MFiXAI Overview

Dirk Van Essendelft, Terry Jordan, Mino Woo, Tarak Nandi

Project Objective

Research Goal

Build an advanced collaborative framework specifically targeted towards CFD on the most advanced HPC/AI hardware with native support for AI and ML algorithms

Aligned with FE Objectives

Increasing computational speed without sacrificing accuracy will directly support:

- Modernization of existing coal plants
- development of coal plants of the future
- Reduction of the cost of carbon capture, utilization, and storage (CCUS)

Project Origins

Driving Question

MFiX + TensorFlow

Can we write MFiX in TensorFlow so that we can create a single, unified framework for doing both CFD and AI/ML on emerging hardware designed for AI/ML?

- TensorFlow is the most used AI/ML framework
- TensorFlow has a simple API and allows for both surface level hardware agnostic coding and the ability to deeply optimize hardware specific implementations if needed
- Get speed boosts from AI/ML hardware
- Get speed boost from AI/ML accelerated algorithms
- Simplify implementation of AI/ML models in MFiX

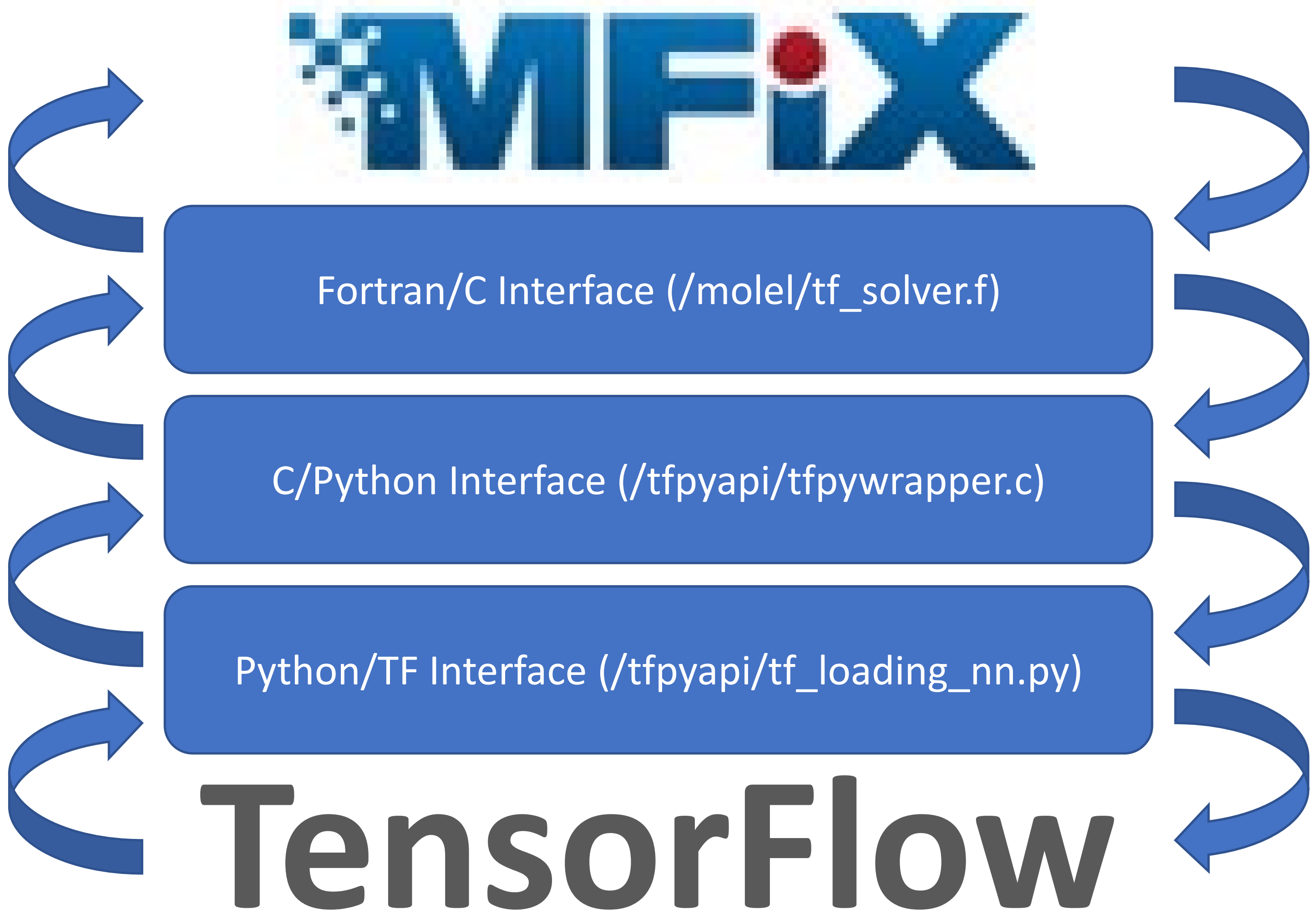
Current Status

Where are we now?

