User's Guide

Reliability Calculation Tool and Excel User Interface

Big Ladder Software

October 8, 2020

Contents

1 Introduction **2**

2 Simulation Overview **2**

3 Concept Overview **4**
3.1 Flows . . . 4
3.2 Components and Ports . . . 4
3.3 Component Types . . . 5
3.3.1 Component Type: Load . . . 5
3.3.2 Component Type: Source . . . 5
3.3.3 Component Type: Uncontrolled Source . . . 5
3.3.4 Component Type: Converter . . . 5
3.3.5 Component Type: Storage . . . 6
3.3.6 Component Type: Pass-Through . . . 6
3.3.7 Component Type: Muxer . . . 6
3.3.8 Component Type: Mover . . . 6
3.4 Networks and Connections . . . 7
3.5 Scenarios . . . 7
3.6 Reliability: Failure Modes and Statistical Distributions . . . 7
3.7 Resilience: Intensities (Damage Metrics) and Fragility Curves . . . 8

4 Input File Format **8**

5 Output Metrics **15**

6 Command-Line Tool **16**
6.1 `erin` . . . 16
6.2 `erin_multi` . . . 17
6.3 `erin_graph` . . . 18

7 Microsoft Excel User Interface **19**
7.1 Software Dependency: Modelkit/Params Framework . . . 19
7.2 Additional Concept: Location . . . 19
7.3 Additional Concept: Network Link . . . 20
7.4 Interface Overview . . . 20

8 Example Problem **22**
8.1 Text Input File 22
8.2 Excel User Interface 25

9 When Things Don't Work: Checklist for MS Excel UI **27**

1 Introduction

The purpose of this User's Guide is to give a working introduction to the command-line version of the resilience calculation tool (*ERIN*[1]) and a user interface for the tool written in Microsoft Excel.

The purpose of the tool itself is to simulate the energy flows through a district energy system composed of an interacting network of components. The main contributions of this tool that we maintain are unique in aggregate are as follows:

- the tool accounts for both reliability (failure and repair) as well as resilience to various scenarios (design basis threats)
- while also accounting for topology and interaction between an open-ended number of energy networks
- while providing key energy usage, resilience, and reliability metrics for the modeler / planner

The resilience calculation tool is available as open-source software written in C++.

Several command-line programs are included with the *ERIN* distribution including 3 key executables along with a library written in the *C++* programming language. Documentation of the library itself is beyond the scope of this document. However, the 3 executables will be given attention in this User's Guide as they are of particular interest to modelers.

The minimal user interface written in Microsoft Excel uses the command-line simulation tool behind the scenes as well as a *Modelkit/Params*[2] template to make it easier to use. We will cover usage of the Microsoft Excel interface in addition to the command-line programs.

2 Simulation Overview

In this section, we describe the simulation process to assess the resilience of a district system network to various scenarios including Design Basis Threats.

District Energy Systems play a major role in enabling resilient communities. However, resilience is contextual. That is, one must specify what one is resilient to. This is specified in the tool using various "scenarios" which represent normal operation and various Design Basis Threats. Design Basis Threats are low-probability, high-impact events such as hurricanes, flooding, earthquakes, terrorist attacks, tornadoes, ice storms, viral pandemics, etc. Taking into account relevant Design Basis Threats is necessary for enabling resilient public communities.

The tool operates over networks that supply energy to both individual buildings and districts. These networks are comprised of components (loads, generation, distribution/routing, storage, and transmission assets) and connections. These connections form the topology of the network

[1] ERIN stands for Energy Resilience of Interacting Networks
[2] *Modelkit/Params* is a separate open-source project available from Big Ladder Software.

– what is connected to what. Multiple flows of energy can be modeled: notably, both thermal (heating/cooling) and electrical flows and their interactions.

This network of components is subject to various scenarios which represent one or more ideal cases (i.e., "blue-sky") as well as Design Basis Threats (also known as "black-sky" events). Each scenario has a probability of occurrence and zero or more "damage intensities" associated with it such as wind speed, vibration, water inundation level, etc. Fragility curves are used to relate the scenario's damage intensities with the percentage chance that a given component will fail to work under the duress of the scenario.

Additionally, reliability statistics can be associated with components to model their routine failure and repair times and to take reliability into account in conjunction with various threats. Note, however, that routine reliability statistics are most likely not applicable to an extreme event such as those represented from a design basis threat. Fragility curves are more appropriate for that kind of assessment.

By looking at the performance of the network while taking into account the possibility of failure due to both typical reliability and failure due to threats, resilience metrics such as maximum downtime, energy availability, and load-not-served can be calculated. This can, in turn, help planners to see whether a proposed system or change to an existing system will meet their threat-based resilience goals.

The workflow for using the tool is as follows:

- using a piece of paper, sketch out the network of locations and components and how they are connected
- using either the Excel user interface or a text editor, build an input file that describes:
  - the network of components
    - component physical characteristics
    - component failure modes
    - component fragility
    - how components are connected to each other
  - the scenarios to evaluate
    - the duration of the scenario
    - the occurrence distribution
    - damage intensities involved
  - load profiles associated with each load for each scenario
- simulate the given network over the given scenarios and examine the results

The simulation is specified using a discrete event simulator. Events include:

- changes in a load
- changes in an uncontrollable source such as PV power generation
- routine failure of a working component under reliability
- routine repair of a failed component
- events due to physical limitations of devices (e.g., depleting the energy in a battery or diesel fuel tank)
- the initiation or ending of a scenario
- application of fragility curves at a scenario start

For every event that occurs, the simulation resolves and negotiates the conservation of energy throughout the network. This results in resolving the flows through all connections in the network after each event. Loads in particular are tracked to identify energy not served, time that a load's request is not fully supplied, and also to calculate the energy availability (energy served x 100% / energy requested). These statistics are calculated *by load* and *by scenario*.

3 Concept Overview

This section gives a quick overview of the key concepts used in the tool. Understanding the concepts will help when authoring an input file as well as in interpreting the output results.

3.1 Flows

A flow is any movement of a type of energy. Examples include "electricity", "heated water for district heating", and "chilled water for cooling". The flows specified are open-ended and not prescribed by the tool. However, to aid new users, the Excel User Interface does limit the available flows to those typically used in an assessment.

By being imaginative, flows that are traditionally not considered as "energy flows" can be modeled as well. For example, a supply of potable water pumped to a building can be modeled by phrasing it in terms of enthalpy times mass flow rate: $h \times \dot{m}$ (making assumptions for line pressures and temperatures). This allows the contribution of a pump (changing the pressure and thus the flow work across the pump) to easily be taken into account.

