Exawind Setup Automation Needs: Interviews

Sydney Richardson

June 2023

1 Feedback Summary

Main Problems	Summary
Restarts	Restart files will need some small changes between runs, usually something like calculations from the initial precursor files, want to automatically update the restart files from the precursors (especially for the turbine files).
OpenFAST	Brought up by multiple different people as tedious. Need different times steps between OpenFAST and AMR wind. Want to automatically "match" the parameters between OpenFAST and other necessary input files.
Input Files	Need many different input files for many turbines and for the different codes being merged together by ExaWind. Want some kind of automated process of applying the parameters that won't change to all the files.
Errors	Having some kind of "validate" function to check if there is any basic errors in the inputs files you created that makes it obvious when there is a basic issue. You know the errors before trying to submit files.
GUI	Generally users and some developers do not like GUIs. It is best to just be able to directly access the script. Creating many say turbine files can be tedious with a GUI. Ideally create a function or class that could be called via terminal to create input files.
Parameters	It would be convenient for new users/users who don't need to understand the physics behind the simulation to have some sort of short explanation for what the input parameters are/units of said parameters. Perhaps have the ability to choose from a library of values that a simulation would typically use.
Documentation	Very important to document everything. Clearly document the code and organize it in a manner that others can read the documentation and easily understand how things work.

Future Problems	Summary
Visualization	GUI can provide easy visualization, but again within a class or a function "visualize" could be created to easy get a visualization of your turbine layout based on the input positions the user has provided. In talking with users seem like most people have their own scripts for visualization.
Compiling	In order to compile ExaWind currently one must download a library. It would be more convenient to be able just to compile from terminal
Post-processing	Mentioned several times in that it would be convenient and helpful to have an automatic post-processing function/display of some sorts. Priority goes to automating the inputs but if time allows look at post-processing.
Crashing	When trying to run the simulation, sometimes after one or two timesteps the simulation will crash but the slurm job will keep running and can't get it to stop.
Time Step	The hybrid solver makes declaring a time step difficult as it requires a constant time step but in some cases it's preferable to not have a constant time step. Ideally one could vary the time step via the codes communicating with one another ie choose from the smallest of the time steps presented by the different codes.
Meshes	Would be helpful to have some assistance in automatically generating meshes something along the lines of the error checker mentioned early but applied to meshes.
Inflow Conditions	A more straightforward way of going from ABL to a blade resolved simulation. How does one generate the inflow conditions for AMR wind?

2 Appendix: stakeholder interviews

2.1 Experienced researcher very familiar with AMReX and Nalu-Wind

- Lawrence's GUI
- Highly graphical
- Has to be scriptable somehow in the backend
  - Always a way to access the code
  - Dump the state of it to reload in real quick or passing onto someone else
- Python package:
  - Package in a way that is easy for others to pick up
  - Poetry
    - Pyproject.toml list dependencies and use to great environment
    - Set up the package
- Python extensions
  - Black for code formatting
  - Flake8
  - Isort
- Some Python stuff
  - State of stuff, import arguments, stick to a standard python format yaml or toml
  - One cool thing to enforce: use pydantic which is a way of structuring classes to conform classes to same schema
- Decluttering Lawrence's code
  - Sit down and do first: document a toml/yaml file which is your farm, turbines locations etc, tool would read that file, output all the amr wind nalu wind input parameters, gui interact with toml file and edit that while also spitting out the input files
  - Generate input files from this yaml or toml file and that would be the thing, choice of either direct file editing or gui
  - That input file that defines your farm
  - Do the top five or something and see if you can go through that
- Functionalities

  - The chain of input files form precursor simulation to generating all the nalu wind files then the bug thing is restarts, restarts are a pain for nalu wind
  - All the turbine input files
  - Chaining these jobs
  - Restarts, change the input files to make sure they output correctly
  - How to go from a background flow simulation to how to put turbines in a simulation