- In Second Full Year of Development, First year in CARD
- Ahead of schedule with Milestones
  - EY20.4.A A demonstration of a multidevice linear solver relative to the existing single device solver (9/30/2020)
  - EY20.4.B A demonstration of a granular simulations using the TensorFlow based solver. (9/30/2020)
  - EY21.4.C A demonstration of a simple fluid bed simulation in the TensorFlow based solver. (3/30/2020)
- Have a functioning, coupled MP-PIC code implemented in TensorFlow
  - Solves all transport equations on available devices followed by a multi-device solve of continuity
  - Ready to accept AI/ML models
  - Does not yet support complex geometries
  - Does not yet support energy, species, or reactions

MFiX – TensorFlow Interface

Making MFiX Talk To TensorFlow

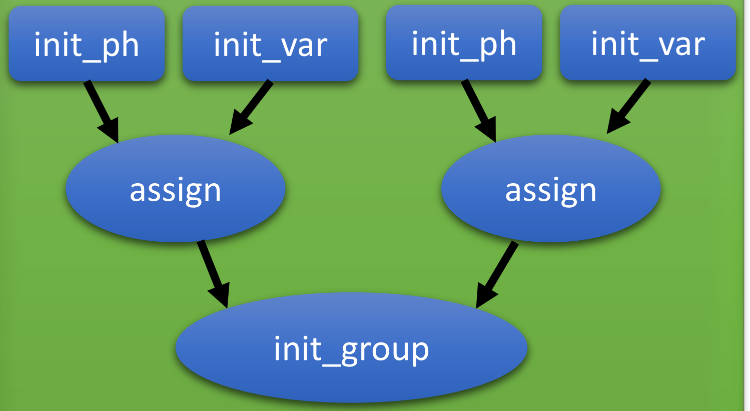

MFiX
Fortran/C Interface (/molel/tf_solver.f)
C/Python Interface (/tfpyapi/tfpywrapper.c)
Python/TF Interface (/tfpyapi/tf_loading_nn.py)
TensorFlow

TensorFlow: Computation Strategy

Break the problem up into sub-graphs which "talk" using variables

Initialization

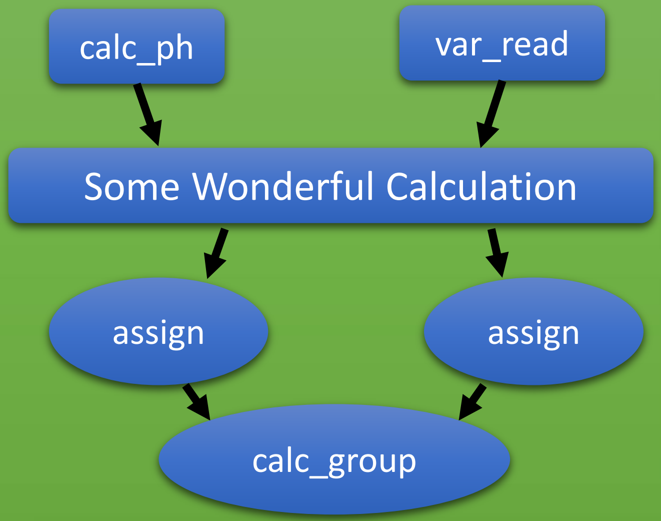
Initialize all data that needs to be persistent as variables



```
sess.run(init_group, feed_dict=init_feed)
```

Calculation

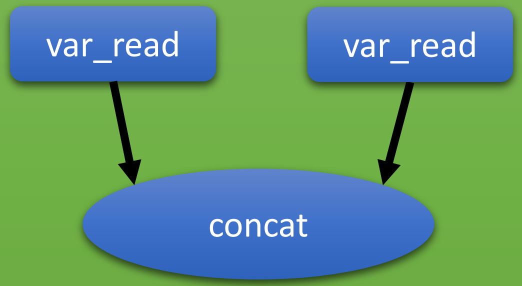
Calculate, but end in an assignment to a variable and call assign op(s)



```
sess.run(calc_group, feed_dict=calc_feed)
```