A flow has a direction associated with it. A flow can be zero (i.e., nothing is flowing) but cannot be negative. Negative flows would imply a change in direction which would greatly increase the complexity of the simulation tool. As such, we do not allow negative flows. However, it is possible to simulate bi-directional flow by connecting components from both directions (more on this later).

3.2 Components and Ports

A component is meant to represent any of a myriad of equipment used in a district energy system. A component has zero or more inflow ports and zero or more outflow ports. These ports take in zero or more flows, route and/or transform them, and output zero or more flows. A component must have at least one port: inflow or outflow.

The fidelity of modeling is that of a 1-line diagram and accounts for energy flows only. A component need only be taken into account if:

- it's function will significantly affect network flows
- and it's failure is statistically significant in the face of either reliability or fragility to a threat

For example, a relatively efficient stretch of pipe in a district heating system can be ignored from an energy standpoint if it's losses are insignificant compared to other equipment. However, if that stretch of pipe is deemed to have a statistically significant possibility of failure during a threat event such as an earthquake, it should be modeled. In this instance, a pass-through component (see below) with a fragility curve (see below) may be a good choice.

3.3 Component Types

Because we model components at a high-level of abstraction, a few component types are all that's needed to model many real-world components. In this section, we discuss the available component types and their characteristics.

3.3.1 Component Type: Load

A load is essentially an exit point out of the network for "useful work". A load typically represents an end use such as a building or cluster of building's electricity consumption or heating load consumption.

A load specifies its load versus time with a load profile which is specified per scenario.

3.3.2 Component Type: Source

A source is an entry point into the simulation for providing energy flow into the network. A source typically represents useful energy into the system such as electrical energy from the utility, natural gas into the district, or diesel fuel transported to a holding tank.

3.3.3 Component Type: Uncontrolled Source

Normally, a source responds to a request up to its available max output power (which defaults to being unlimited). In contrast, an uncontrolled source cannot be commanded to a given outflow because the source is uncontrollable. Typical examples of uncontrolled sources are electricity generated from a photovoltaic array, heat generated from concentrating solar troughs, or electricity from a wind farm. Another typical uncontrolled source is heat to be removed from a building as a "cooling load".

An uncontrolled source specifies its supply values versus time with a supply profile which is specified per scenario. Note: functionally, a supply profile and load profile are the same thing.

3.3.4 Component Type: Converter

A converter represents any component that takes in one kind of flow and converts it to another type of flow, usually with some loss. Converters have an efficiency associated with them. The current version of the tool only supports a constant-efficiency. Typical examples of converter components are boilers, electric generators (e.g., fired by diesel fuel or natural gas), transformers, and line-losses.

The loss flow from one component can be chained into another converter component to simulate various loss-heat recovery mechanisms and equipment such as combined heat and power (CHP) equipment.

3.3.5 Component Type: Storage

A storage component represents the ability to store flow. The storage unit has both a charge (inflow) port and a discharge (outflow) port. The storage component cannot accept more flow than it has capacity to store. Similarly, a storage component cannot discharge more flow than it has stored. Typical examples of a storage component include battery systems, pumped hydro, diesel fuel storage tanks, coal piles, and thermal energy storage tanks.

The current version of the storage tank does not have an efficiency or leakage component associated with it. However, charge/discharge efficiency can be approximated with converter components and leakage via a small draw load.

3.3.6 Component Type: Pass-Through

A pass-through component is a component that physically exists on the system but that only passes flow through itself without disruption. As such, it is does not change the energy flow of the network. Therefore, the main use for a pass-through component is in providing equipment to associate failure modes and fragility curves (discussed below) with. Since failure of the component results in a loss of a flow, it may be important to take into account. Typical examples of pass-through components are above-ground and below-ground power lines, natural-gas pipe runs, district heating pipe runs, etc.

3.3.7 Component Type: Muxer

A "muxer" or multiplexer component represents various components for splitting and joining flows. Typical examples include manifolds, routers, electrical bus bars, and the like.

Muxers can have multiple inflow ports and multiple outflow ports. Muxer's contain dispatch strategies to choose how requests are routed. There are two dispatch strategies available in the current tool:

- in-order dispatch: all flow is requested to be satisfied from the first inflow port first. If that flow is insufficient, the second inflow port is requested for the remainder until all inflow ports are exhausted. For cases where outflow request is not met, the first outflow port is satisfied first. If flow remains, that flow is routed to the next port until it is satisfied or the flow is spent, and so on to the next port, etc.
- distribute dispatch: all flow is distributed between all ports. In this strategy, requests are distributed evenly between inflow ports. When flow is insufficient to meet all outflow request, available flow is distributed evenly to outflow ports.

These strategies are not sophisticated enough to cover advanced energy saving strategies. However, they should be sufficient to mimic basic dispatch strategies for assessing load supply.

3.3.8 Component Type: Mover

Note: the mover component is currently only available from the command-line interface. It has not yet been made available for the Excel User Interface.

A mover component is a component that moves energy from its inflow port to its outflow port with the assistance of a support flow. Movers can be used to represent chillers and heat pumps (which move heat) as well as pumps and fans (which move fluids).

3.4 Networks and Connections

Component connections via ports form a network. Networks describe the interaction of various flows.

A connection describes:

- a source component and its outflow port
- a sink (i.e., receiving) component and its inflow port
- and the type of flow being delivered

3.5 Scenarios

Scenarios represent both typical usage (i.e., blue sky events) and design basis threat events (class 4 hurricanes, earthquakes, land-slides, etc.).

A scenario has:

- a duration (how long the scenario will last)
- an occurrence distribution which is a cumulative distribution function that expresses the likelihood of occurrence
- a maximum number of times the scenario can occur during the entire simulation (either *unlimited* or some finite number)
- and various damage intensities associated with the scenario

The damage intensities associated with a scenario are open-ended but are meant to represent numerical quantities that correspond with a fragility curve. Some examples of damage intensities that could be associated with a scenario are "wind speed", "inundation depth", "vibration", etc. Scenarios with no damage intensities are completely fine – these would represent "blue-sky" scenarios (typical operation).

3.6 Reliability: Failure Modes and Statistical Distributions

Reliability is handled strictly as a statistical matter using failure modes. A failure mode is an associate between a failure cumulative distribution function and the corresponding repair cumulative distribution function. Multiple failure modes can be specified for a single component. For example, a diesel back-up generator may have one failure mode associated with its starter battery and another to represent more serious issues with the generator itself.