2.2 Experienced researcher familiar with AMR-Wind, extensive user but not a core developer

- AMR wind: documentation could be improved, looking to see how I enable this functionality, inputs: a lot you can do with AMR wind, most people want to just get up and running, people want quick gratification
- They've been setting up very complex systems, layout of a ton of turbines with different locations, don't want to manually specify things:
  - Take data you have and automatically put into input files with minimum errors possible
  - Documentation, some examples: docs, look through different tutorials, run through using the python script/Jupyter notebook
  - How do you want to interact with automation
    * Run through and setup case, python makes it easy to do automation
- AWAKEN summit setup
  - Very tedious through GUI
  - Want to setup a case with one turbine for different conditions but want to run through different settings
- So much you can do with AMR wind—start with small portion of it
  - Think about what most people want to do
- Main headaches:
  - Interacts with openFAST
    * That is completely separate code with its own versions with different kinds of input files
    * Have to keep up with changes in both
    * Have to get files and models from separate codes and make sure it works with openFAST

  - Visualizing where things are

- Helpful coding tips
  - Tkinter
  - He structured his code with a different library to handle all the tkinter things (yaml files)
  - Split your code to deal with what all the different inputs mean, keep input variables separate from main code
  - All the configuration is separated out for plodding, handling how things are generated, etc

- Hybrid solver
  - Two different codes with two different inputs but keep straight what are overlapping inputs/need to be kept separate and generate multiple different input files going into this different codes
  - Know which parameters remain the same

- Paul Fleming doing numerical flow simulations

- "no need to reinvent the wheel"

- Define automation:
  - Doing it as easy as you can or creating more complex simulations?
  - Two end users: someone who doesn't work in our group (if someone just wants to setup a one turbine simulation) generate base input file; restart mechanism and just make it easier for large number of turbines

- Automatic meshes?

- Write documentation!!
  - About features
  - Try as much as you can describing what all the inputs are

2.3 Experienced researcher intimately familiar with the entire code-base, but not an extensive user

- OpenFAST files (mentioned again a headache!)
  - Not much of an opportunity to share files and that it has to be regenerated for every turbine
  - Automation in regards to the controllers
  - Naming convention can autopopulate all the autoFAST turbines and then be able to integrate that into CFD

- Lawrence GUI
- Good exercise:
  - Go through and set up one of these problems and everywhere you have to copy and paste circle that to update
  - Some of these things explore doing this in memory—only regenerate the portions of the file that need to change and reload from memory
  - As few files sitting on the file system as possible
- Coding tips:
  - Object oriented programming reading some articles on that
  - Use things like numpy and other things
  - Organization:
    - Comfort with classes and how to use them how to extract functions and all that
- Reusing Lawrence's code
  - What is the difference between an openFAST definition for openFAST wind versus a turbine definition from the driver
  - NALU: input files that are yaml, load into memory as yaml representation then injected into NALU and parsed
    - I can do something similar with frontend, drive everything in yaml, change in python with yaml interface, generating files and circle back to in memory things
    - Fundamental object of representing a simulation if you can create that abstraction it should be able to plug in pretty well to the tool Lawrence has
- openFAST and precursor to inflow/outflow will have impact to most projects
  - Blade resolved stuff beneficial but to smaller group
- create a function that can be callable?
- Write everything down on a script like you would on your homework
- NREL 5 mw baseline and a secondary set of parameters at these locations at these locations and this yaw etc and it generates the info you need
- Fancy—xml file that you could put into solver

2.4 Has been using AMR-Wind for a while, but not a developer

- Precursor to input file that should be automatic
- Read what will go into input
- Ross needs to be on specific version
- Error displays like before running and you know which files should be running there
  - Cross check commit and that it will at least run
- Account on Eagle
- AMR wind: Lawrence developed GUI
  - Work around behind GUI by using elements of GUI
  - No GUIs!
  - Direct access to script
  - Essentially a class maybe a function would be ideal
    - Already have parameters
    - You can just pass these things in a function and a nice input file
    - Read precursor file, modify what needs to be modified then output an input file
  - Ganesh doesn't like notebooks but Regis does
- Visualization: reading input file into a class, you can have a visualization in your vlass because you have all the information there
- Something to go into check, function read precursor modify what I need
- He included a call to call AMR wind
- Lawrence's tool automatically created fields around the blades, the flows automatically locate to your turbines, this was very nice
  - Refinement regions
  - I don't want to specify manually where each turbine what is refinement you want and automatically create refinement regions
  - Following the wind direction
  - But if your turbine is yawed that also changes refinement
    - Function of yaw and wind direction
    - He created a fake wind direction of wind to have it correct for the yaw
- Someone using AMR for fastFARM
  - If hear enough people talking about it he can give details about how to plug in

