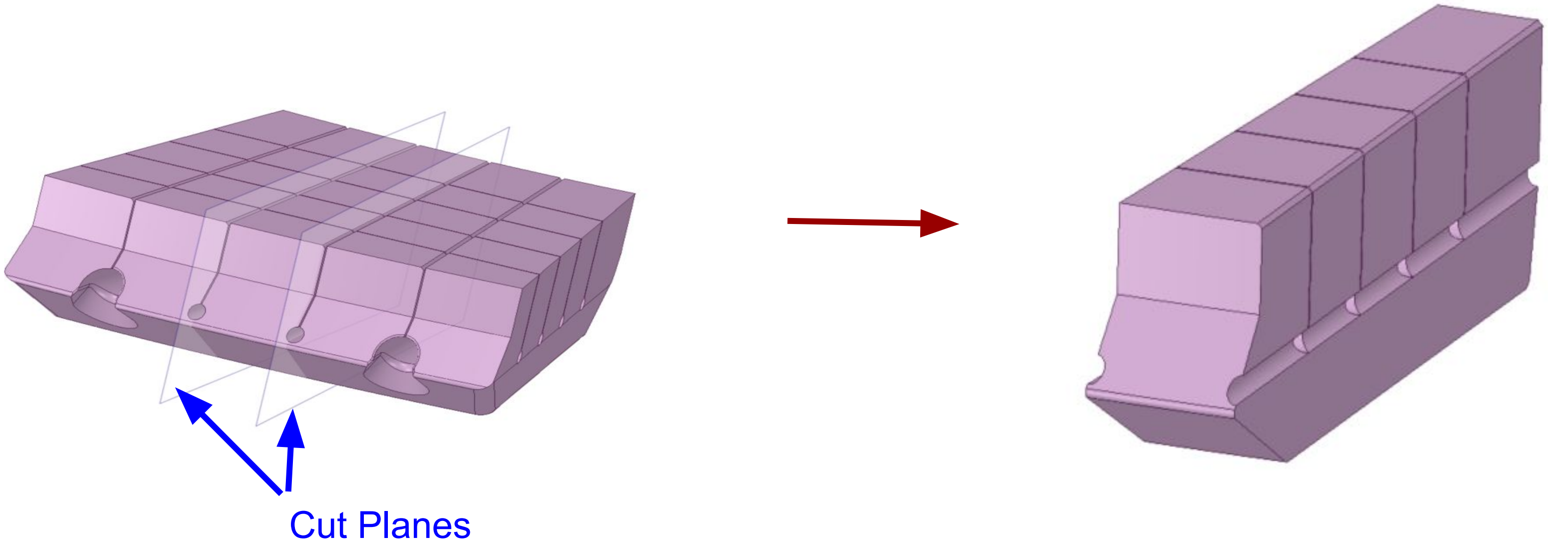
Project Objective 1: Tile Response

1. Simulate the response of NSTX-U graphite PFCs to spatially and time varying heat fluxes.
2. Demonstrate how unknown heat flux model parameters can be derived with various sampling mechanisms within a given parameter space.
3. Demonstrate (2) but now add demonstrated uncertainties to measurement and model support parameters.



Project Objective 1: Tile Response Simulation

1. Simulate the response of NSTX-U graphite PFCs to spatially and time varying heat fluxes.

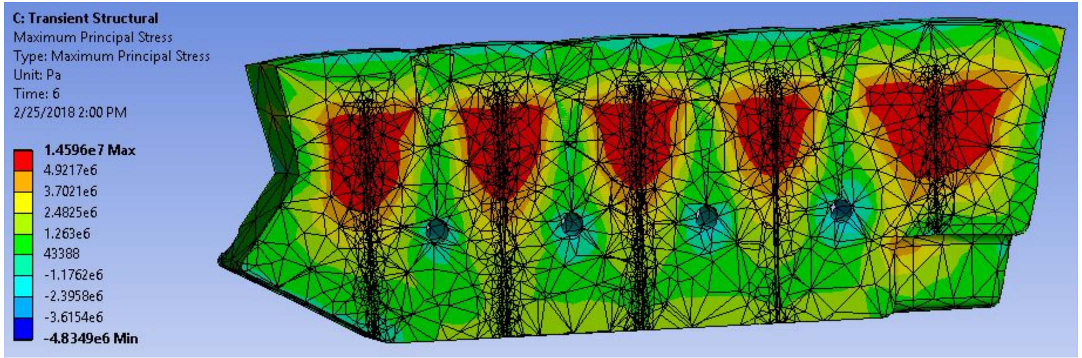


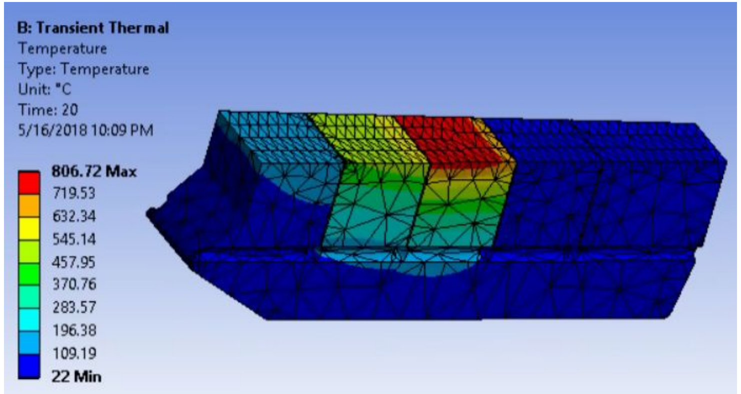
*Note: Jan 2018 version of IBHD

Project Objective 1: Tile Response Simulation

1. Simulate the response of NSTX-U graphite PFCs to spatially and time varying heat fluxes.

Random ANSYS Demonstrations / Examples:



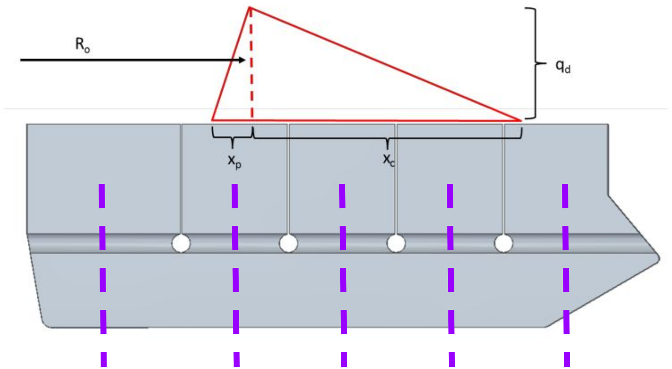


(~7.75 MW/m^2 limit discovered for 5s "flat" shot)

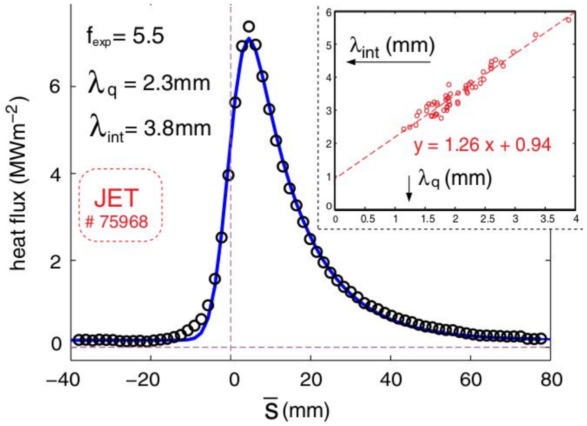
Project Objective 1: Tile Response Simulation

1. Simulate the response of NSTX-U graphite PFCs to spatially and time varying heat fluxes.

(very) Simplified Eich Model:



Eich Model:



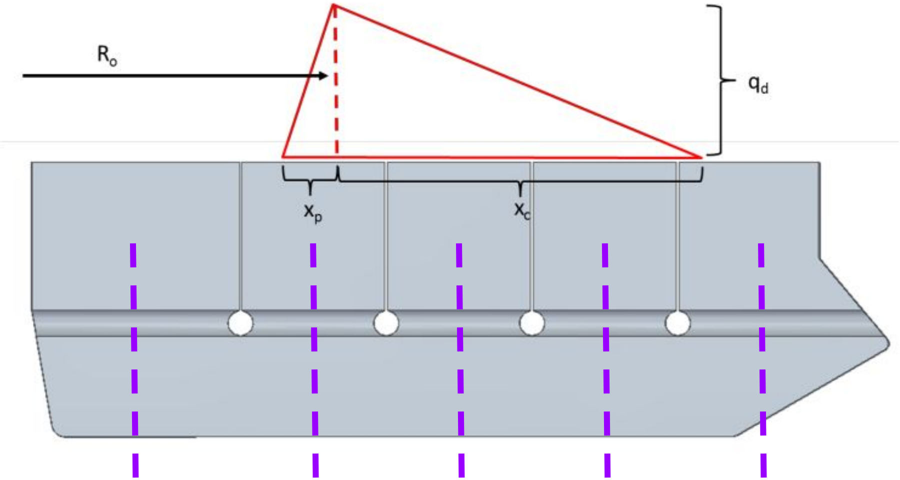
T. Eich *et al.*, "Inter-ELM Power Decay Length for JET and ASDEX Upgrade: Measurement and Comparison with Heuristic Drift-Based Model," *Physical Review Letters*, vol. 107, no. 21, Nov. 2011.

Project Objectives

1. Simulate the response of NSTX-U graphite PFCs to spatially and time varying heat fluxes.
2. Demonstrate how unknown heat flux model parameters can be derived with various sampling mechanisms within a given parameter space.
3. Demonstrate (2) but now add demonstrated uncertainties to measurement and model support parameters.

Project Objective 2: Extract "Eich Parameters"

2. Demonstrate how unknown heat flux model parameters can be derived with various sampling mechanisms within a given parameter space.