Every failure mode in the simulation is turned into an "availability schedule". That is, for each failure mode, the dual cumulative distribution functions are alternatively sampled from time 0 to the end of the overall simulation time to derive a schedule of "available" and "failed". When a scenario where reliability is calculated is scheduled to occur, the relevant portion of the availability schedules for components with failure modes are used to "schedule" the component as available and failed to simulate routine reliability events during that scenario's simulation.

3.7 Resilience: Intensities (Damage Metrics) and Fragility Curves

Resilience reflects how components react to the intense stresses of a design basis threat event. Each scenario can specify an intensity or damage metric. Any component having a fragility curve that responds to one or more of the scenario intensities is evaluated for failure due to the scenario's intensity.

For example, above-ground power lines may have a fragility to wind speed. If a scenario specifies a wind speed of 150 mph, the above-ground power line component will use its fragility curve to look up its chance of failure. For fragility, a component is evaluated for failure at scenario start and either passes (staying up during the scenario) or fails (going down for the entire scenario).

4 Input File Format

The simulation engine is a command-line program. Even when it is accessed via the Excel User Interface, a text-based input file is written to describe the network of components and scenarios to simulate.

The input file format is written using the TOML[3] input file language. TOML is a plain-text input file format.

The file consists of the following sections that describe the various concepts described above:

- `simulation_info`: general simulation information
- `loads`: load profiles (includes supply profiles for uncontrolled sources)
- `components`: all components in the network are described here
- `fragility`: all fragility curves are described here
- `cdf`: cumulative distribution functions
- `failure_mode`: failure modes are described here
- `networks`: networks are described here
- `scenarios`: scenarios are described here

Valid entries for each of the sections are described in Table 1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, 16.

The types given are one of:

- str: a string of characters in "quotes"
- bool: true or false
- real: a real number (0.0, 1.5, 2e7, etc.)
- real>0: a real number greater than 0.0. $0.0 <$ real>0
- int: an integer
- int>0: an integer > 0
- $[X]$: an array of the given type, X
- $[[X]]$: an array of arrays of X
- time: time unit. One of {"years", "days", "hours", "minutes", "seconds"}
- cap: capacity unit. One of {"kJ", "kWh"}
- disp: dispatch strategy. One of {"distribute", "in_order"}.
- frac: real fraction. $0.0 \leq$ frac ≤ 1.0
- frac>0: real fraction greater than 0.0. $0.0 <$ frac ≤ 1.0
- rate: the rate unit. Currently, only "kW" is accepted.

[3]TOML is described in detail here: https://toml.io/en/

- $X \to Y$: designates a map data structure (a.k.a., dictionary, hash table, table, etc.). Associates Y with X.

In the TOML input file, all constructs except `simulation_info` have an id. The id is used when one construct references another.

This looks as follows:

```
[loads.load_id_1]
...
[loads.load_id_2]
...
[components.comp_id_1]
...
[components.comp_id_2]
...
[fragility.fragility_id_1]
...
[cdf.cdf_id_1]
...
[failure_mode.fm_id_1]
...
[networks.nw_id_1]
...
[scenarios.scen_id_1]
...
```

An id must follow the rules of TOML "bare keys"[4] with the exception that dashes (`-`) are not allowed and the key must start with an ASCII letter:

> Bare keys may only contain ASCII letters, ASCII digits, underscores, … .

Table 1: `simulation_info` specification

key	type	required?	notes
`time_unit`	time	no	The time unit. Default "years"
`fixed_random`	frac	no	Sets the random roll to a fixed value
`fixed_random_series`	[real]	no	Sets random numbers to the given series
`random_seed`	real	no	Sets the random number generator's seed
`max_time`	int	no	Maximum simulation time. Default: 1000

Note: Table 1 specifies various random values. At most, one of these values can be specified.

Table 2: `loads` specification

key	type	required?	notes
`csv_file`	str	no	path to CSV file with profile
`time_rate_pairs`	[[real]]	no	array of (time, rate) pairs
`time_unit`	time	no	time unit for `time_rate_pairs`
`rate_unit`	rate	no	rate unit for `time_rate_pairs`

[4] see https://toml.io/en/v1.0.0-rc.1#keys

For Table 2, one must specify either a `csv_file` *or* `time_rate_pairs`, `time_unit`, and `rate_unit`. Unfortunately, only "kW" is available for `rate_unit` at the moment although `time_unit` accepts "years", "seconds", or "hours". Practically speaking, you will almost always use a `csv_file` unless you just want to test a simple load.

For the `csv_file`, the header must be "hours,kW" with data filled into the rows below. The "hours" column is the elapsed time in hours. The "kW" column is the flow in kW. The first column header can be set to values beside "hours"; any time unit is valid. However, the rate unit is currently locked in as "kW".

Table 3: `components`: common attributes

key	type	required?	notes
`failure_modes`	[str]	no	failure mode ids for component
`fragilities`	[str]	no	fragility curve ids for component

Table 3 lists the attributes common to all components. These relate to reliability and resilience: failure modes and fragility curves.

Table 4: `components`: Source Component

key	type	required?	notes
`type`	str	yes	must be "source"
`outflow`	str	yes	type of outflow
`max_outflow`	real	no	maximum allowable outflow

Table 5: `components`: Load Component

key	type	required?	notes
`type`	str	yes	must be "load"
`inflow`	str	yes	type of outflow
`loads_by_scenario`	str $\rightarrow$ str	yes	map of scenario id to load id

In Table 5, the `loads_by_scenario` structure is specified as follows:

```
loads_by_scenario.scenario_id_1 = "load_id_1"
loads_by_scenario.scenario_id_2 = "load_id_2"
```

Table 6: `components`: Converter Component

key	type	required?	notes
`type`	str	yes	must be "converter"
`inflow`	str	yes	type of inflow
`outflow`	str	yes	type of outflow
`lossflow`	str	no	type of lossflow. Default: inflow
`constant_efficiency`	frac>0	yes	constant efficiency

Table 7: `components`: Storage Component

key	type	required?	notes
`type`	str	yes	must be "store"
`flow`	str	yes	type of flow (inflow, outflow, stored)
`capacity_unit`	cap	no	capacity unit. Default: "kJ"
`capacity`	real	yes	capacity of the store
`max_inflow`	real	yes	maximum inflow (charge rate)

During simulation, the `max_inflow` sets the requested charging rate for a storage unit (see Table 7). By default, a storage unit will always request to charge itself to its maximum capacity. However, it will always honor its discharge request above its charge request. That is, if discharge is requested, it will discharge rather than charge. If charging and discharging at the same time, charge flow will "short circuit" to meet the discharge request first. Any flow left over will charge the store.