2.5 New researcher who has used AMR-Wind for some research projects

- AMR wind mostly for ABL precursor simulations by varying stability using surface heat flux
  - openFAST
- Pain point: when a restart was required specifically with the turbine simulations but also sometimes with the AML wind, but for the turbine simulations it's a pain
  - Ganesh wrote a script to read an output calculate some of the things you need and output what you need for the restart for the next input file
  - The timestep AMR wind uses is sometimes different from time step you use in openFAST so you have to factor in these two different time step sizes
  - BodyForce.magnitude and incflo.velocity, and a few moer were calculated based on the precursor run, ABL.wf velocuty, vmag, theta which are generated using a different precursor script
    - Automatically make that happen
- Another point of automation:
  - Defining where they want the sampling planes
  - Typically someone would say I want n planes (5) spread throughout the domain
  - Or I want one vertical sampling plane at hub height but will ultimately generate the sampling plane location data you would need to input into the simulation
- No GUI! Not most efficient way to run multiple things
- Having a couple example runs for documentation just to get something that runs
  - Some tests essentially
  - For me to just give it to someone else who doesn't use exawind could you just try and run something as a test case ssentially
- Large eddy simulation software after creating Input file you could use a command like validate if there was some base level errors
- If everything can just be ran straight from terminal

2.6 Relatively new to the code-base, but has used Nalu-Wind extensively

- How Shreyas uses:
  - AMR wind and NALU wind hybrid solver
  - AMR wind
    - Time step computation can either be fixed or tsl number use this time step if it greater than 0 if 0 use something else
    - Having the time.cfl doesn't make sense either have supply cfl number or suppky the time step
- Automating the restart file
- Adaptative mesh refinement if it is static whether you need to specify this text file refinement def, cartboxrefinement that is default is that necessary
- Linear Solver:
  - Doesn't change most of things
  - May want to include the hypre solver whether user wants
  - Same for NALU wind you have many lines for linear solver you don't change
    - Realms: you don't really toucg this stuff either
    - All those tolerances
- Boundary conditions need to be edited
- Hybrid factor: don't touch at all
- alpha upw factor actually used
- GUI is useful, as a new user its very friendly to new users, as an expert user wants the file
- One turbine simulations mostly
- Where you specify NALU and AMR wind input file that could be automated too but user would have to specify file names
- Postprocessing: Paraview, opening the paraview GUI, you need to get a job first then there is a bunch of things that need to be entered before you can open paraview
  - Once you have a job you need to request the specific server and number of proceces
  - Super long steps to open up paraview

2.7 Moderate user of AMR-Wind, not a developer

- AMR wind mostly pretty good in terms of inputs, AMR wind generally has pretty good inputs
- Did not like NALU wind the indentation for the yaml files and the lists within lists all very convoluted
- Specifying the density in more than one place you forget to change it in one so to have consistency in density
- The code should be able to handle it from one place
- Inflow.density and BC xflow.density and one more location
- Units and its annoying when things don't have units things based on time step versus actual simulated time
- Output frequency 1 conufsing
- Compiling the exawind
  - There's a library just to compile you have to download it
  - Would be easier to access all from terminal
- Preference for GUI or functions?
  - Big fan of examples and changing those manually
  - Was given that feedback for NALU wind repository
- Error check some kind of prediction would be great
  - Hard to do without just running the code
  - End up just running the code anyways
- Just one turbine running but used for multiple turbine tests
- Visualizing data: he has his own script
  - Turbine data from openFAST
  - Actual AMR wind file has own scripts
  - Paraview painful for only hybrid solver
- Likes the plt files to look at outputs
- Ganesh: running the blade result simulations even for just one turbine in flow, very tedious
  - Ideally a way to give it a turbine and a way to click run given a domain size just click run and get what you need out

- Picking the number of processors and the number of grid points
  - AMR wind has some special requirements about blocks divisible by 8 and AMR wind won't do it because it wants 128
  - Blocking ratio of something about AMR wind
  - Have to match that to number of processers
  - 36 cores per node but AMR wind needs 32 cores you can't ask for 1 node you should just run on 32 cores
- Most of the things only run on Eagle anyways
- Run the code to get an idea of things mentioned