(very) Simplified Eich Model:

$$x_P = S f_x$$
$$x_C = \lambda_q f_x$$
$$P_{heat} = \int_{R_o - x_p}^{R_O + x_C} q(R) 2\pi R dR$$

$$\frac{S}{\lambda_q} = C_1$$
$$\lambda_q[mm] = C_2 P_{heat}^{C_3} B_p^{C_4}$$

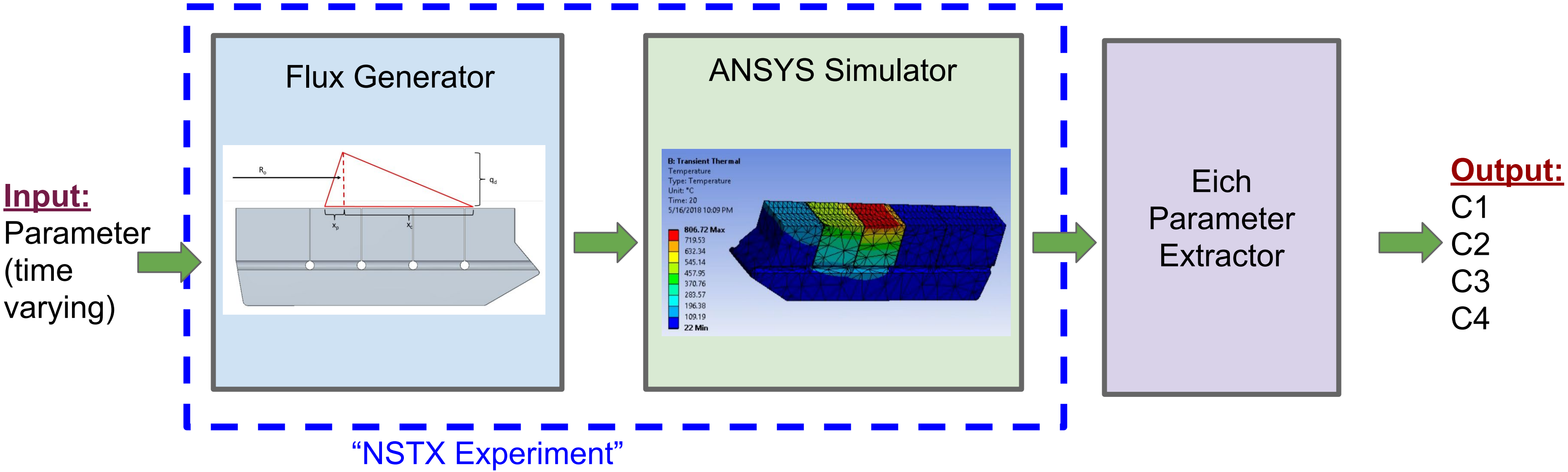
$$0.1 < C_1 < 0.3$$
$$1.0 < C_2 < 2.5$$
$$-0.1 < C_3 < 0.25$$
$$-1.2 < C_4 < -0.5$$

$$0.2 < B_p[T] < 0.6$$
$$0.5 < P_{heat}[MW] < 4.9$$
$$4 < f_x < 30$$
$$46.0 < R_o[cm] < 57.5$$
$$1 < \Delta t\ [sec] < 5$$

T. Eich *et al.*, "Inter-ELM Power Decay Length for JET and ASDEX Upgrade: Measurement and Comparison with Heuristic Drift-Based Model," *Physical Review Letters*, vol. 107, no. 21, Nov. 2011.

Project Objective 2: Extract "Eich Parameters"

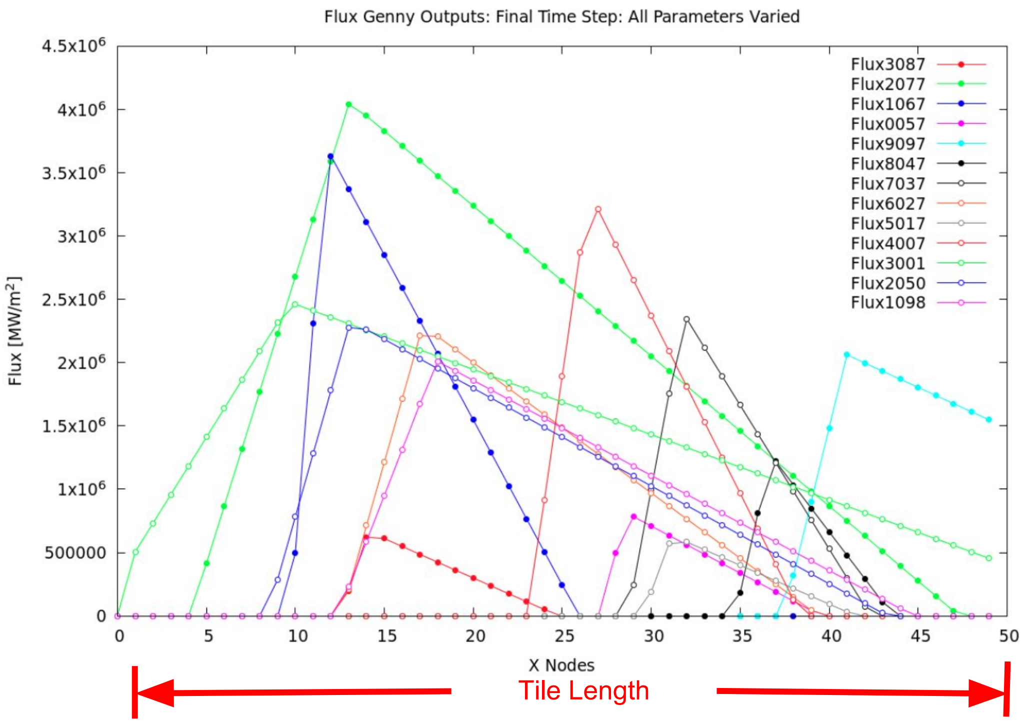
2. Demonstrate how unknown heat flux model parameters can be derived with various sampling mechanisms within a given parameter space.



Project Objective 2: Heat Flux Generator

- Monte Carlo Style
- Pulls random deviates for each model / machine parameter (ie: Bp, C1, etc.)
- Samples entire allowable operational domain for each parameter
- Can produce arbitrary number of flux profiles
- Currently producing ~10k profiles per case

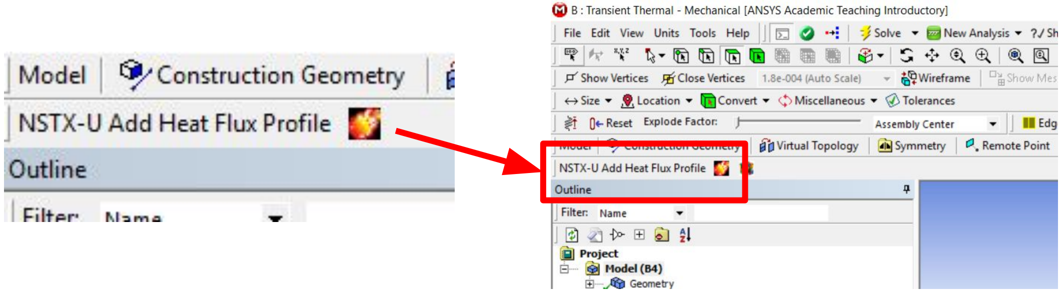
Example: Several randomly selected profiles for

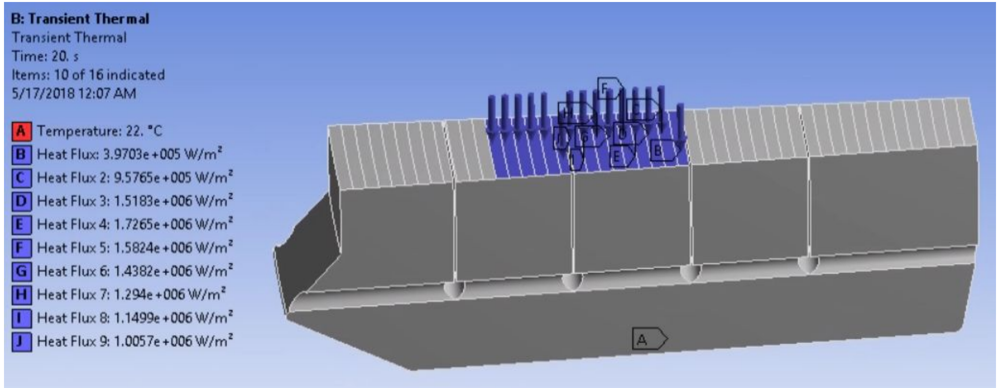


Project Objective 2: ANSYS Simulator

- Created autonomous ACT script to run in "batch" mode
- Applies heat flux to tile and solves heat diffusion PDE (to TCs)
- Not quite as fast as direct APDL script, but possible to do more (access to python modules) using API