Table 8: `components`: Muxer Component

key	type	required?	notes
`type`	str	yes	must be "muxer"
`flow`	str	yes	type of flow (inflow, outflow)
`num_inflows`	int	yes	the number of inflow ports
`num_outflows`	int	yes	the number of outflow ports
`dispatch_strategy`	disp	no	dispatch strategy. Default: "`in_order`"

In Table 8, the `dispatch_strategy` refers to the strategy at the outflow of the muxer. The inflow strategy is always "`in_order`". That is, the first connected port gets the full request. If that inflow port can't meet the full flow, we request the remaining flow from the second inflow port, etc. The outflow strategy is set in the model input file using the `dispatch_strategy` key as shown in Table 8.

The `dispatch_strategy` for a muxer only manifests when there is a flow deficiency. That is, normally, all requests at each outflow port are achieved. However, when there is not enough flow, "`in_order`" dispatch feeds the first outflow port first and then turns its attention to the second and so on until flow runs out. For a "`distribute`" `dispatch_strategy`, when flow is lacking, the available flow is distributed evenly.

Let's consider an example. A muxer with 4 outflow ports gets the following request: [50, 50, 50, 50] (= 200 kW). However, only 100 kW is available to supply these outflow requests. An "`in_order`" dispatch will provide [50, 50, 0, 0] (= 100 kW) to its four outflow ports. In contrast, a "`distribute`" `dispatch_strategy` will provide [25, 25, 25, 25] (= 100 kW) to each outflow port. Consider a non-uniform request of say [50, 10, 90, 50] (= 200 kW) on the same mux; again, however, only 100 kW is available. An "`in_order`" dispatch would provide [50, 10, 40, 0] (= 100 kW). In contrast, a "`distribute`" dispatch strategy would provide [30, 10, 30, 30] (= 100 kW) to each outflow port.

Table 9: `components`: Pass-Through Component

key	type	required?	notes
`type`	str	yes	must be "pass_through"

key	type	required?	notes
`flow`	str	yes	type of flow (inflow, outflow)
`max_outflow`	real>0	no	defaults to infinite flow

Table 10: `components`: Uncontrolled Source Component

key	type	required?	notes
`type`	str	yes	must be "uncontrolled_source"
`outflow`	str	yes	type of outflow
`supply_by_scenario`	str → str	yes	scenario id to load profile id

Similar to the load component, the uncontrolled source's `supply_by_scenario` specifies supply profiles by scenario. These look like the following:

```
supply_by_scenario.scenario_id_1 = "load_id_1"
supply_by_scenario.scenario_id_2 = "load_id_2"
```

Note that the uncontrolled source supply profiles are also drawn from the same section of the input file specified as `loads`.

Table 11: `components`: Mover Component

key	type	required?	notes
`type`	str	yes	must be "mover"
`inflow0`	str	yes	the inflow being "moved"
`inflow1`	str	yes	the "support" inflow that enables "moving" to occur
`outflow`	str	yes	the outflow
`cop`	real>0	yes	the coefficient of performance

In Table 11, the `cop` field ties together the three flows `inflow0`, `inflow1`, and `outflow` using the following relations:

$$cop = \frac{inflow_0}{inflow_1}$$

$$inflow_0 = cop \times inflow_1$$

$$inflow_1 = inflow_0 \times \frac{1}{cop}$$

$$outflow = (1 + cop) \times inflow_1 = (1 + \frac{1}{cop}) \times inflow_0 = inflow_0 + inflow_1$$

Table 12: `fragility` specification

key	type	required?	notes
`vulnerable_to`	str	yes	the scenario intensity (i.e., damage metric) vulnerable to
`type`	str	yes	must be "linear"
`lower_bound`	real	yes	the value below which we are impervious to damage
`upper_bound`	real	yes	the value above which we face certain destruction

Fragility curves are specified using the attributes listed in Table 12. Figure 1 shows a graphical representation of the data specification.

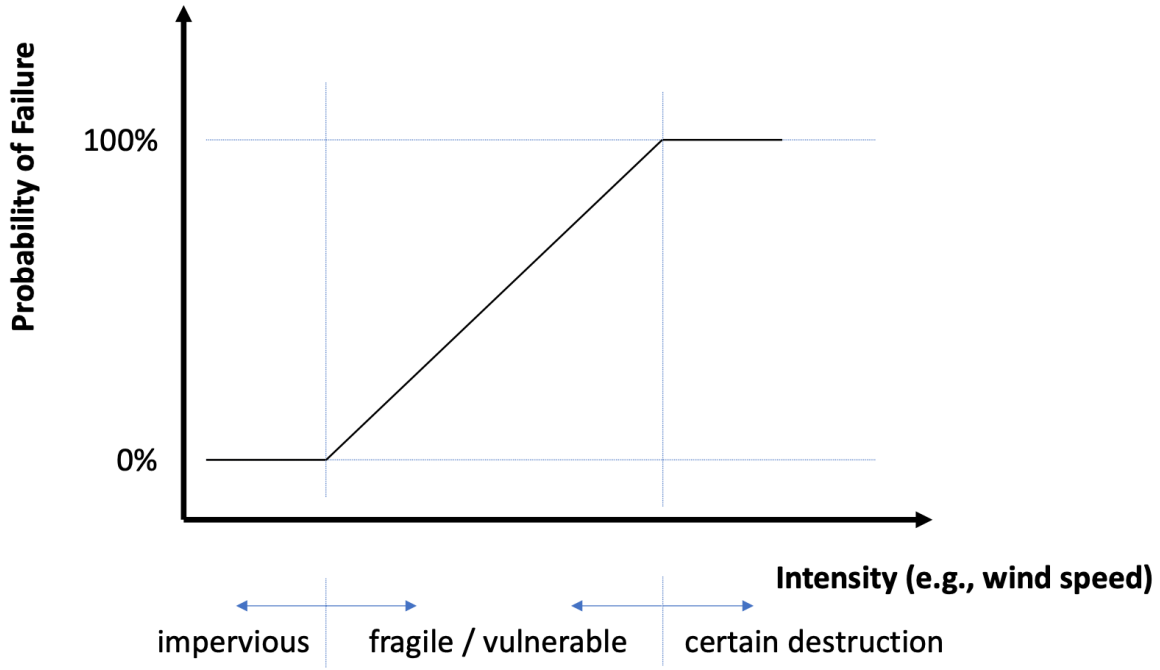

Figure 1: Fragility Curve

A fragility curve maps a scenario's intensity (i.e., damage metric) to a probability of failure. We must specify which damage metric is of interest and also the curve relationship. Currently, the only available fragility curve type is linear. For the linear curve, we specify the `lower_bound`, the bound below which we are impervious to destruction. We also specify the `upper_bound`, the bound above which we face certain destruction.