Data Retrieval

Use Data Retrieval as sparingly as possible, especially on accelerators



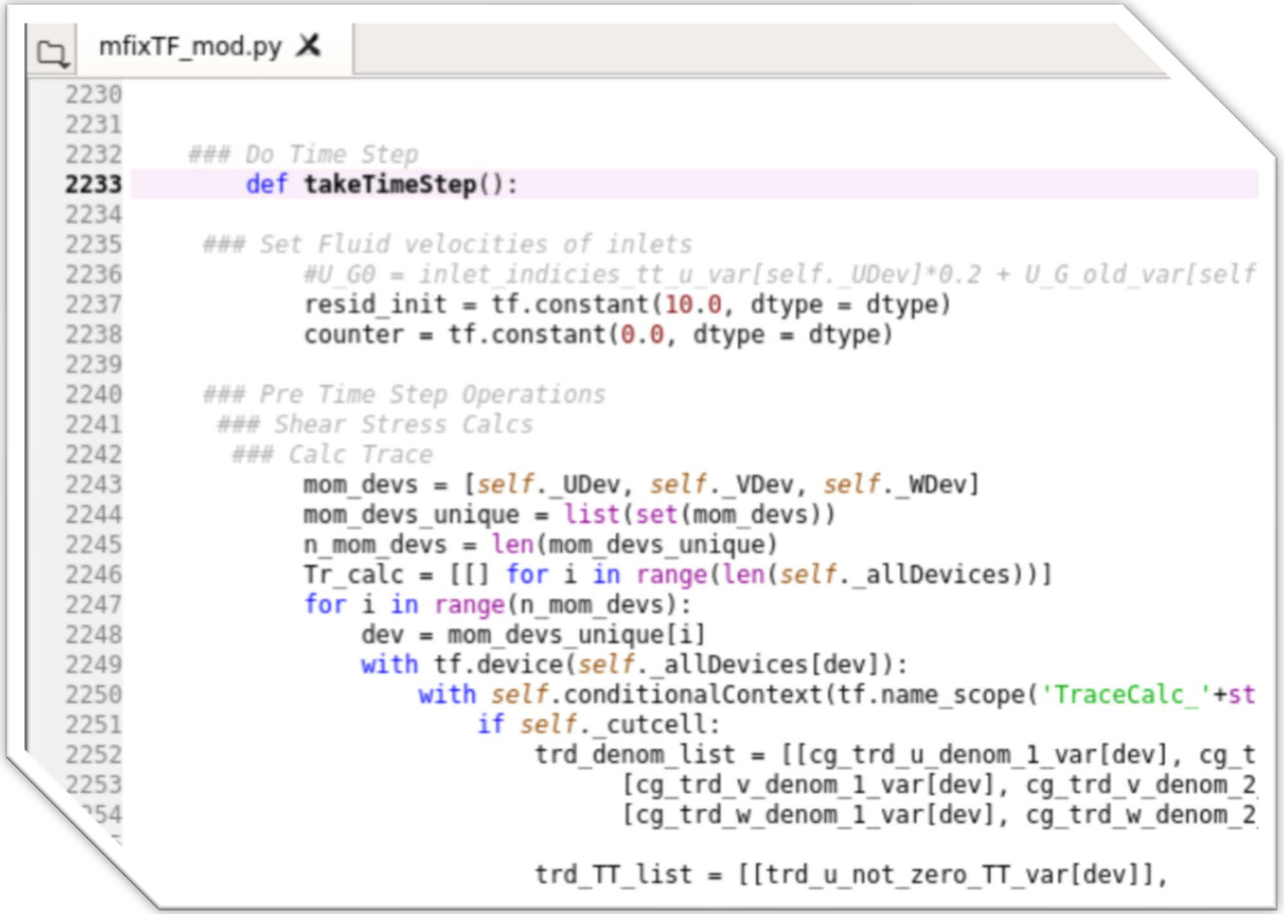
```
output = sess.run(concat)
```

```
outputList = sess.run([var1, var2])
```