Example: NSTX ANSYS Toolbar and Fluxes





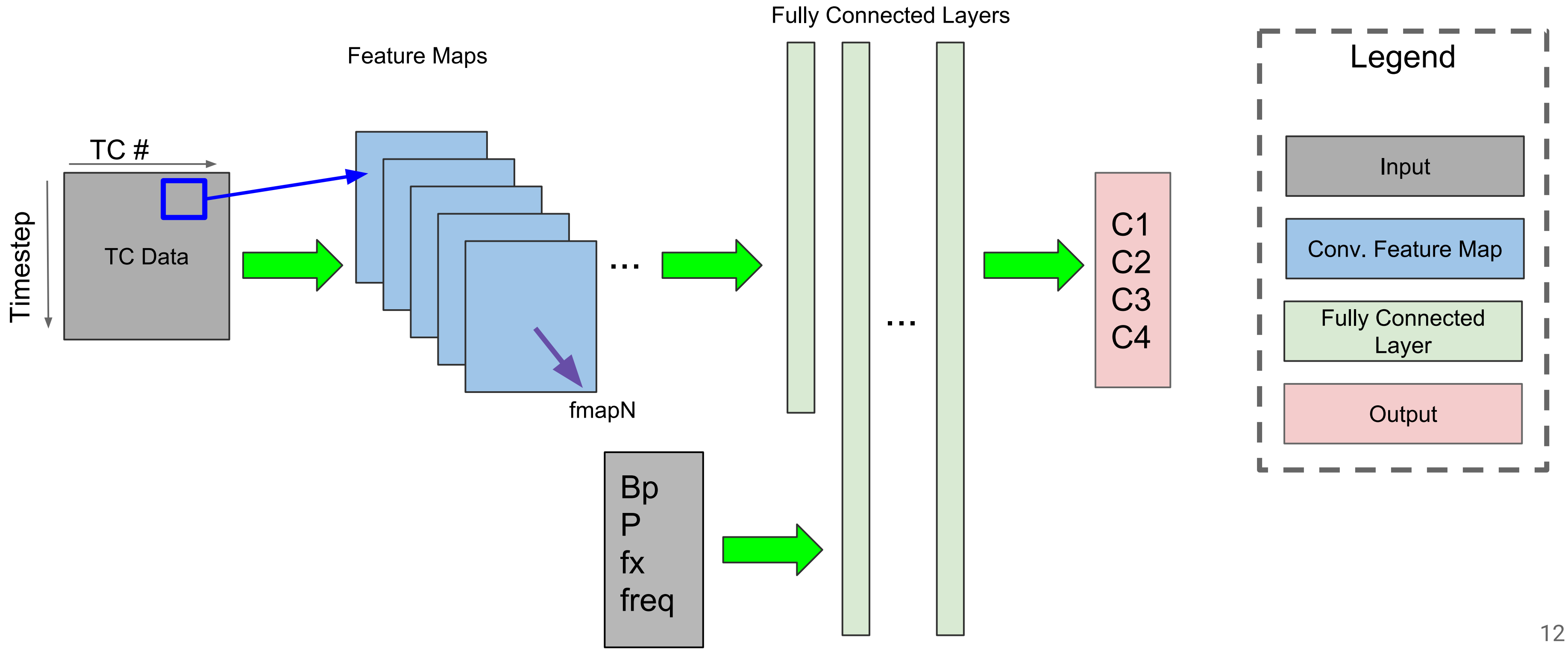
Project Objective 2: Extraction Options

- Solve heat diffusion equation (analytically or numerically) to derive time varying heat flux from temperature.
- Utilize neural networks to derive transfer function between TC profile and Eich parameters (C1, C2…)

Performance Variables		Heat Diffusion EQ		Machine Learning	
	Weight	Score	S*W	Score	S*W
Requires Minimal Assumptions	15.00%	50.00	7.5	90.00	13.5
Requires Small / Sparse Dataset	10.00%	80.00	8	20.00	2
Provides Intuitive Insight	15.00%	90.00	13.5	40.00	6
CPU Time	10.00%	70.00	7	40.00	4
Can Be Expanded to More Complex Problems	15.00%	30.00	4.5	90.00	13.5
Can Be Expanded to other Scientific Domains	10.00%	40.00	4	80.00	8
Potential to be Utilized in Real Time Systems (<1ms)	15.00%	60.00	9	95.00	14.25
Potential for Machine Spec Optimization	10.00%	60.00	6	90.00	9
Total			59.5		70.25

Note: These performance variables are subjective to my own research interests. Undoubtedly there are other important considerations. I am open to feedback!

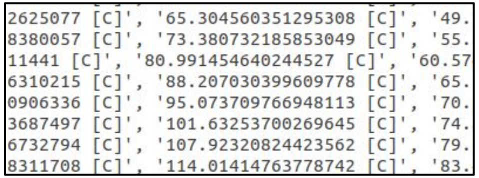
Project Objective 2: CNN Example



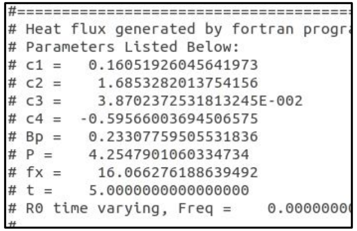
Project Objective 2: Challenges

- Steep learning curve
- So many hyperparameters...
- No well-defined procedure for NN selection
- Most NN API tutorials are for classification, not hard sciences with continuous outputs
- Data Selection
- Data Preprocessing!!!
- That being said, TensorFlow (from google) is a relatively easy API to use

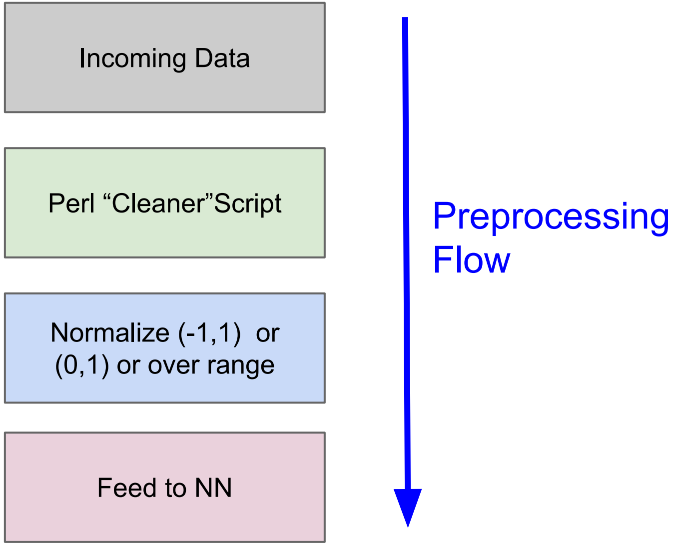
Incoming data looks like this:



```
2625077 [C]', '65.304560351295308 [C]', '49.
8380057 [C]', '73.380732185853049 [C]', '55.
11441 [C]', '80.991454640244527 [C]', '60.57
6310215 [C]', '88.207030399609778 [C]', '65.
0906336 [C]', '95.073709766948113 [C]', '70.
3687497 [C]', '101.63253700269645 [C]', '74.
6732794 [C]', '107.92320824423562 [C]', '79.
8311708 [C]', '114.01414763778742 [C]', '83.
```

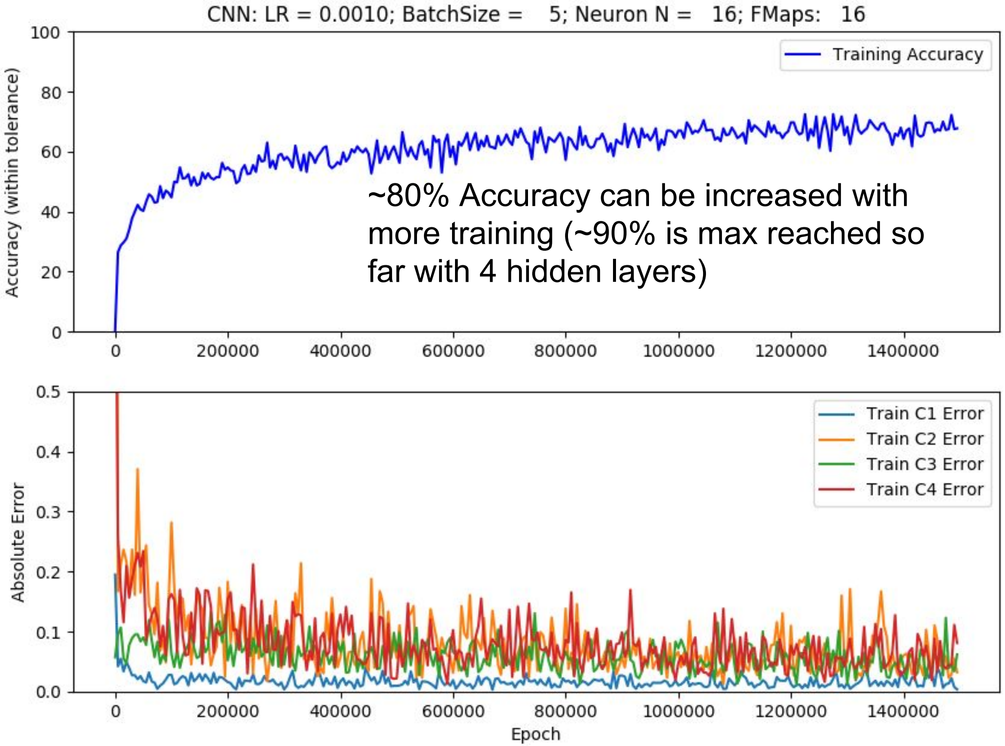


```
#======================================
# Heat flux generated by fortran progr
# Parameters Listed Below:
# c1 =   0.16051926045641973
# c2 =    1.6853282013754156
# c3 =    3.8702372531813245E-002
# c4 =  -0.59566003694506575
# Bp =   0.23307759505531836
# P =    4.2547901060334734
# fx =    16.066276188639492
# t =    5.0000000000000000
# R0 time varying, Freq =     0.0000000
```



Project Objective 2: Progress

Note: 500 datasets used here, no test dataset testing performed yet



"Accuracy" is when prediction is within 0.05 of C value.