Table 13: `cdf` specification

key	type	required?	notes
`type`	str	yes	must be "linear"
`value`	real	yes	the value of the fixed CDF
`time_unit`	time	yes	the time unit used to specify the fixed value

Table 13 specifies a cumulative distribution function. At this time, the only distribution type available is "fixed". A fixed distribution is a degenerate distribution that always samples a single point – the `value`.

Table 14: `failure_mode` specification

key	type	required?	notes
`failure_cdf`	str	yes	the failure CDF id
`repair_cdf`	str	yes	the repair CDF id

Table 15: `networks` specification

key	type	required?	notes
`connections`	[[str]]	yes	the connections

The `networks` data definition involves a "mini-language" to specify connections. The language is as follows:

```
connections = [
  [ "src_comp_id:OUT(outflow_port)", "sink_comp_id:IN(inflow_port)", "flow"],
  ...
  ]
```

The `connections` key is an array of 3-tuples. The first element of the 3-tuple is the source component id separated by a ":" and then the word "OUT(.)". You will type the outflow port in place of the ".". Note that numbering starts from 0.

The second element of the 3-tuple is the sink component id. That is, the component that *receives* the flow. The sink component id is written, then a ":", and finally the word "IN(.)". You will type the inflow port id in place of the ".". Numbering of inflow ports starts from 0.

The final element of the 3-tuple is the flow id. You are requested to write the flow id as a check that ports are not being wired incorrectly.

Table 16: `scenarios` specification

key	type	required?	notes
`time_unit`	time	no	time units for scenario. Default: "hours"
`occurrence_distribution`	table	yes	see notes in text
`duration`	int>0	yes	the duration of the scenario
`max_occurrences`	int	yes	the maximum number of occurrences. -1 means unlimited
`calculate_reliability`	bool	no	whether to calculate reliability. Default: false
`network`	str	yes	the id of the network to use
`intensity`	str → real	no	specify intensity (damage metric) values

In Table 16, the `occurrence_distribution` is currently implemented as a literal table:

```
occurrence_distribution = { type = "linear", value = 8, time_unit = "hours" }
```

The possible values for the `occurrence_distribution` table are given in Table 13.

5 Output Metrics

The metrics used to assess resilience are given an overview in this section. Figure 2 depicts the metrics graphically. It is important to note that metrics are calculated *by load* and *per scenario*.

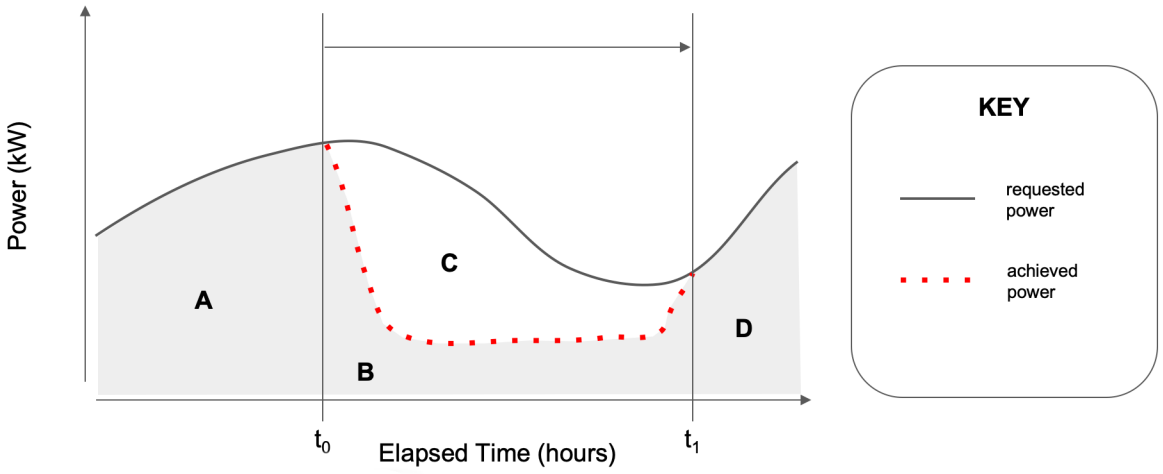

Figure 2: Resilience and Energy Metrics

As seen in Figure 2, there are three basic calculations:

- energy availability
- load not served
- and max downtime

Figure 2 shows four areas of flow integration over time: A, B, C, and D. The sum of A, B, and D is the energy delivered to this load for this scenario. C represents the load not served. The ratio of $\frac{(A+B+D)\times 100\%}{A+B+C+D}$ is the energy availability. The duration of load interruption (from t_0 to t_1) is the max downtime.

The energy availability is calculated as follows:

$$EA = \frac{E_{achieved} \times 100\%}{E_{requested}}$$

In the above equation, the energy, E, is the integral of the flow, f, over time:

$$E = \int_{t=0}^{t_{end}} f \cdot dt$$

Max downtime is the duration of load interruption:

$$T_{\text{down}} = \int dt \text{ where } f_{\text{achived}} < f_{\text{requested}}$$

Load not served is then:

$$E_{\text{not served}} = \int_{t=0}^{t_{end}} (f_{\text{requested}} - f_{\text{achieved}}) \cdot dt$$

6 Command-Line Tool

Three command-line programs are available for simulation and assistance. They will be given an overview here.

6.1 `erin`

Simulates a single scenario and generates results.

USAGE:

`erin <input_file_path> <output_file_path> <stats_file_path> <scenario_id>`

- `input_file_path`: path to TOML input file
- `output_file_path`: path to CSV output file for time-series data
- `stats_file_path`: path to CSV output file for statistics
- `scenario_id`: the id of the scenario to run

The output from the call to `erin` will be written into two files: an output file and a statistics file.

The output file has the column headers shown in Table 17.