2.8 A linear solver expert, largely treats the CFD solvers as a black box

- Primary concern is getting the entire stack running on AMD hardware particularly frontier
- Most of difficulties are within the linear solvers
- But most issues ironed out
- Challenge for linear solver user: being able to build the necessary files, so eliminating that would be great, being able to populate the simulation for an arbitrary number of turbines
- One input file, additional meta data could be provided in data
- Not managing a bunch of files or having to generate them but handled in the driver source, NALU file would have these generic files then you want to populate it across however many turbines
- Manage input dex to help it cleaner
- Base NALU file and just give that info in yaml format to exawind driver
  - Read in generic file and intercept at important times information you want to vary
  - This wouldn't need additional files
  - One generic file to manage and have code vary it
  - As a starter generate the n files then move to something more general
- Minimize number of files needing to manage
- Wouldn't use a GUI workflow is text files
- Does a lot of scripting based on how many turbines he needs to run he writes job launching scripts to automate that but automate input files

- Some kind of visualization function just a simple thing to know they are in a reasonable position esp in relation to meshing of AMR levels and a plot of that would be nice some cross sections of each turbine would be nice
- Base file creation
- Validation: "crude validation" relating to the mesh
  - Validate in response to AMR file
  - Some type of go in and make sure the turbines are in bounds
  - Throw some kind of warning
  - Are the turbines going to hit each other
- Plot function that generates some pngs to easily view lowest priority

2.9 Core developer of AMR-Wind

- A lot of inputs in a simulation file
- Unclear when you are getting an input file which ones actually matter
- Some kind of sense of what the inputs are for and why they are important
- When two codes are working together:
  - Some things should be treated more directly
  - Restart files but that's not specified in a single file
  - When its looking for the same thing only need that once
- Time step: essential to match in both files
- Likes the idea of base file
- The way the hybrid solver works is it requires there is a constant time step in each code but in some simulations its just better not to have a constant time step
  - It would be nice to be able to vary the time step and the codes could kind of communicate where each code could choose which time step it wants but then choose from smallest of the time steps presented
- Visualization: it can be a little tedious because you have different sets of files and you have to put it in a visualization software
  - Tedious to put that together
  - Potentially something to do with keeping track of files keeping compatible

- Basic errors:
  - For AMR wind there is a built in tool that you fcompare directly
  - Visualization of differences
- Boundary conditions
  - More work on input files side
  - Doesn't know exactly what the options are for the boundaries
  - For certain boundary conditions you need values to fulfill the boundary condition with
  - Depending on the type of boundary condition
    - What do I need to input depending on the boundary condition
    - What are the options for changing
    - Additional arguments
- An input file checker where you would instead of submitting the simulation you would use an executable to check input file to make sure that it's not going to fail when you try to run the simulation

2.10 Core developer of Nalu-Wind, and extensive user of the entire code-base

- Ganesh wrote a script to read an output calculate some of the things you need and output what you need for the restart for the next input file
- If can't accomplish all documenting everything very thoroughly
- Implement on of those things in a decent manner to set an example
- First:
  - ABL—AMR wind, precursor simulation then to turbine setup, actuator disk/line, Blade result simulations
  - Just blade result simulations
    - Here is the turbine
    - Here is the diameter
    - Very common use is ontology—see git link for how file should be set up
    - Generating a mesh from this yaml file
    - Converting it into a full blown mesh, how big should my domain how big should refinment zone be—essentially setting up AMR wind setup
  - Cross sections of these blades
    - OpenFAST: airfoils

        * Yamls cd, cl, cm, andle of attack; some attempt of automation
    - OpenFAST
        * Partially nice any regression tests
        * Biggest problem is entering version mismatch
        * Rosco controller has to upgrade to some specific version and this all has to work with AMR wind
        * Ensuring version consistency, if there is someway to say if you are using this version
        * OpenFAST coupled to AMR wind for actuator length two specific inputs need to be changed and that's it but most people don't know/forgot
    - Same thing for blade result simulations two specific things need to be changed so have both pre processing scripts such that if you have a set of openFAST things that work alone how to convert it to new desired files
        * Don't change openFAST inputs
        * If you need to work with toolbox from openfast see git link can use this tool to quickly create a bunch of simulation files
        * Change just the windspeed, rpm, and pitch it will change everything it will work for openFAST but not with AMR wind