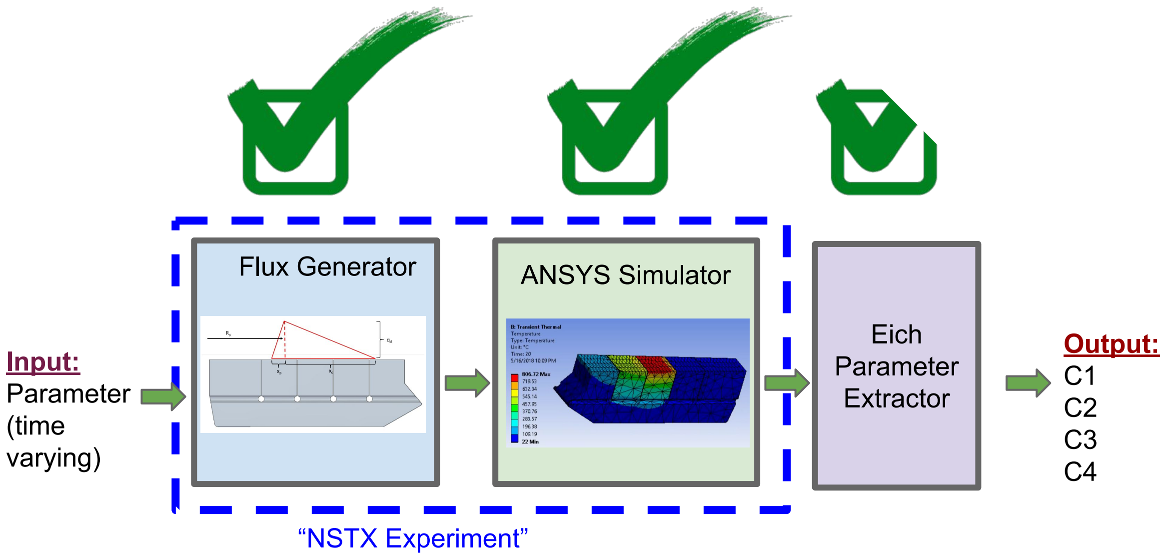
(using boxcar convolution running average with window length = 20 epochs)

"Error" signifies abs(prediction - target)

(error is averaged over epochs)

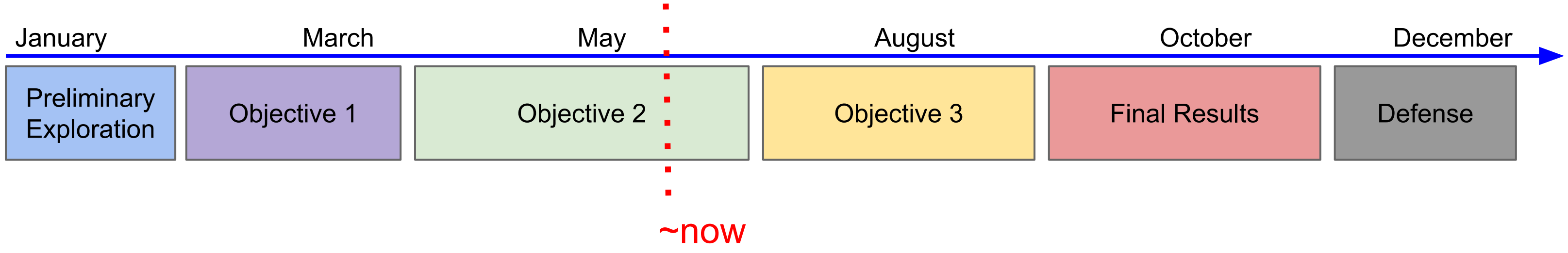
Project Objective 2: Progress

- Heat Flux Generator and ANSYS ACT script completed
- 3 NNs constructed
  - Deep Neural Network (DNN)
  - Convolution Neural Network (CNN)
  - Recurrent Neural Network (RNN)



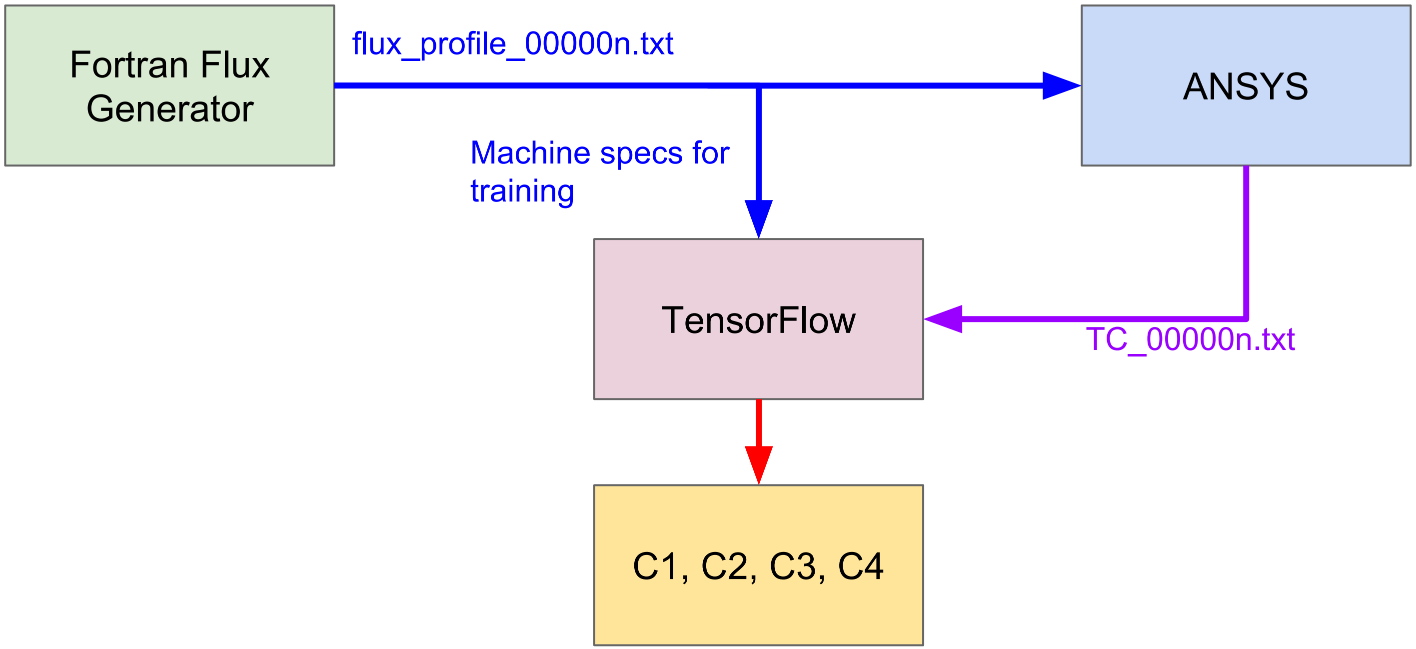
Next Steps

- Finish Objective 2: Eich parameter extraction
- Finish Objective 3: Add noise / error to the inputs
- Determine minimum number of shots for validation
  - Build importance map for entire domain
  - Locate high importance regions
  - Determine minimum shots to sample all linearly independent dimensions



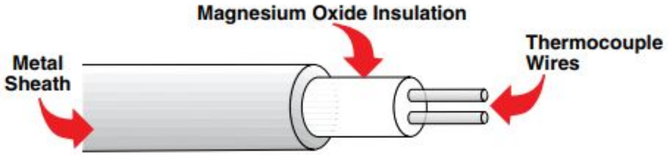
Questions…?

Concept Process Flow



Thermocouple: Data from Omega Datasheets

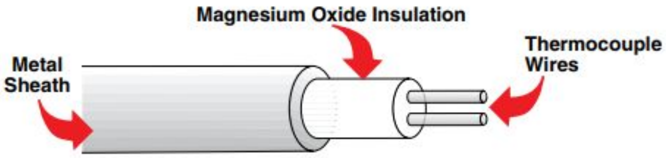
- Part Number: SCASS-040U-6-SHX
- Type K: Ni-Cr
- Diameter 0.04”



CHROMEGA®* ALOMEGA® 304 SS Sheath K	0.010"*	SCASS-010G-6-SHX	SCASS-010U-6-SHX	SCASS-010E-6-SHX
	0.020"	SCASS-020G-6-SHX	SCASS-020U-6-SHX	SCASS-020E-6-SHX
	0.032"	SCASS-032G-6-SHX	SCASS-032U-6-SHX	SCASS-032E-6-SHX
	0.040"	SCASS-040G-6-SHX	SCASS-040U-6-SHX	SCASS-040E-6-SHX
	0.062"	SCASS-062G-6-SHX	SCASS-062U-6-SHX	SCASS-062E-6-SHX
	0.125"	SCASS-125G-6-SHX	SCASS-125U-6-SHX	SCASS-125E-6-SHX

Thermocouple: Data from Omega Datasheets

- Max TC Temp:
  - ~1370°C
- Max Sheath Temp:
  - ~700°C
- If Range (20°C, 700°C) and 12-bit uProc
  - Max Possible Resolution = (700-20)/2^12 = ~0.166°C
- Voltage Resolution @ Max T Domain:
  - ~37.36 µV/°C
  - ~25mV total range for (20°C,700°C)



ANSI Code	ANSI MC 96.1 Color Coding: Thermocouple Grade	Extension Grade	Alloy Combination: + Lead	– Lead	Comments Environment Bare Wire	Maximum T/C Grade Temp Range	EMF (mV) Over Max Temp Range
K	+ –	+ –	CHROMEGA® NICKEL-CHROMIUM Ni-Cr	ALOMEGA® NICKEL-ALUMINUM Ni-Al (magnetic)	Clean Oxidizing and Iner Limited Use in Vacuum or Reducing. Wide Temperature Range, Most Popular Calibration	–270 to 1372°C –454 to 2501°F	–6.458 to 54.886