Where We are Now

MP-PIC Capable TensorFlow Solver

- Entry Point to The code is mfixTF_mod.py
- 2/3 of this file is setting up input graph
- Solver Starts at takeTimeStep
- SIMPLE Iteration defined in outerLoopBody
- Most formation steps are done in *_mod.py files
- We use TensorFlow custom operators for many critical subroutines for efficiency
    - Custom ops live in /tfpyapi/customops

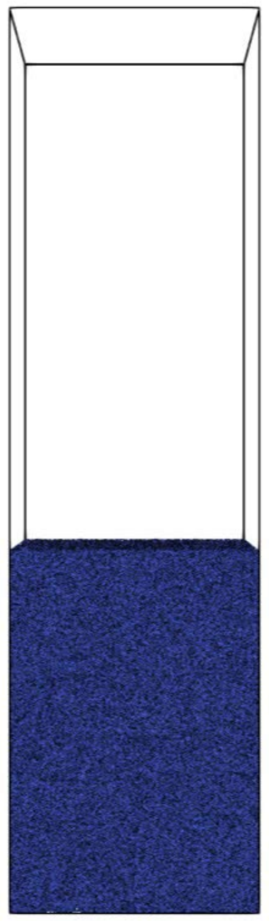

mfixTF_mod.py

```


    ### Do Time Step
        def takeTimeStep():

     ### Set Fluid velocities of inlets
            #U_G0 = inlet_indicies_tt_u_var[self._UDev]*0.2 + U_G_old_var[self
            resid_init = tf.constant(10.0, dtype = dtype)
            counter = tf.constant(0.0, dtype = dtype)

     ### Pre Time Step Operations
      ### Shear Stress Calcs
       ### Calc Trace
            mom_devs = [self._UDev, self._VDev, self._WDev]
            mom_devs_unique = list(set(mom_devs))
            n_mom_devs = len(mom_devs_unique)
            Tr_calc = [[] for i in range(len(self._allDevices))]
            for i in range(n_mom_devs):
                dev = mom_devs_unique[i]
                with tf.device(self._allDevices[dev]):
                    with self.conditionalContext(tf.name_scope('TraceCalc_'+st
                        if self._cutcell:
                            trd_denom_list = [[cg_trd_u_denom_1_var[dev], cg_t
                                   [cg_trd_v_denom_1_var[dev], cg_trd_v_denom_2
                                   [cg_trd_w_denom_1_var[dev], cg_trd_w_denom_2

                            trd_TT_list = [[trd_u_not_zero_TT_var[dev]],
```

Where We are Now

MP-PIC Capable TensorFlow Solver

- Wrote a first generation distributed linear solver in TensorFlow that can accelerate the fluid solver by as much as 4x compared to MFiX Classic (next generation solvers could be much more)
- Wrote a first-generation MP-PIC granular motion solver that is almost 10x faster than MFiX classic
- Coupled the fluid and solids solvers to simulate simple fluid beds
- Ready to accept arbitrary AI models anywhere in the code
- Validated against analytical solutions
- Ongoing verifications against MFiX Classic

