Examples using the analysis actions

Xaxis: Antonella, Andreas

- **Notes:**
  A warning to advice the users that every symbolic or numerical variable don't need any space would be useful, like ex: `var var_1 NE.phi+45`
  You would need to use brackets, implementing would be too difficult. Might it be possible to implement some form of warning
- If we set `model.NI.phi = var_1`, and then we scan on NE.phi using Xaxis(), a message that advises the users that the Xaxis() is changing the content of NI.phi too would be useful.

- the `math` command needs better examples in the documentations, so the user knows how to hold a record of the different variables values. Example for the math detector to show how to keep track of the symbolic relationship.
- Overview of symbolic relationships in the model:

```
for p in model.all_parameters:
    if p.is_symbolic:
        print(p.name, p)
```

XNaxis:

- Notes: When testing it for 5 axes, we get into memory errors if steps are too many. Maybe a warning could be implemented. The parameters that are given have to be divisible by 5, but it does not tell which one is missing if one is.
- ```xnaxis(param1, mode1, start1, stop1, step1,(param2, mode2, start2, stop2, step2, 3 more times)` . Should this be sliced into or grouped into parameters, modes, start points, stop points and steps?
- Example with N (5) axes and one plotting method in 3D?
- Note that this should not be used for studies of sensing matrices.
- Should there be some utility function for easily accessing the output array by parameter name?
- Try and catch memoryerrors on initialization and print a more useful message
- Tasks:
- Issues:

TemporaryParameters – Bas, Riccardo

- **Tasks:** We tested with temporary_parameters() and TemporaryParameters. The goal was to verify whether all parameters linked by symbolic values were correctly reset. We were unable to produce a case where parameters were reset to an incorrect value or where the symbolic connection between parameters was lost. Both methods for modifying parameters proved effective.
  The code can be found at this link:
  https://gitlab.nikhef.nl/finesse/virgo_simulations/-/blob/master/workshop_Nov2024/Wednesday/test_temporary.ipynb?ref_type=heads
- **Issues:** We also attempted to use the Temporary action but found its purpose unclear and were unable to produce a working example. The TemporaryParameters action was added to the Finesse codebase as an improved version of Temporary, based on the commit message introducing TemporaryParameters. Since Temporary appears to be an outdated version of TemporaryParameters, it should be removed, deprecated, or at least marked with a flag indicating it is not recommended for use.
  The breaking example can be found at this link:
  https://gitlab.nikhef.nl/finesse/virgo_simulations/-/blob/master/workshop_Nov2024/Wednesday/crash_temporary.ipynb?ref_type=heads

  Should probably delete the Temporary action.
  TemporaryParamters action should have an example

BeamTracing: PropagateBeam - Anna, Michael, Mischa

Relevant notebooks: propagate_beam_ACG.ipynb;

- When we cannot beamtrace, e.g when there is no “gauss” or “cav” it correctly gives an error but might be good to point at docs how to fix it.
- In models with all lengths 0m, the error by `propagate_beam(...)`**`.plot()`** is unintuitive (plot labeling of components between 0m spaces is also unreadable)
  a. beamtrace.py line 1297
  b. beamtrace.py line 1114
  c. beamtrace.py line 1002

c. is the one giving a division by zero.
Note that printing the `PropagationSolution` gives the correct result.

- 4 syntax options for propagate_beam, and significant but decentralized/incoherent documentation:
    a. `ps = finesse.tracing.tools.propagate_beam(from_node=test.L0.p1.o, to_node=test.n0.p1.i)`
        - base function that the others call
        - doesn't accept strings
        - not implicitly connected to a model so subject to more general errors: e.g. "ValueError: n0.p1 and L0.p1 are from different models"
          Clear but with a stacktrace
    b. `ps = test.propagate_beam(from_node=test.L0.p1.o, to_node=test.n0.p1.i)`
        - typical use currently
        - calls (a)
        - (b)-(d) can specify the nodes as string such as "L0.p1"
        - Compact but obscures that this is an action
    c. `ps = test.run(fac.beam.PropagateBeam(from_node=test.L0.p1.o, to_node=test.n0.p1.i))`
        - Action
        - calls (b)
        - useful if you ever want to run as part of a series, but this is rare.
        - documentation unclear (e.g. what arguments to parse to the action), and sends you through several hoops (c->b->a) before pointing to the funcs above
    d. `ps = test.run("propagate_beam(from_node=L0.p1.o, to_node=n0.p1.i)")`
        - katscript version of (c)
- Error when incorrect `i/o` nodes are provided could be clarified:
  "NetworkXNoPath: No path between m2.p1.i and L0.p1.o."
  ok-ish but with long stacktrace and could be good to say: "did you mean…"
  Note: i and o are optional, but it is not sure this always goes right also in complicated cases with a via.