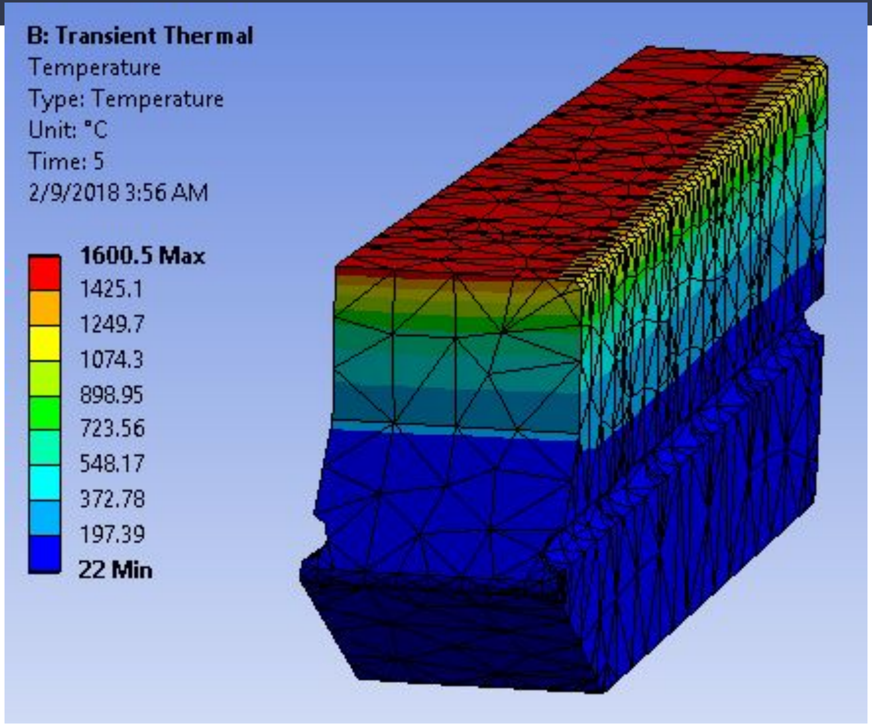
Upper Temperature Limit in °C (°F) of OMEGACLAD® Vs. Sheath Diameter

T/C \ Sheath Dia.	0.020" 0.5 mm	0.032" 0.8 mm	0.040" 1.0 mm	0.062" 1.6 mm	0.093" 2.4 mm	0.125" 3.2 mm	0.188" 4.8 mm	0.250" 6.3 mm
J	260 (500)	260 (500)	260 (500)	440 (825)	480 (900)	520 (970)	620 (1150)	720 (1300)
K & N	700 (1290)	700 (1290)	700 (1290)	920 (1690)	1000 (1830)	1070 (1960)	1150 (2100)	1150 (2100)
E	300 (570)	300 (570)	300 (570)	510 (950)	580 (1075)	650 (1200)	730 (1350)	820 (1510)
T	260 (500)	260 (500)	260 (500)	260 (500)	260 (500)	315 (600)	370 (700)	370 (700)

Initial Case 1: No TC Recess – Tile Slice

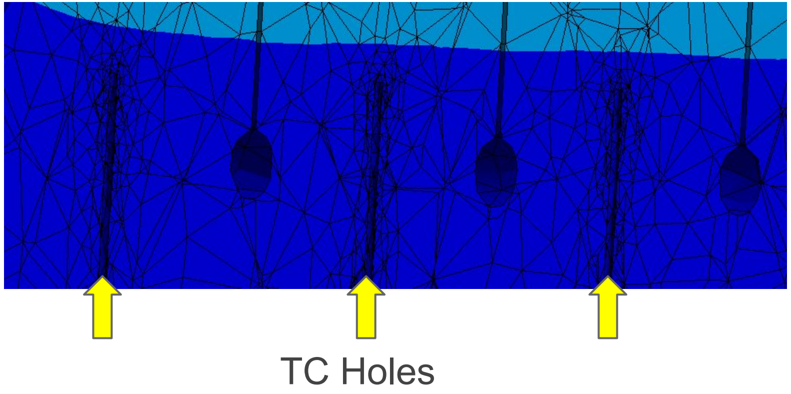
1600°C on upper tile surface reached at 7.75 MW/m^2

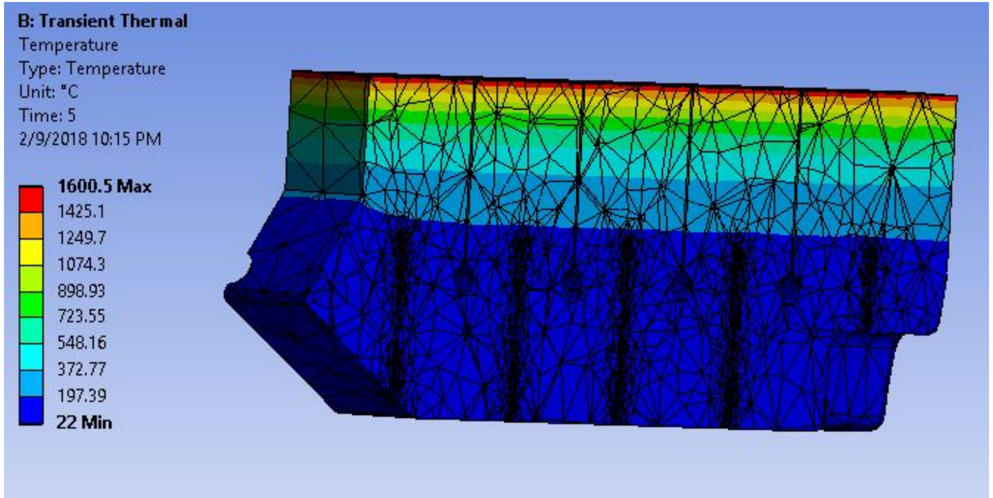
Max temp occurs at time of max flux during 30s simulation



Case 1: TC Recess 1”

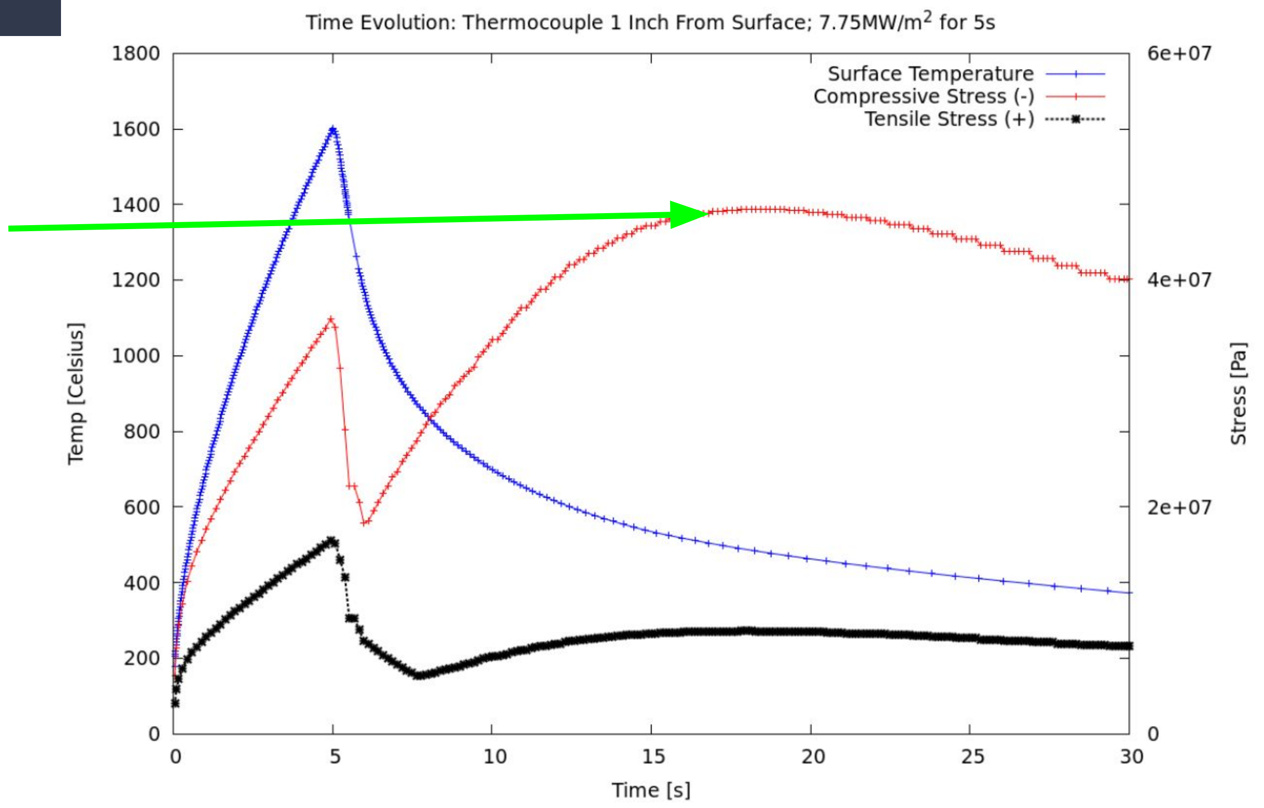
1600°C on upper tile surface reached at 7.75 MW/m^2





Case 1: TC Recess 1”: Temp/Stresses v. Time

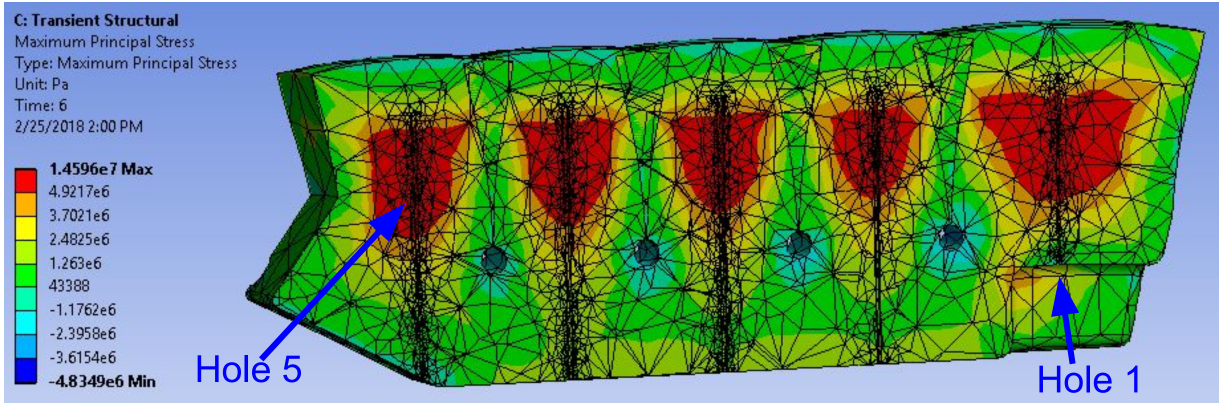
Exaggerated High Stress due to simplified boundary conditions