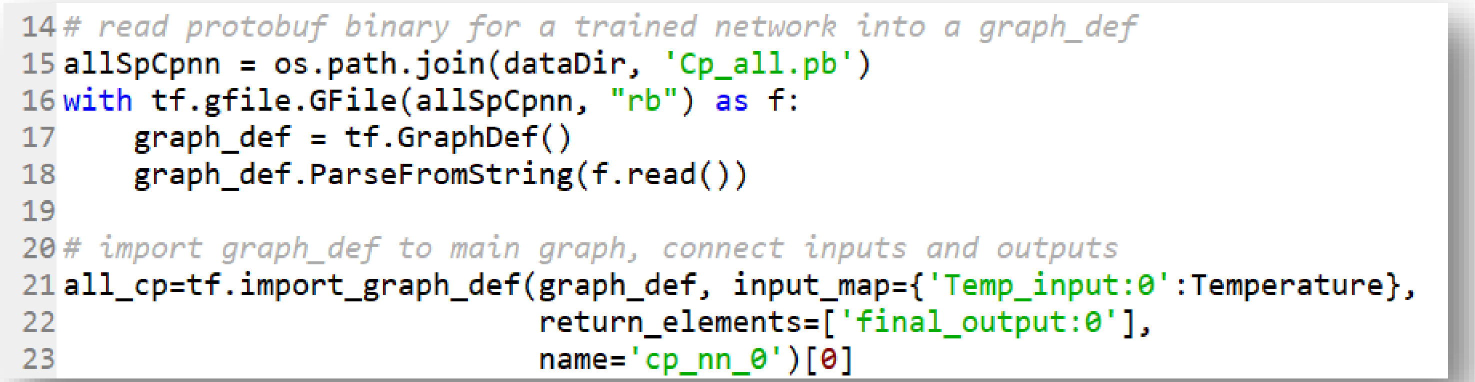

3.8 Million Cells, 16.8 Million Parcels

Cognitive In-The-Loop MFIXAI

Making CFD Smart with ML

Very Easy ML Integration

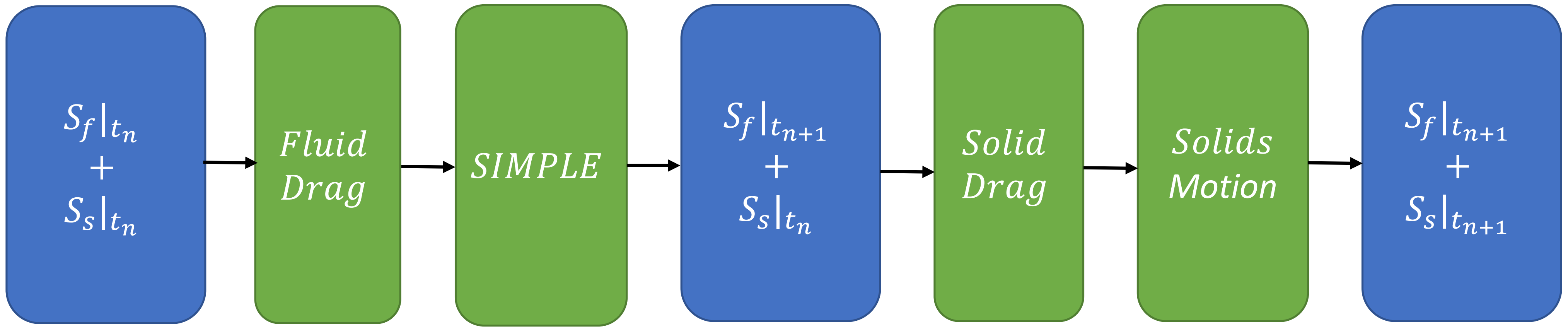
- Load Trained Model
- Connect Inputs and outputs
- Done
- Example: Specific Heat Calcs
    - Almost twice as fast as evaluating polynomials (even without tensor cores)
    - All species properties predicted at once with two matrix multiplications

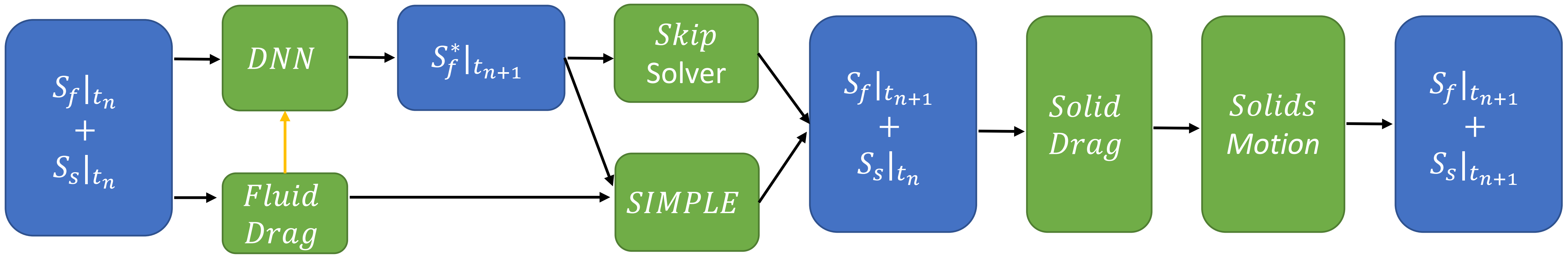


```python
# read protobuf binary for a trained network into a graph_def
allSpCpnn = os.path.join(dataDir, 'Cp_all.pb')
with tf.gfile.GFile(allSpCpnn, "rb") as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())

# import graph_def to main graph, connect inputs and outputs
all_cp=tf.import_graph_def(graph_def, input_map={'Temp_input:0':Temperature},
                           return_elements=['final_output:0'],
                           name='cp_nn_0')[0]
```