Table 17: `erin` Output

Column	Description
time (hours)	the elapsed time since scenario start in hours
*:achieved (kW)	the achieved flow at the event time for each component/port recorded
*:requested (kW)	the requested flow at the event time for each component/port recorded

The statistics file has the column headers shown in Table 18.

Table 18: `erin` Statistics

Column	Description
component id	the id of the component
type	the type of the component (e.g., load, source, etc.)
stream	the stream flowing through the given component / port
energy availability	the energy availability for the given component
max downtime (hours)	the maximum number of contiguous hours when load not fully met

Column	Description
load not served (kJ)	the load not served in kJ
X energy used (kJ)	for each flow, report out the energy used in kJ
TOTAL (X)	the total energy used by flow by component type
ENERGY BALANCE	a sum of the energy balance. Should be 0

6.2 `erin_multi`

Simulates all scenarios in the input file over the simulation time and generates results.

USAGE:

```
erin_multi <input_file_path> <output_file_path> <stats_file_path>
```

- `input_file_path`: path to TOML input file
- `output_file_path`: path to CSV output file for time-series data
- `stats_file_path`: path to CSV output file for statistics

The output files from `erin_multi` are very similar to those shown in Table 17, 18. The main difference is that `erin_multi` aggregates across multiple scenario instances and multiple scenario types. The column headers used in the event output file for `erin_multi` are shown in Table 19.

Table 19: `erin_multi` Output

Column	Description
scenario id	the id of the scenario simulated
sceanrio start time	start time of scenario in ISO 8601 duration format
elapsed (hours)	the elapsed time since scenario start in hours
*:achieved (kW)	the achieved flow at the event time for each component/port recorded
*:requested (kW)	the requested flow at the event time for each component/port recorded

The statistics file for `erin_multi` has the column headers as shown in Table 20.

Table 20: `erin_multi` Statistics

Column	Description
scenario id	scenario id for the scenario reported out
number of occurrences	number of times the scenario occurred during simulation
total time in scenario (hours)	total time spent in the scenario during simulation
component id	the id of the component
type	the type of the component (e.g., load, source, etc.)
stream	the stream flowing through the given component / port
energy availability	the energy availability for the given component

Column	Description
max downtime (hours)	the maximum number of contiguous hours when load not fully met
load not served (kJ)	the load not served in kJ
X energy used (kJ)	for each flow, report out the energy used in kJ
TOTAL (X)	the total energy used by flow by component type
ENERGY BALANCE	a sum of the energy balance. Should be 0

6.3 `erin_graph`

Generates an input file for use with Graphviz. Graphviz is an external dependency. You do not need Graphviz to generate the Graphviz input file. However, you *do* need Graphviz to process that input file into a `.png` or `.pdf` file.

USAGE:

```
erin_graph <input_file_path> <dot_file_path> <network_id>
```

- `input_file_path`: path to TOML input file
- `dot_file_path`: path to Graphviz DOT file to write
- `network_id`: id for the network to plot from input_file_path

Upon successful execution, you can render your Graphviz dot file into a PNG (image file) as follows:

- `dot -Tpng input.gv -o output.png` The above generates a png (`-Tpng`) from the `input.gv` and saves to `output.png` (`-o`)

Similarly, you can render your Graphviz dot file into a PDF as follows:

- `dot -Tpdf input.gv -o output.pdf` The above generates a png (`-Tpdf`) from the `input.gv` and saves to `output.pdf` (`-o`)

`erin_graph` is capable of creating sophisticated topological graphs such as the one in Figure 3.

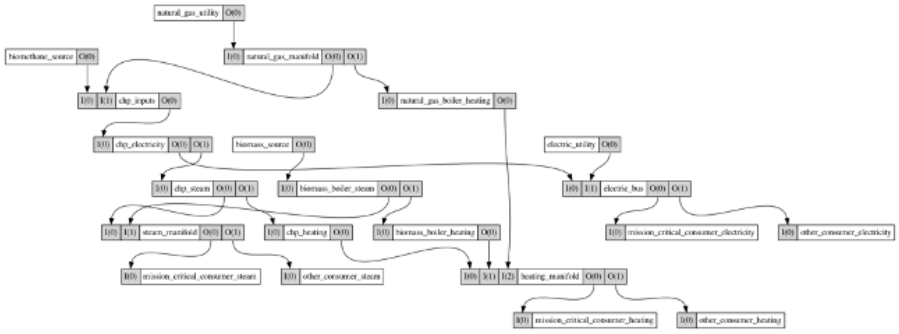

Figure 3: Topology Example of a Network Rendered with Graphviz and `erin_graph`

7 Microsoft Excel User Interface

A simple interface using Microsoft Excel has been created to ease the creation of an input data file for `erin`. This interface runs the simulation on behalf of the user and also pulls the input Due to limitations in Excel's Visual Basic, the current version of the Microsoft Excel user interface is limited to the Windows Operating System.

7.1 Software Dependency: Modelkit/Params Framework

Modelkit/Params is required to allow the Microsoft Excel user interface to render an input file template for `erin_multi`. *Modelkit/Params* is a third party dependency available as open-source software from Big Ladder Software:

https://bigladdersoftware.com/projects/modelkit/

Please install *Modelkit Catalyst*. Version 0.5 or later is required.

Further detail about *Modelkit* can be found at the above link.

7.2 Additional Concept: Location

To make it easier for modelers to specify a network of components, the Excel user interface introduces an additional concept called a "location". A location is open-ended although the names and ids must follow the same rules for ids as in the input file (see Section 4).

At a given location, any number of components can be specified. The Excel User Interface uses Big Ladder's *Modelkit/Params* to render an input file from a template. That template assumes a given topology for each location as shown in Figure 4.