TC Analysis Example Slide: 5s Pulse

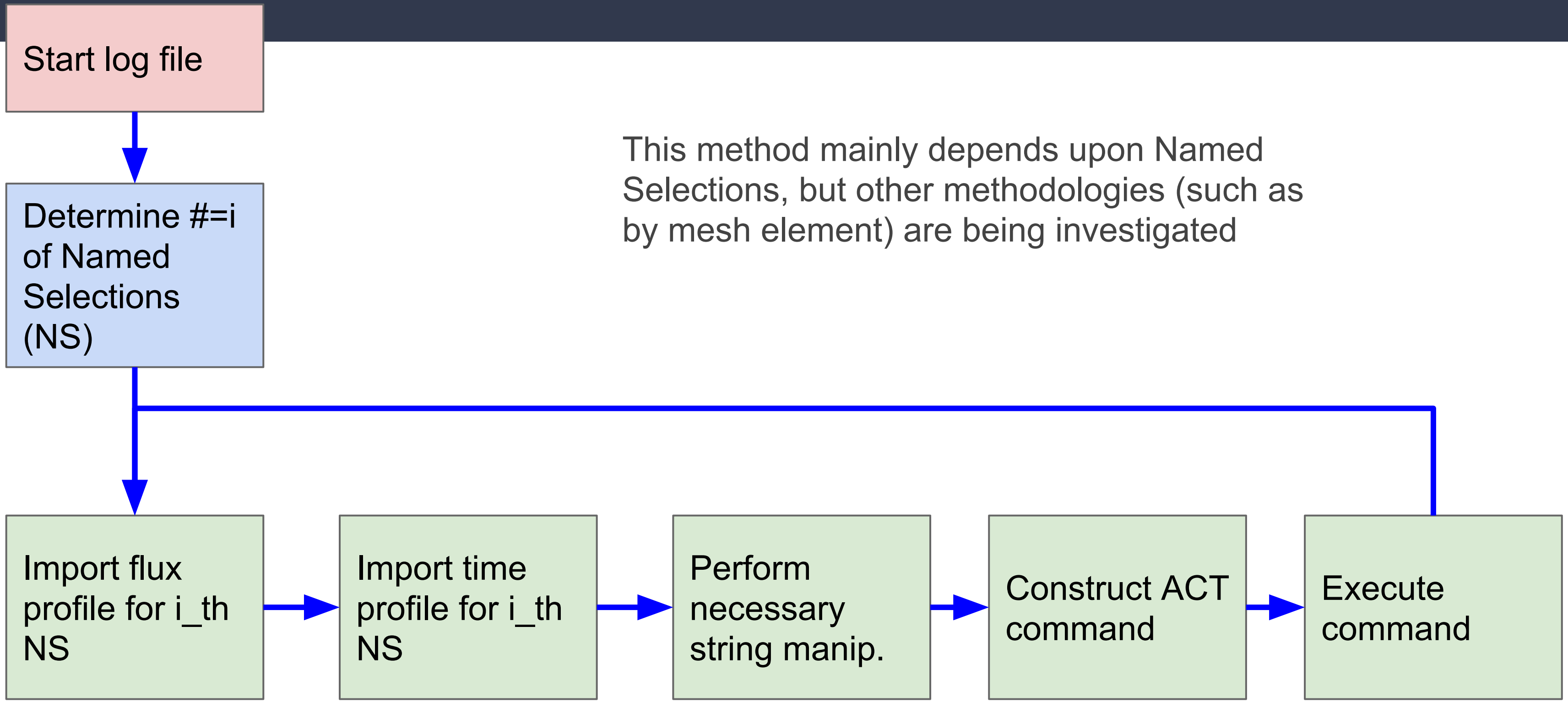
All stresses within nominal ranges. This is for 5s shot with 1600° reached on tile surface. Note that there will be some residual heat stresses (as indicated in last slide) that aren't included here.

Stress / Temp vs. TC Distance from Surface				
Dist. from Surface [in]	Max TC Hole Temp [degC]	Hole # [from IB]	Compressive Stress [MPa]	Tensile Stress [MPa]
0.3	789.79	4	38.565	14.752
0.5	562.37	4	38.565	10.889
1	210.17	1	38.565	8.2032

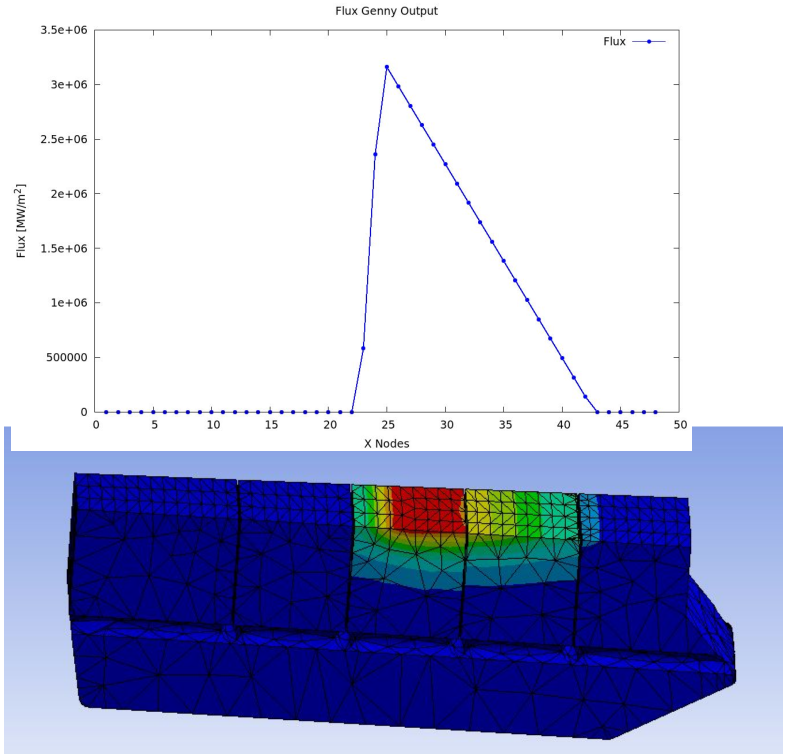


Flux Importation Algorithm

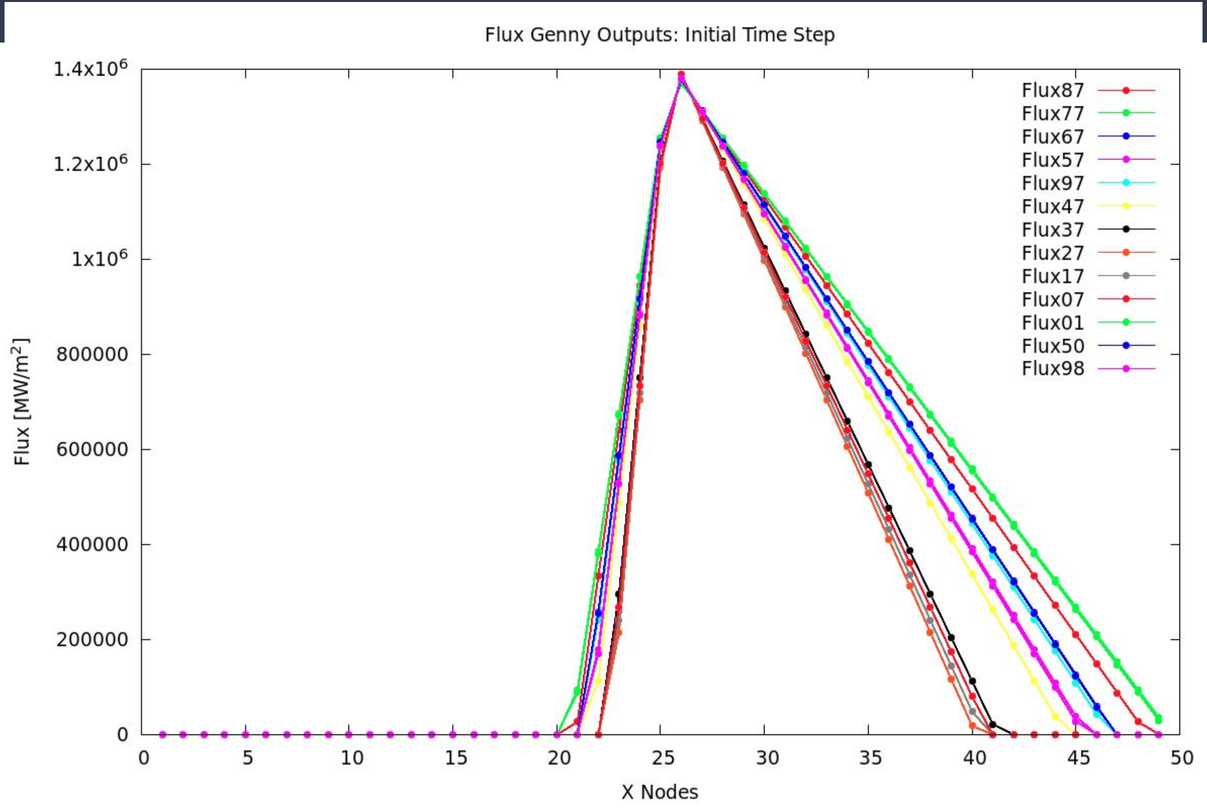
This method mainly depends upon Named Selections, but other methodologies (such as by mesh element) are being investigated