Cognitive In-The-Loop MFIXAI

Integrating ML into a E-L Solver


$S_f|_{t_n} + S_s|_{t_n}$
Fluid Drag
SIMPLE
$S_f|_{t_{n+1}} + S_s|_{t_n}$
Solid Drag
Solids Motion
$S_f|_{t_{n+1}} + S_s|_{t_{n+1}}$

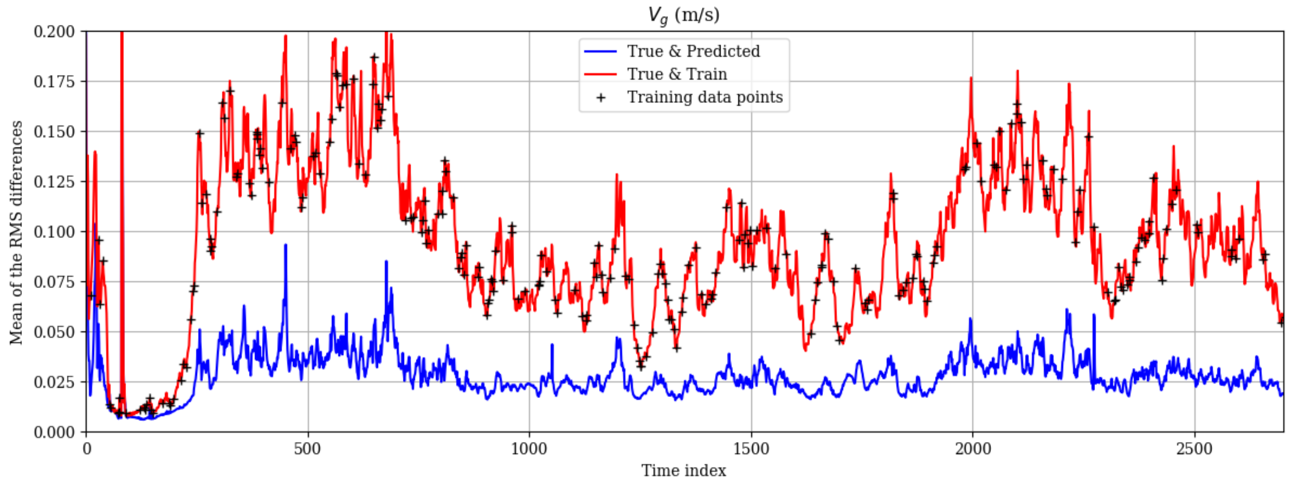

$S_f|_{t_n} + S_s|_{t_n}$
DNN
$S_f^*|_{t_{n+1}}$
Skip Solver
Fluid Drag
SIMPLE
$S_f|_{t_{n+1}} + S_s|_{t_n}$
Solid Drag
Solids Motion
$S_f|_{t_{n+1}} + S_s|_{t_{n+1}}$

Better Predictions by the DNN mean fewer SIMPLE iterations

Cognitive In-The-Loop MFIXAI

Predictions from a ML model: transient flow in a fluidized bed

- Trained on randomly chosen 10% data points
- Training data at time step *n*
- True and predicted data at *n+1*



- Red line is actual change in the field variables in CFD
- Blue line is the error in the predictions
- Perfect learning would drive the difference between the True and the Predicted data (blue line) to zero.
- Our results indicate a partially learned ML model. The subsequent SIMPLE iterations will be used to drive the errors even lower.

Better Predictions by the DNN mean fewer SIMPLE iterations

Cognitive In-The-Loop MFIXAI

Best Practices for AI/ML Development

- Make sure you input/output data available MFiX variables
  - Important when scale bridging or generating training data from different codes
- Name input/output tensors with name parameter or tf.identity method
  - Makes connecting to MFIXAI much easier
- Make Your Models Accept/Output Individual Rank 1 Tensors
  - All field variables and most derivative variables are rank 1 tensors in IJK/particle index order
- Make Your Models Accept/Output natural variables in SI units
  - Do normalization/un-normalization in the TF graph as part of training not as part of data prep
  - Embeds data pre/post processing in model so users don't have to worry about it when applying it
- Make sure you use tensors/placeholders with an undefined axis 0 shape
  - Makes models agnostic to cell/particle numbers and avoids need to do chunking, maintains efficiency