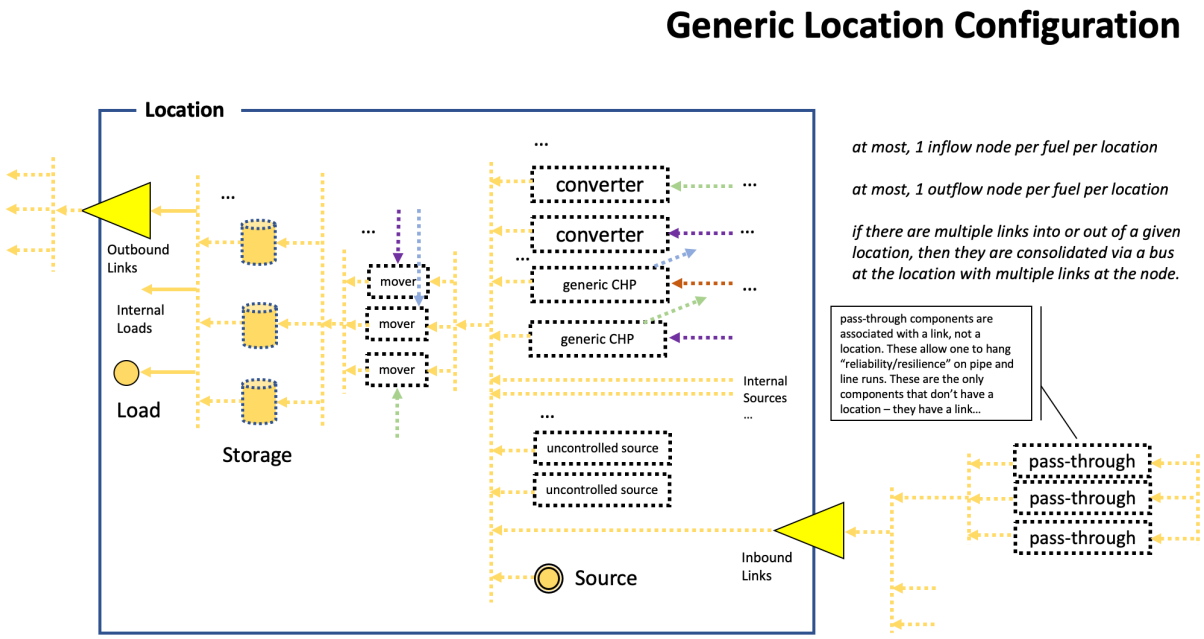

Figure 4: Topology at a Location

As depicted in Figure 4, for any given flow type, the following sets of components are in series:

- all loads: load components, internal loads, and outbound links
- all storage for the given flow type (multiples are in parallel)
- all mover components: multiples are in parallel
- all converters and sources: converters (including CHP which is modeled as chained converters), internal sources, uncontrolled sources, inbound links, and normal source components

7.3 Additional Concept: Network Link

The location topology template shown in Figure 4 alludes to inbound and outbound links. The links themselves are called "network links". They are similar to normal connections except that they connect locations.

We believe the template in Figure 4 to be typical of how components at a location are typically connected, topologically speaking. However, if further variation is needed, it can often be achieved by creating multiple locations and linking them together. For example, if one wanted to model two storage units in *series* (vs *parallel*), they need only create a storage in location A and another in location B and denote that location B has a network link from B to A.

7.4 Interface Overview

The Excel user interface to `erin_multi` is laid out logically to help new users specify a component network to simulate.

The major screens are:

- Instructions (see Figure 5)
- Settings (see Figure 6)
- Components (see Figure 7)
- Network (see Figure 8)
- Scenarios (see Figure 9)

The "Instructions" sheet gives light instructions on how to use the workbook. The "Settings" tab is where the path to `erin_multi.exe` is set. A modeler can also add additional statistical distributions, failure modes, and fragility curves here. The "Components" tab is where a modeler can add different types of components to a location. The "Network" tab is where network links between locations can be specified. The "Scenarios" tab is where different Scenarios can be added and configured.