- For the actions (c and d): when we try specifying two strings (assuming this might be the from/to), the first is actually setting the "`name`" parameter for PropagateBeam (which defaults to "propagation"), and therefore leads to a confusing error about expecting 0 or 1 parameters (the from_node and to_node parameters need to be specified as named parameters):
  "TypeError: PropagateBeam.__init__() takes from 1 to 2 positional arguments but 3 were given

NoiseProjection - Miron, Anna

Relevant notebooks: noise_projection_ACG.ipynb;

- There is zero documentation for this action, the only thing we could find is https://gitlab.com/ifosim/finesse/finesse3/-/issues/330 (using very outdated syntax which we got working)
- If you add a readout component with just specifying a Port to connect to, not a Node, the readout will be added but not connected to anything
- It is not clear at all that the NoiseProjection action only makes sense when also adding ClassicalNoise elements (or other noise types) to your model
- It is not clear that the ASD parameter of the ClassicalNoise element should scale with fsig
- It is not clear that the NoiseProjection action is using a frequencyresponse under the hood, so it will not work when specifying optical nodes either to the NoiseProjection action or the ClassicalNoise component
- Keep forgetting that `output_detectors=True` for readouts adds a detector with underscore name like 'readout_DC' or 'readout_I'

RunLocks: Antonella , Michael

- If the model is already tuned by default setup the RunLocks() will display a blank plot that may be confusing for users—> add a message to advise the users that the model is already perfect?
- RunLocks() doesn't collect the output of the PDs in the model, it would be nice to have at least the output of the PDs as default solution in the `sol = model.run(RunLocks(method='...'))`

PseudoLockCavity:

- **Notes:**
  - Simple action to lock cavity using eigenmodes. Cannot and should not work with more complicated examples.
  - PseudoLockDRFPMI exists but is rigidly LIGO specific and tagged as WIP.
- **Issues:**
  - Feedback can be any node, including laser power, which doesn't make sense.
  - Providing a non-existent cavity name throws unfriendly `KeyError` exception
- **Tasks:**
  - Check parameter type for feedback
  - Fail gracefully if cavity does not exist
  - Warn for more complicated examples?

Minimize/Maximize - Mischa:

- **Notes:**
    - Simple methods to optimize parameters but optimizing is obviously a hard and not always trivial problem. It's difficult to say how reliable the results are
- **Tasks:**
    - Improve documentation, see below
    - Maybe add a katscript optimize action?
- **Issues:**
    - Documentation for Maximize and Minimize is misleading/confusing:
        - As it includes the documentation from Optimize it mentions a parameter that cannot be used: `kind` (since it's `max` for `Maximize` and `min` for `Minimize`).
        - Additionally the beginning is confusing since it says *both* that it "maximizes" (or minimizes) and "optimizes by either maximizing or minimizing".
        - It points to `scipy.optimize.minimize` because it uses minimize also for the `Maximize` action, but this can be confusing for readers. Docs should explain that the optimization is done using the SciPy minimize function
        - the name parameter of Optimize defaults to "optimize" but the docstring says it defaults to "maximize". It is also unclear what the purpose of "name" is in particular in relation to the "kind" parameter (which is either "max" or "min")
        - Related bug: code checks whether kind is "max" with an if/else. So anything other than "max" uses the code for "min" without error (optimisation.py lines 229-232)
    - The Optimize action has no katscript version, unlike minimize and maximize
    - Unlike for example the sweep/xaxis actions, the different optimization actions change the model parameters. That is not clear from the documentation (even though perhaps expected).
    - The examples in Maximize and specially Minimize are rather trivial.

General comments:

Exceptions:
- Finesse exception, better printing in ipython environment (avoid horizontal scrolling)

- Finesse exceptions should include name of command that triggered them, i.e. if from an Actions it should include action name?

Features:
- utility (Python) function that shows all symbolic references to a model parameter?

Manual:
- API results should come later or have a link to non API documentation, could we sort the results somehow?
- Give example on how to use the 'math' detector to add variables to a solution object

Building:
- make commands should not run conda update, or any update. The documentation should be adapted to say how/when updating conda might be useful

- Current instructions could like to channel mixing between conda and conda-forge. We should go back to advise something like this, even though it looks more complicated:

Use these instructions:

```
conda create -n finesse3
conda config --env --add channels conda-forge
conda config --env --set channel_priority strict
conda activate finesse3
conda env update --file environment.yml
```