The simulationDriver for cantera is being upgraded to the YAML way because it will allow surface reactions to be modified on the fly.

To convert to the YAML way, we have two things to convert:

1. The input files (currently in cti format for starting top of cti file and also at intermediate steps).
2. The functions from my simulationDriver.

The functions are actually easier to convert. However, we need to convert the input files to the YAML way first because the old cti way does not create the right reaction object type (in cantera python objects) for on-the-fly modification.

\*\*\*

First, let’s consider how the CTI file gets made so that we can switch it to the YAML way. Actually, we can just, for now, use the CTI to YAML converter at the very end of full cti file creation because we already know that will convert to a suitable YAML file.

The current flow basically occurs inside a single function, as seen inside example\_runfile.py, it is the function call: canteraSimulate.create\_cti\_and\_SimulatePFRorTPRwithCantera

That function takes a CTI top info file (like ceO2\_cti\_top\_info.cti ) which has the species names and phases, and it also takes an array of *reactions and parameters*.

It passes the information to another function, create\_cti\_and\_cantera\_phases, which creates the cti\_file using canteraKineticsParametersParser.create\_full\_cti and then goes on to create the phases using cantera directly.

Those phases are then passed back to canteraSimulate.create\_cti\_and\_SimulatePFRorTPRwithCantera which then also calls the simulation function.

So to interrupt that flow and introduce YAML while allowing backwards compatibility, there are a few obvious ways to do it.

1. Create a function create\_yaml\_and\_cantera\_phases, make that function then call a helper function called canteraKineticsParametersParser.create\_full\_yaml
   1. Make that function then simply convert the cti file.
   2. Note: The YAML way still has a file that can be divided into top and full. So it may be worth converting just the top into yaml with the cti converter and to then add the yaml way.
   3. It looks like I can probably take the cti reactions string and convert it directly the yaml way with the yaml converter rather than having to make the yaml. This also has a backwards compatibility advantage in that the cti to yaml converter will automatically convert “surface\_reaction” into “reaction”
      1. The following syntax seems to work (but see next point):
         1. import cantera
         2. cantera.cti2yaml.convert(text='surface\_reaction("Acetaldehyde + CeCation(S) => Acetaldehyde1-Ce(S)", stick(1, 0, 2000))', output\_name="Testing.yaml")
      2. However, it seems to be only appropriate, or at least most appropriate, for converting an entire cti file string to yaml. So evidently I will need to change “create\_full\_cti” to “create\_full\_yaml” essentially one line at a time, but it does not seem too hard.

There is going to need to be a full development directory for this upgrade, and it will need the following steps.

1. Execute the existing run.
2. Use the cantera cti to yaml converter to get the \*\*full\*\* file into yaml. Use the full yaml to do the simulation (instead of the cti).
   1. Will do this by making this function: “create\_full\_yaml” as hardcoded to point to the converted yaml file.
   2. To use that for the simulation, I need to also create create\_yaml\_and\_cantera\_phases
   3. Then can try to do a simulation using
3. Then go back and fix create\_full\_yaml.
   1. As described below, will initially need to use ct.Reaction.fromYaml, then later will need to change to ct.Reaction.from\_yaml for cantera versions greater than 2.6. (so should do a version check).

\*\*\*

1. – could not run the existing cti file with cantera 2.6.0. It seems it’s no longer compatible. I am getting an error that looks like this if I try to run example-runfile.py:

File "C:\Users\fvs\Documents\GitHub\PEUQSE\PEUQSE\simulationDriver\canteraSimulate.py", line 247, in simulatePFRorTPRwithCantera

gas\_rates.append(surf.get\_net\_production\_rates('gas'))

File "cantera/kinetics.pyx", line 978, in cantera.\_cantera.InterfaceKinetics.get\_net\_production\_rates

File "cantera/kinetics.pyx", line 946, in cantera.\_cantera.InterfaceKinetics.\_phase\_slice

File "cantera/kinetics.pyx", line 943, in cantera.\_cantera.InterfaceKinetics.phase\_index

KeyError: 'gas'

I was able to install an earlier version of cantera using: conda install --channel cantera cantera==2.5

Now I was able to run “example-runfile.py”

Additionally, I went to the directory C:\Users\Public\OneDrive - Oak Ridge National Laboratory\DataAnalysis\220518\_surf\_pfr\_modify\_Reaction\_testing and I was able to run that yaml file with version 2.5 also, and even able to modify the surface rate constants. I just needed to use a slightly older (and now deprecated) syntax:

reactionObjectNew = ct.Reaction.fromYaml(rxnStringWithValues , kinetics=surf) #This is the correct way for cantera version 2.5

#reactionObjectNew = ct.Reaction.from\_yaml(rxnStringWithValues , kinetics=surf) #This is the correct way from version 2.6

2. Convert full cti to full yaml

I succeeded in doing this, but I needed to change thermo to “None” in the full cti file.

I verified that example\_runfile.py (still the cti way) works even with thermo = None

2.1 made function “create\_full\_yaml” as hardcoded to point to the converted yaml file. I tested that it gave back a yaml string.

Tried using create\_yaml\_and\_cantera\_phases (which depends on create\_full\_yaml), and was getting an error. Trying to use “ceO2\_yaml\_full.yaml” by hardcoding to see if that gets further.

reactions\_parameters\_array = "ceO2\_input\_reactions\_parameters.csv"

canteraSimulate.create\_yaml\_and\_cantera\_phases('ceO2', reactions\_parameters\_array, ceO2\_input\_simulation\_settings)

For making cantera phases, I was having problems with yaml import. Initially imported the yaml using the “infile” argument, then switched to using the yaml\_string using the “yaml” argument.

Needed to give a finite pressure because gas phase can’t have zero density. Made the pressure 1E-20. (I’m not sure what units that was in. – the cti has a comment of Pascal by default.

Should add comments into the yaml file also.

2.2 Now need to try to make a simulation using

canteraSimulate.create\_yaml\_and\_SimulatePFRorTPRwithCantera

Was able to run it.

Will slowly check the ability to run example 1, example 2, example 3, example 4, and example 5 with the cti way vs the yaml way. Also need to be careful to restart the kernel each time because I found out it can cause problems in some cases if I am doing this strange thing.

Example 1 check:

Ran the regular simulationDriver directory only to example 1, then copied the relevant files over to the YAML directory so there is a 'base case' to compare to.

* + - * + The cti case in the YAML directory now shows no changes for example 1. This is despite a pressure difference of 1E-20 vs 0.0. It’s a bit concerning that I don’t even see rounding errors.
        + Now using the YAML case in the yaml directory. Again, not even rounding errors.

- Deleted the relevant files (to make sure they are really getting created) and ran again, still no changes. So now I have confidence things are working okay despite no visual rounding error.

Example 2 check:

Ran the regular directory to example 2 and then copied the files over into the YAML directory.

🡪 checking the CTI way shows no changes.

🡪 Checking the yaml way shows changes that don’t look negligible to me.

Will try making some plots in the directory of acetaldehyde gas from ceO2\_output\_rates\_gas.csv.

There is a distinct difference now: the CTI way is showing two peaks, and the YAML way is showing 1 peak. I suspect that the YAML way is not modifying parameters because it is hardcoded to just read the full yaml file.

I have confirmed that yaml Example 1 is giving the same output as yaml example 2. This makes sense because the array “modified\_reactions\_parameters\_array” is not being used right now – currently the YAML file (and yaml\_string) are never being modified, always only being read from file.