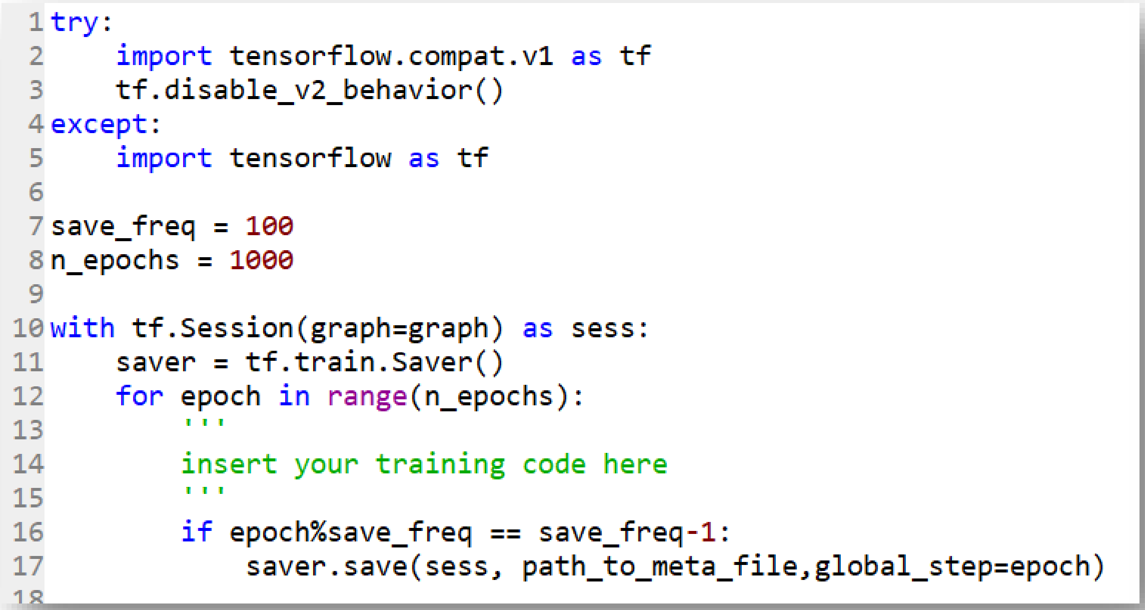
Cognitive In-The-Loop MFIXAI

Best Practices for AI/ML Development

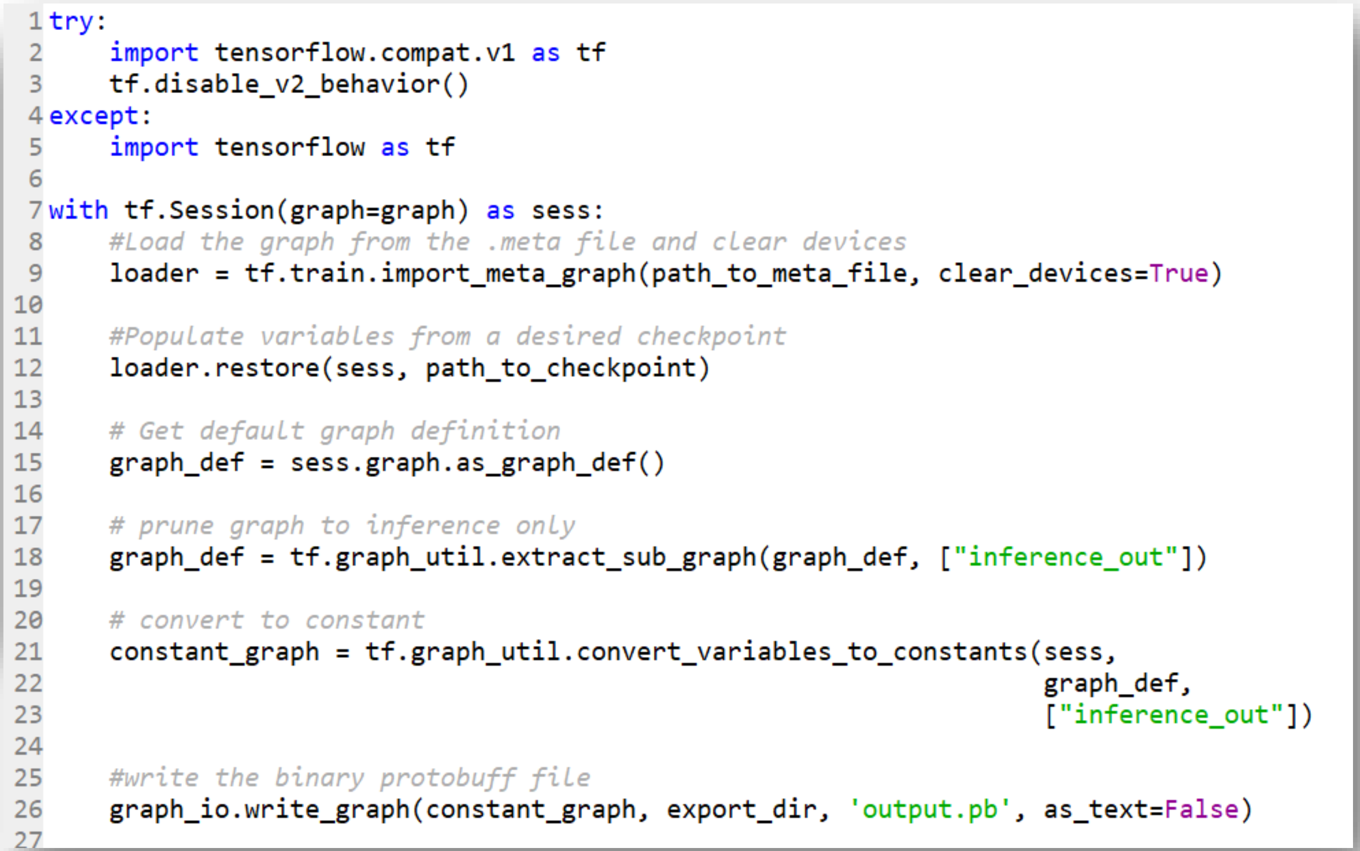
- Make sure you save the graph with device placements cleared so that the model can be placed on the devices local to the data being used
    - Recommend using the older tf.train interface to load/save models during training because it has options to automatically clear devices
- Final output should be saved in Protobuf format (*.pb)
    - Easiest and most portable format to use in MFIXAI
- Make sure to freeze and prune the graph prior to export as *.pb
    - No variables, no extra nodes, smallest possible representation
- Be careful to use ops and dtypes that are compatible with GPU's
    - Off GPU ops are a performance killer
- Please catalog and share all *.py, *.index, *.meta, *.data, *.h5 and *.pb files
    - Makes debugging easier, enables collaboration, enables work extensions

Cognitive In-The-Loop MFIXAI

Best Practices for AI/ML Development



```python
try:
    import tensorflow.compat.v1 as tf
    tf.disable_v2_behavior()
except:
    import tensorflow as tf

save_freq = 100
n_epochs = 1000

with tf.Session(graph=graph) as sess:
    saver = tf.train.Saver()
    for epoch in range(n_epochs):
        '''
        insert your training code here
        '''
        if epoch%save_freq == save_freq-1:
            saver.save(sess, path_to_meta_file,global_step=epoch)
```



```python
try:
    import tensorflow.compat.v1 as tf
    tf.disable_v2_behavior()
except:
    import tensorflow as tf

with tf.Session(graph=graph) as sess:
    #Load the graph from the .meta file and clear devices
    loader = tf.train.import_meta_graph(path_to_meta_file, clear_devices=True)

    #Populate variables from a desired checkpoint
    loader.restore(sess, path_to_checkpoint)

    # Get default graph definition
    graph_def = sess.graph.as_graph_def()

    # prune graph to inference only
    graph_def = tf.graph_util.extract_sub_graph(graph_def, ["inference_out"])

    # convert to constant
    constant_graph = tf.graph_util.convert_variables_to_constants(sess,
                                                                  graph_def,
                                                                  ["inference_out"])

    #write the binary protobuff file
    graph_io.write_graph(constant_graph, export_dir, 'output.pb', as_text=False)
```

Conclusions

Building the First Cognitive Simulation Capability in TensorFlow

- TensorFlow is proving to be a powerful framework for both physical simulations and AI/ML
  - Wonderful framework for Cognitive Simulations!!!
- Following a few simple rules makes AI/ML integration trivial
- Register on the MFiX website and email me your name, user-name, and email and I will add you to the repositories for MFIXAI and all the AI/ML training tools (dirk.vanessendelft@netl.doe.gov)
- Please feel free to reach out to us. We will help where we can and are more than happy to provide guidance to ensure that your tools integrate with the maximum possible ease