**Restructuring GenX and DOLPHYN**

Ruaridh Macdonald Last edited: June 20th, 2023

This is an initial sketch of a revised structure for GenX and DOLPHYN (henceforth enX[[1]](#footnote-1)). While enX works well right now, it is increasingly difficult to add new features because of the complex web of constraints and expressions which tie the different pieces together. These problems are compounded when working across multiple energy sectors.

The main goal of this rewrite is extensibility, i.e. to restructure enX so that it is easy to add new technologies, policies, sectors and features. This will help avoid the need for future rewrites and make enX better able to model a wide variety of macro energy systems. By making it simple to adapt or add features, we will make it easier to tailor enX to a given problem and hence it will also be a more flexible tool.

In restructuring enX, there are three profiles we should keep in mind:

1. *Users* – people who want to use enX as-is to answer their questions. They may or may not be able to provide their own data and post-processing tools.
2. *Modelers* – people who are comfortable adding new resources, policies, sectors, etc. using the tools and features which exist in enX.
3. *Developers ­*– people who capable of adding new features to enX and improving or adapting the core infrastructure.

There are not hard boundaries between these profiles and any given person is likely to straddle two. However, as we restructure enX we will likely have to make choices which improve the experience of one profile over the others. For example, reducing the memory required by a feature will improve the experience of users and modelers but likely require the feature to be more complex, making future development more challenging.

When trade-offs are required, I would favour improving the experience of users and modelers over developers. Those two groups are potentially much large and enX is unlikely to be used (and be worthwhile to continue developing) if it is not easy to use and performant. This choice will increase the barrier to entry for developers, but that is the group most willing to invest time in understanding and overcoming difficulties.

To make enX more extensible, I propose restructuring it using three core ideas:

1. enX should compose technologies, policies, etc. from base features and use abstract structs to control which functions can be applied to different resources. For example, a storage resource would be defined by calling functions which add an energy capacity, (a)symmetric discharge capacity, state of charge variable, ramping limits, etc. This will improve code re-use and reduce complexity.
2. enX should construct JuMP model(s) by looping through inputs rather than types of resource, policies, etc. This will make it easier to adjust how each input is handled.
3. enX should have a sector-agnostic core infrastructure to allow the greatest range of macro energy system problems to be addressed and to avoid the needs of any one sector over-determining how enX functions.

To motivate the first two ideas, let’s consider the current process of adding a new storage technology which has constant energy leakage, rather than it being state-of-charge dependent. If I wanted to re-use the existing storage variables, constraints and expressions, I would have to run storage\_all!() and then replace the energy balance constraint. This will break the output-writing process for all storage technologies unless I carefully reassign the constraint. Alternatively, I could separate the energy balance constraint from storage\_all!(), making all other forms of storage more complicated. Finally, I could create a new technology by copy-pasting some code and adding more input files, resource files and flags in Generators\_data.csv.

Under the current structure, I can’t pick and choose which features to apply with enough granularity. The current structure is fine in many respects, but the functions involved should add a small number of variables, constraints, etc.; not define each technology. Additionally, because enX currently loops through resource and policy types and applies them to multiple inputs at the same time (e.g. defining the energy balance for all storage technologies at once) it is difficult to customize resources and re-use existing code without creating new variations of a technology or copy-pasting functions – which in turn makes it harder to maintain the codebase.

Under the new enX structure, defining a new technology should only involve creating a new Julia file describing how to load the relevant data, what features to add to the JuMP model, and any specific outputs which should be created. Figure 1 shows a sketch of how this would work.

1. The user provides input data for a resource including the resource type, “example” here.
2. The inputs are loaded into a temporary struct. This could just be a dictionary.
3. enX then looks for a resource file matching the resource type: Example.jl, in the enX directory and uses the construct\_\*() function to build the relevant struct. In this case construct\_example() is used to build an Example struct.
4. enX uses the data in the Example struct and the functions defined in Example.jl to add the resource to the JuMP model. We can take advantage of multiple dispatch so that a general sequence of functions can be used for different resources, e.g. build(Example).
5. enX then repeats the process for each input, including other “Example” resources in different zones.
6. After the model has been solved, enX writes general outputs for all resources and resource-specific outputs defined in Example.jl

Because enX will now only be looping over the inputs, adding a new technology does not require the generate\_model() function (or equivalent) to be changed. All that is needed is a new resource file. There will be some connections between resources and other parts of enX. For example, resource definitions will have to include the data required by emissions policies or constraints, so writing a new resource file will have to take other parts of enX into account. This structure can also be used for policies or in non-electricity sectors. A similar concept could also be used to define zones.

A picture containing screenshot, black, font, text

Description automatically generated

Figure 1. Example of loading resource and adding it to the JuMP model.

To avoid rewriting resource and policy features from scratch for each technology, we can again use multiple dispatch to encourage code re-use. We can define features such as unit commitment constraints, time-dependent availability, etc. for an abstract resource and then define other resources as subtypes of that abstract resource. Whenever a resource needs a different version of a feature, then it can be redefined for that subtype. Other resources can in turn be defined as subtypes of that resource. If a feature becomes ubiquitous and subtyping becomes difficult, it can be promoted to its own function with a different name.

Unlike an object-oriented language, these subtypes will not inherit parameters from each other, only the ability to be used as arguments in the same functions. Care will need to be taken to ensure the correct parameters and data are defined for each resource.

A picture containing text, businesscard, screenshot, font

Description automatically generated

Figure 2. Example of using structs and subtypes to allow for code re-use.

It is harder to justify whether to have the electricity sector be at the core of enX or have a core infrastructure which is sector agnostic. My main argument for a sector agnostic infrastructure so is that it will be no more work to make it sector agnostic and may make enX more flexible. My experience integrating GenX and DOLPHYN suggests me that what is suitable for the electricity sector can create difficulties for other sectors or for integrating them together. I think it is inevitable that putting one sector above others will accidentally cause us to favour it.

Further notes to add:

* How adding zones and policies could work
* Setting up sectors to have the option to be price-takers or use optimized results from another sector
* The potential for decomposing optimizations across sectors

1. Though there are energy and software companies called enX and nX, so we may want to reconsider the name to avoid being lost in Google results [↑](#footnote-ref-1)