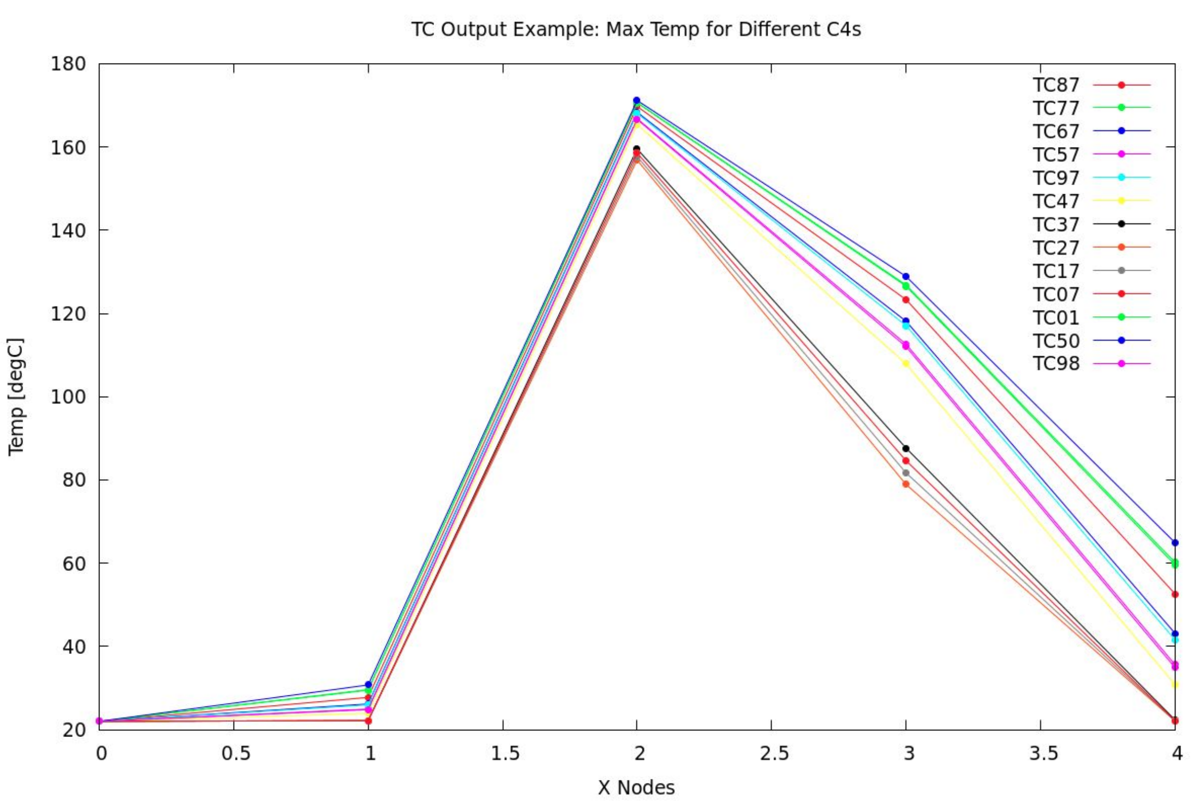
Flux Generator to ANSYS Example



Flux profile comparison: varying c4 only

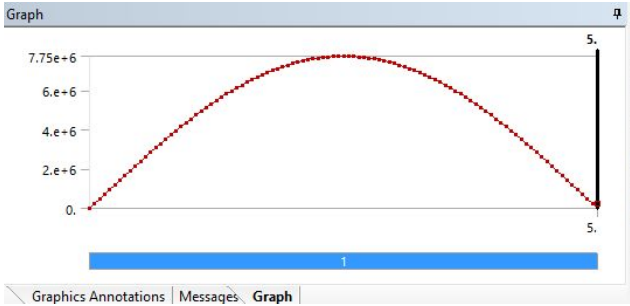


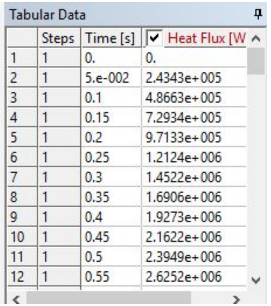
TC profile comparison: max temp: C4 only



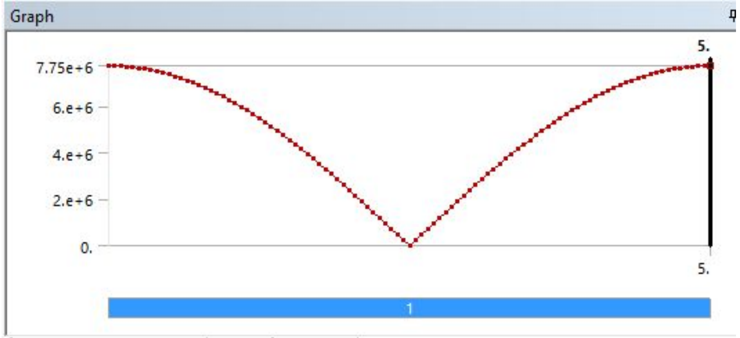
Example Fluxes

Each of these ‘Heat Flux’ modules can have an independent time dependent flux profile. Example below has only two seperate fluxes, but arbitrary resolution is possible (within reason…).




Tabular Data

	Steps	Time [s]	Heat Flux [W
1	1	0.	0.
2	1	5.e-002	2.4343e+005
3	1	0.1	4.8663e+005
4	1	0.15	7.2934e+005
5	1	0.2	9.7133e+005
6	1	0.25	1.2124e+006
7	1	0.3	1.4522e+006
8	1	0.35	1.6906e+006
9	1	0.4	1.9273e+006
10	1	0.45	2.1622e+006
11	1	0.5	2.3949e+006
12	1	0.55	2.6252e+006



TensorFlow

- Developed by Google Brain team
- Made for dataflow programming
- Open Source
- Lots of blogs, forums, examples, etc.
- Easy to implement NNs, CNNs, predictors, etc.
- Python Programming
- APIs for a myriad of packages
  - Plotting
  - Maps
  - Error Tracking
- Can be compiled on almost any CPU arch.

TensorFlow Example: Iris

Trained model to identify 3 types of iris (source: TensorFlow website). Data comes in as a list of vectors: [a, b, c, d, answer]. a-d is dataset, answer is the desired model output (used for training).

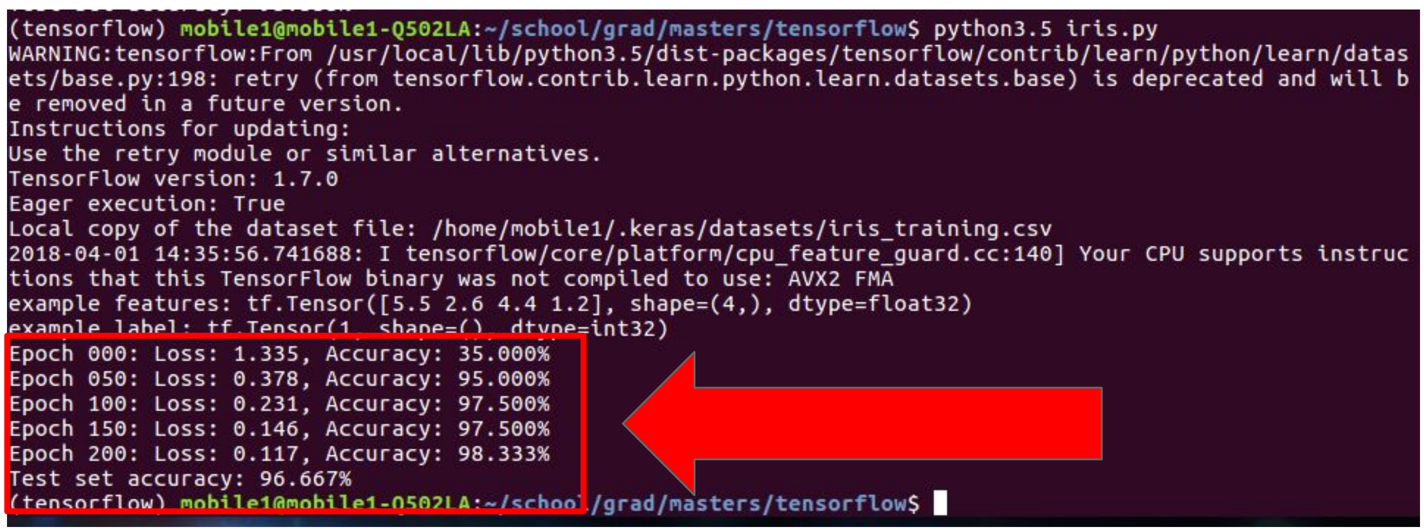
Input Data:

```
is_training.csv
120,4,setosa,versicolor,virginica
6.4,2.8,5.6,2.2,2
5.0,2.3,3.3,1.0,1
4.9,2.5,4.5,1.7,2
4.9,3.1,1.5,0.1,0
5.7,3.8,1.7,0.3,0
4.4,3.2,1.3,0.2,0
5.4,3.4,1.5,0.4,0
```

Figure 1. Iris setosa (by Radomil, CC BY-SA 3.0), Iris versicolor (by Dlanglois, CC BY-SA 3.0), and Iris virginica (by Frank Mayfield, CC BY-SA 2.0).