- Focus on documentation and pick one that would have the biggest impact
- As far as AMR wind is concerned Lawrence's GUI
    - Not a fan of the GUIs
    - What would be nice is the separate the GUI aspect from the non GUI aspects
    - Doing the same things as the GUI just with input files instead
    - Like about not using GUIs: automation possible to run a bunch of GUIs
        * Something not copied over you need the automation to make sure this is actually correct
        * Add to GUI enable functionality through GUI and input files
- Prioritize AMR wind first, one turbine a case setup and you want 16 turbines still do it manually but if project doesn't have other needs automate AMR wind
- Ideally have a simpler input file and not make the user try to copy a sample and edit for needs is not the prefereed workflow, have only stuff users will need in input file, and can still modify the file after
- Simple, clean, user can make less mistakes

- Targeting lay users who don't really user MAR wind I have this hub height, turbine height, boundary layer height, they don't care, give them options from a library
  - Free to modify after but have something to start
- Once finish running precursor these are my turbine locations that are otherwise identical
  - You have done precursors so you generate automatically the refinement zone
  - 0 north, 90 east, 180 south, 270 west
  - Have north east south etc instead of value
  - How to handle terrain: somebody goes to google maps finds lat and long, and you just select something there and your wind farm is there

2.11 Moderately uses AMR-Wind, not a developer

- better error messages when you mess up an input. I've literally lost a week of work to debugging AMR-Wind, and I apparently just missed one letter in my config file. Lawrence Cheung has already put together an interface to help automate AMR-Wind configs, but I know myself and many others find it much quicker to write configs manually
- He does like the GUI
  - A little buggy but with lots of turbines and lots of meshes the gui is good for lot
- Input files
  - It would be great if it fixed itself
- Body forcing
  - Ganesh gave script so it automatically updates the output file
  - So he did it and has that file
  - Calc_inflow_outflow_stats
  - He was going to push that this week
- Errors
  - Body forcing parameter: vmag or mag type in single variable name, check that input names are okay
  - Big one: simulations will crash one minute or say into a run, but the slurm job won't stop it will just keep going, if I'm running a huge simulation and it keeps going if you doing xyz it should fix it but it doesn't

  - Inflow outflow simulations: need to include an average temperature file avg_txt or theta or something if you forget to put it in the same directory / config file AMR wind will just crash you need a reminder for this one

- He's updating some user things
- Examples of things working
- We don't need a new GUI

2.12 Familiar with maintaining the code-base on different architectures, but treats the CFD solvers and dependent libraries as a black box

- Get the setup and end up running it on a new machine
- Fill in the holes being able to build it and portably run things
- Fundamental stuff that needs to happen for it to work
- He can manipulate the files to some extent but the NALU part will probably be the harder part to do
- https://github.com/jrood-nrel/job-scholar
  - Read in a yaml script to generate scaling studies very easily
  - Submit jobs python script
- Error checking as best as you can before actually submitting the script
  - Location of file
  - Checks that executable exists
  - Check input files exist
  - Reading in the yaml files
- Multiple NALU files
  - Having AMR wind files
  - Instances of overlapping
  - Try not to assume AMR wind always has a single input file have it treated similar to the NALU files
- Supposed to be doing complex terrain with the towers pointing in certain directions
- Not even just for iterating on input files for different turbines but NALU instances for like the ground

  - Another assumption to be aware of
- Offshore stuff gets even more complicated
- OpenFAST problems
  - The full path to everything but you have to deal with those too
- Restart:
  - They will do like job chaining in slurm as soon as you submit a job once that job finishes you want to automatically have the next job queued so it would be great to automate the restart abilities
  - Checking AMR and NALU at the same time step and look for the latest one
  - Basically, babysitting slurm to submit jobs
- One master input file would be nice to submit
- The mesh:
  - Different types of turbines
  - Mesh for complex terrain
  - Restart files for turbulent ABL
  - A lot of binary files to manage at some point
- A single case essentially belongs in one file, and you just transfer that case around
- Another good thing to realize is that we aren't just going to be on eagle so it's not easy in any sort of framework
  - Some of this stuff should be trivially portable but try and make it portable
  - HPC: problem is mostly dealing with the newest compiler
  - Found numerous complier bugs
  - Bash portability (ask Phil about spack portability) scripting in bash vs zsh
  - Worry in generating bash type scripts
- The whole python 2 vs 3 thing and having things up to date
  - If you are going to rely on NumPy or pandas or something to generate/organize structures that stuff gets to be difficult to make sure that the person has that on the machine
  - Correct version checker