TensorFlow Example: Iris

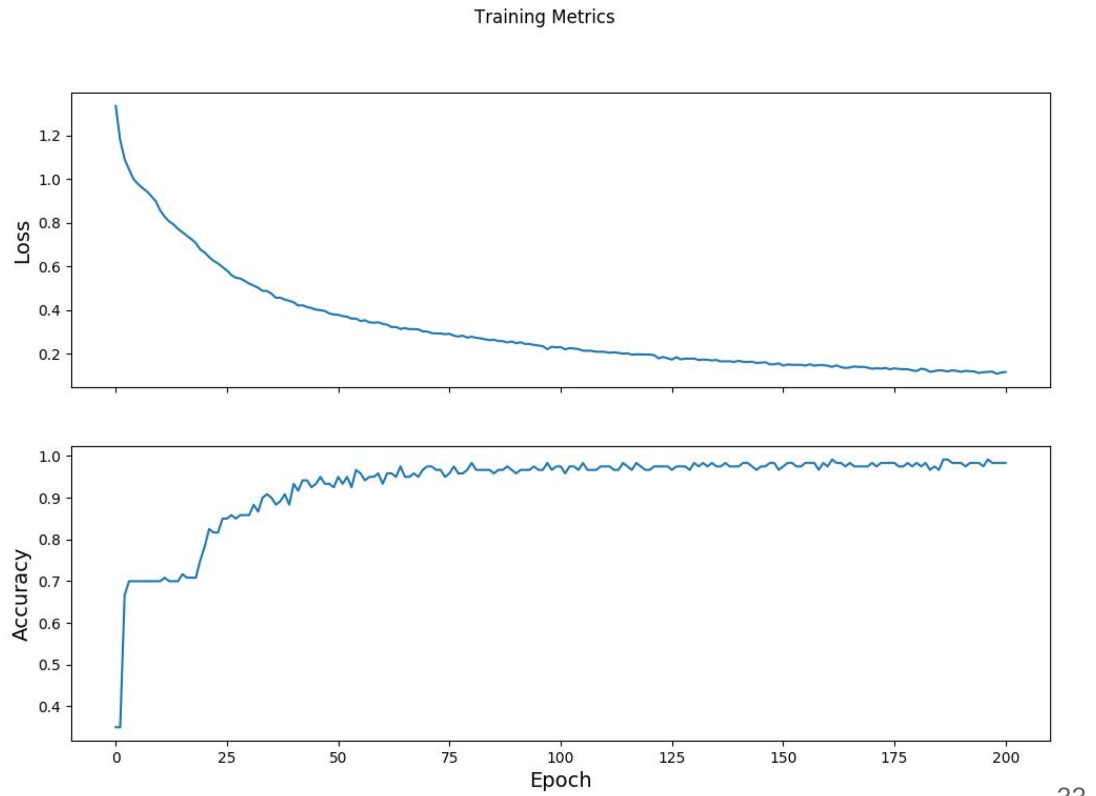
Real time output in terminal on my local machine. Uses 3 Layer NN with 10 nodes each. Rectified Linear Unit (ReLU) activation function.



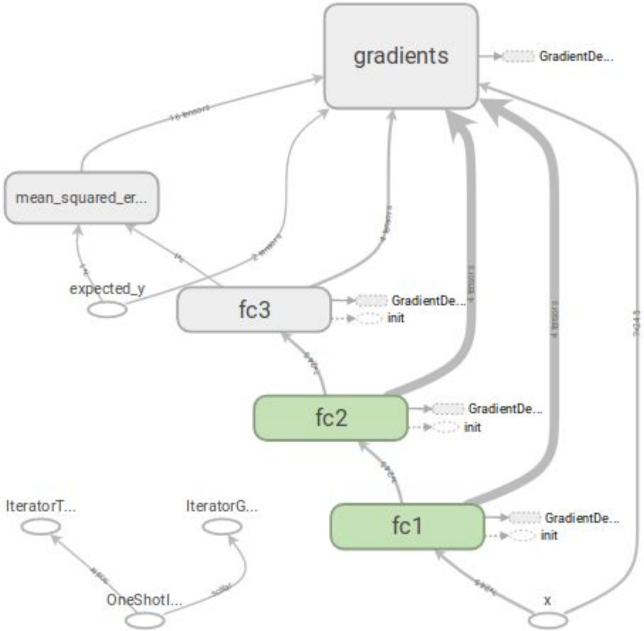
```
(tensorflow) mobile1@mobile1-Q502LA:~/school/grad/masters/tensorflow$ python3.5 iris.py
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow/contrib/learn/python/learn/datas
ets/base.py:198: retry (from tensorflow.contrib.learn.python.learn.datasets.base) is deprecated and will b
e removed in a future version.
Instructions for updating:
Use the retry module or similar alternatives.
TensorFlow version: 1.7.0
Eager execution: True
Local copy of the dataset file: /home/mobile1/.keras/datasets/iris_training.csv
2018-04-01 14:35:56.741688: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instruc
tions that this TensorFlow binary was not compiled to use: AVX2 FMA
example features: tf.Tensor([5.5 2.6 4.4 1.2], shape=(4,), dtype=float32)
example label: tf.Tensor(1, shape=(), dtype=int32)
Epoch 000: Loss: 1.335, Accuracy: 35.000%
Epoch 050: Loss: 0.378, Accuracy: 95.000%
Epoch 100: Loss: 0.231, Accuracy: 97.500%
Epoch 150: Loss: 0.146, Accuracy: 97.500%
Epoch 200: Loss: 0.117, Accuracy: 98.333%
Test set accuracy: 96.667%
(tensorflow) mobile1@mobile1-Q502LA:~/school/grad/masters/tensorflow$
```

TensorFlow Example: Iris

- Plot Error (termed loss) and model accuracy. This is on training data.
- Matplotlib has a direct interface to tensorflow for plotting
- TensorBoard is another option for creating visual maps and plots

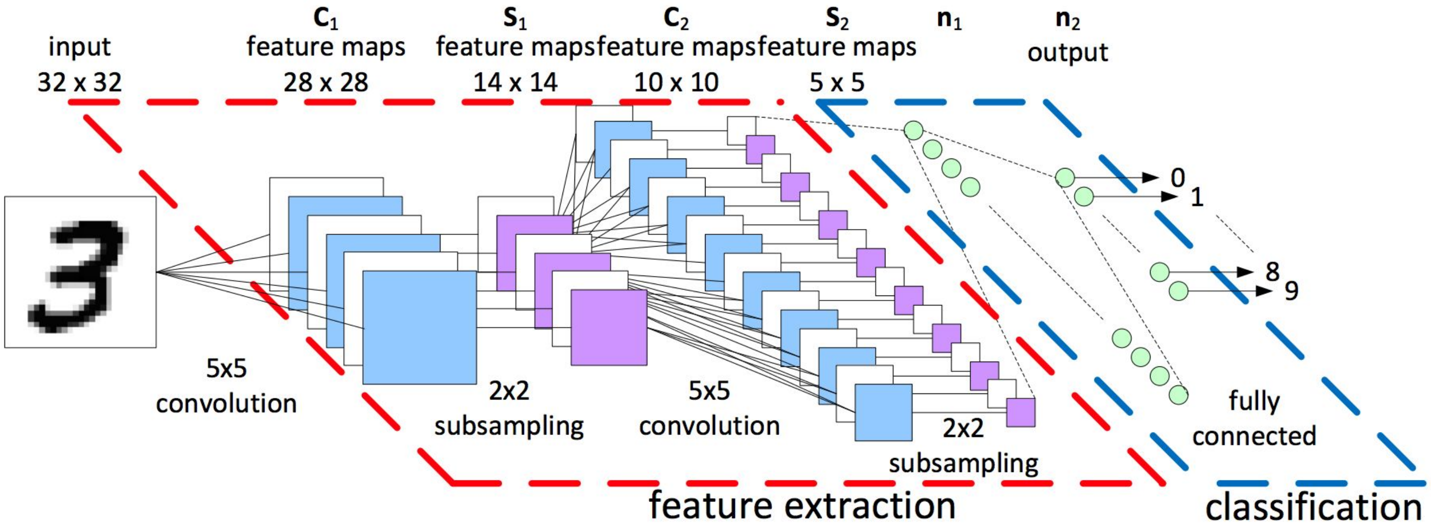


Tensorboard



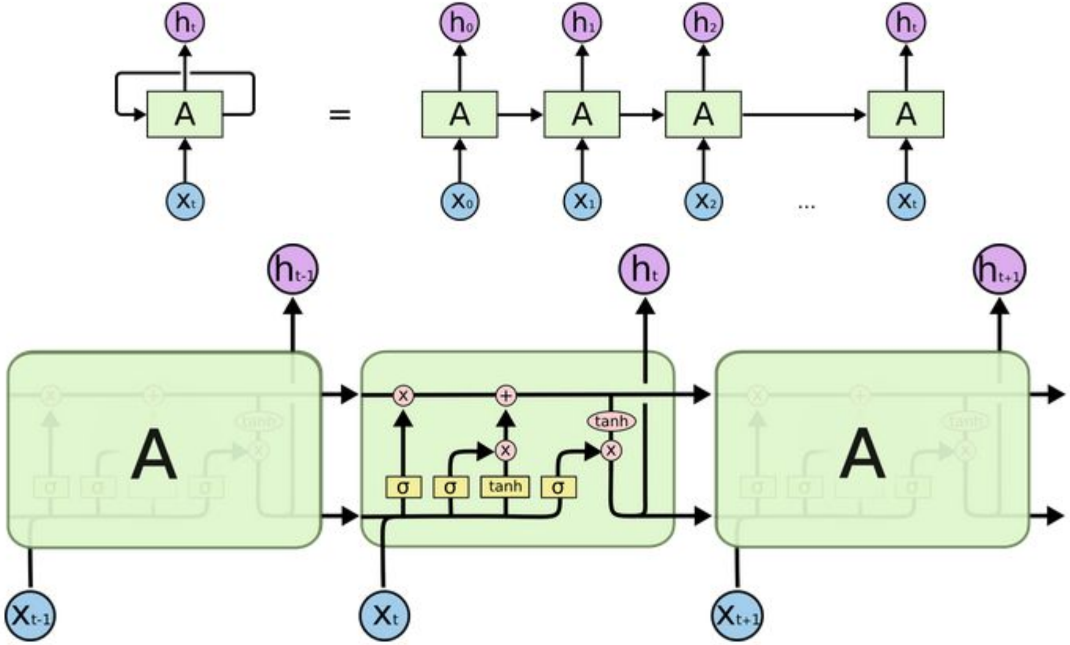
GUI for NN visualization and heuristic generation

Basic CNN



C_1
S_1
C_2
S_2
n_1
n_2
input
32 x 32
feature maps
28 x 28
feature maps
14 x 14
feature maps
10 x 10
feature maps
5 x 5
output
0
1
8
9
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
fully
connected
feature extraction
classification

Recurrent Neural Networks



http://colah.github.io/posts/2015-08-Understanding-LSTMs/