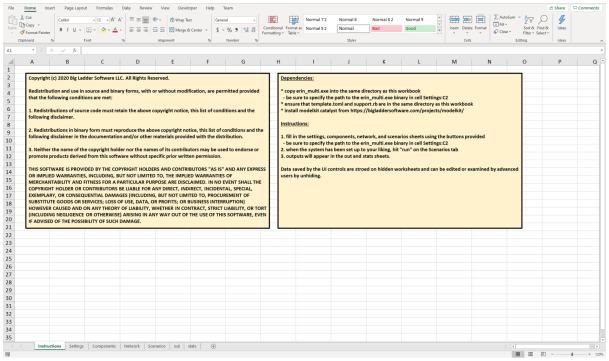

Figure 5: Excel Interface: Instructions Sheet

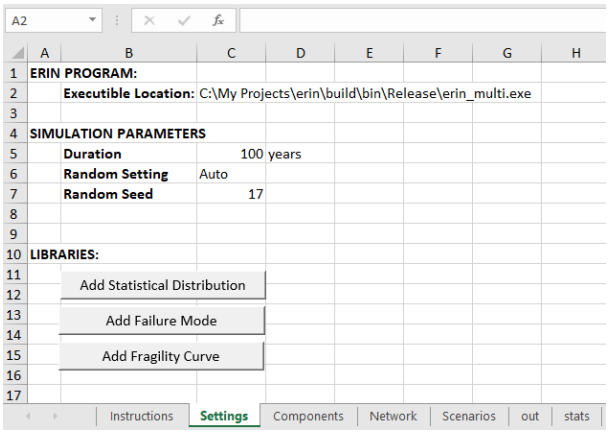

Figure 6: Excel Interface: Settings Sheet

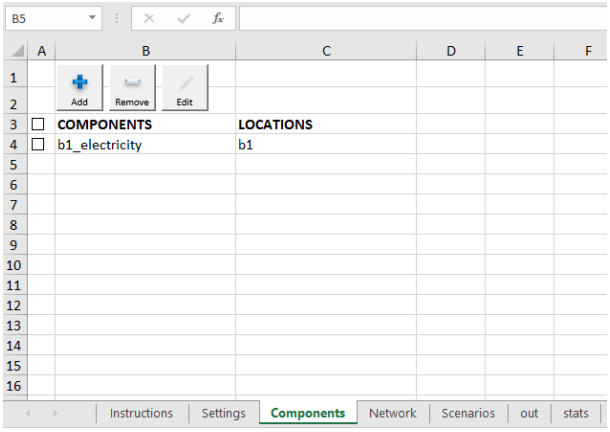

Figure 7: Excel Interface: Components Sheet

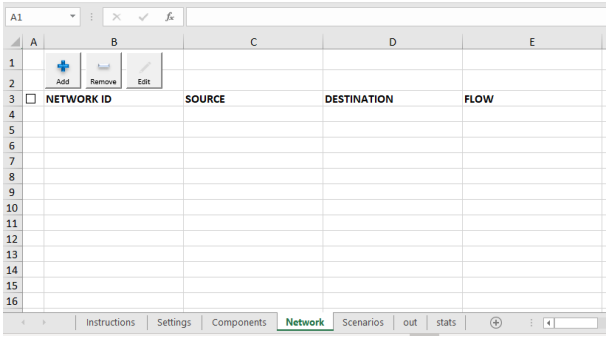

Figure 8: Excel Interface: Network Sheet

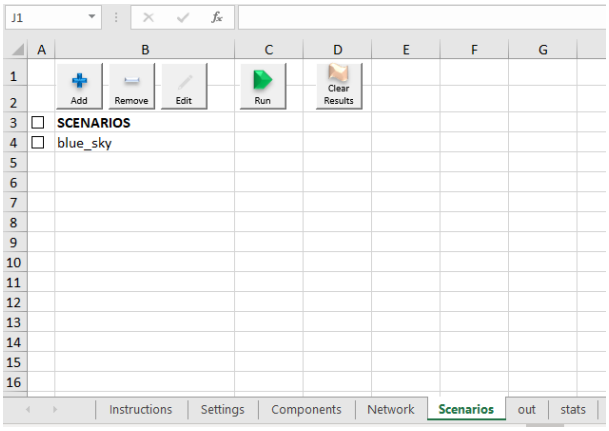

Figure 9: Excel Interface: Scenario Sheet

8 Example Problem

In this final section, we will specify a simple problem using both the input file and the Excel User Interface. The problem will involve a single building with an electrical load, an electric generator on-site, and a utility supply of natural gas. We will simulate two scenarios: a blue-sky scenario and a class 4 hurricane scenario.

An iconic sketch of the network we will build appears in Figure 10.

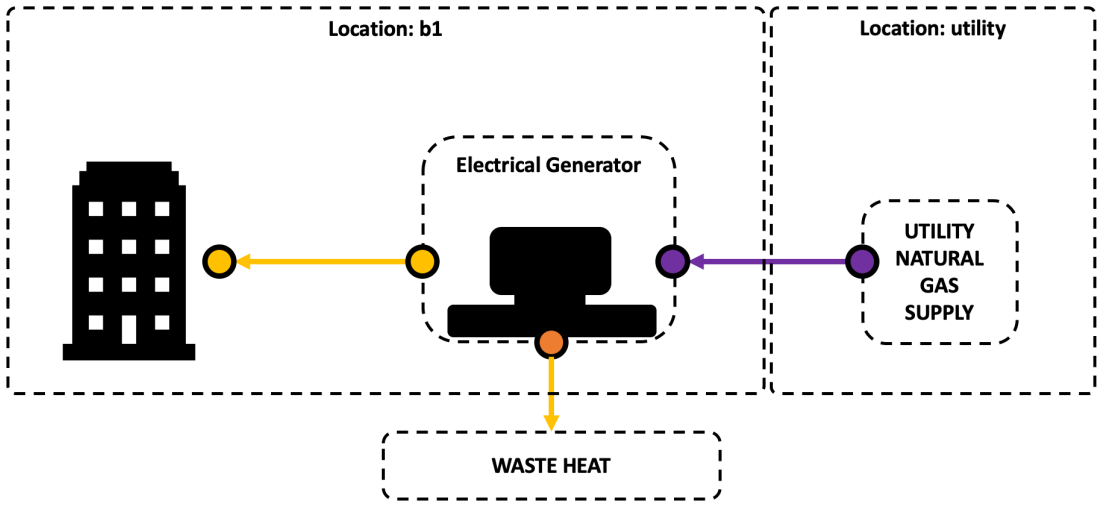

Figure 10: Example Network

The steps to create this network and simulate it are as follows:

8.1 Text Input File

1. Open a new file `input.toml` for editing using your favorite text editor. Add the following simulation information:

```toml
[simulation_info]
rate_unit = "kW"
quantity_unit = "kJ"
time_unit = "years"
max_time = 100
```

2. Create a simple load profile by hand. Open the file `b1-load-profile.csv` in your favorite text editor. Type in the following and save:

```
hours,kW
0,100
8760,0
```

3. Back in `input.toml`, add the load profile information at the end of the file:

```toml
[loads.lp1]
csv_file = "b1-load-profile.csv"
```

4. Next, still within `input.toml`, let's add the components:

```toml
[components.utility_ng_source]
type = "source"
outflow = "natural_gas"

[components.b1_electricity]
type = "load"
inflow = "electricity"
loads_by_scenario.blue_sky = "lp1"
loads_by_scenario.c4_hurricane = "lp1"

[components.b1_electric_generator]
type = "converter"
inflow = "natural_gas"
outflow = "electricity"
lossflow = "waste_heat"
constant_efficiency = 0.42
fragilities = ["flooding", "wind"]
```

In the table above, we have added a natural gas source (`utility_ng_source`), an electrical load at building 1 (`b1_electricity`), and an electrical generator at building 1 (`b1_electric_generator`). The electric generator has an efficiency of 42% and has two fragilities: "`flooding`" and "`wind`". Neither of the fragilities have been specified yet, so we'll tackle them next.

5. Within `input.toml`, specify the fragility curves.

```toml
[fragility.flooding]
vulnerable_to = "inundation_depth_ft"
type = "linear"
lower_bound = 4.0
upper_bound = 8.0

[fragility.wind]
```

```
vulnerable_to = "wind_speed_mph"
type = "linear"
lower_bound = 150.0
upper_bound = 220.0
```

These fragility curves reflect the specific situation of the equipment versus the threat.

6. Specify the network connections.

```
[networks.nw]
connections = [
  ["utility_ng_source:OUT(0)", "b1_electric_generator:IN(0)", "natural_gas"],
  ["b1_electric_generator:OUT(0)", "b1_electricity:IN(0)", "electricity"],
  ]
```

7. Specify the scenarios.

```
[scenarios.blue_sky]
time_unit = "hours"
occurrence_distribution = {type = "linear", value = 0, time_unit="hours"}
duration = 8760
max_occurrences = 1
calculate_reliability = true
network = "nw"
[scenarios.c4_hurricane]
time_unit = "days"
occurrence_distribution = {type = "linear", value = 30, time_unit="years"}
duration = 14
max_occurrences = -1
calculate_reliability = true
network = "nw"
intensity.wind_speed_mph = 155.0
intensity.inundation_depth_ft = 6.0
```

The finished file should look like the following:

```
[simulation_info]
rate_unit = "kW"
quantity_unit = "kJ"
time_unit = "years"
max_time = 100

[loads.lp1]
csv_file = "b1-load-profile.csv"

[components.utility_ng_source]
type = "source"
outflow = "natural_gas"

[components.b1_electricity]
type = "load"
inflow = "electricity"
loads_by_scenario.blue_sky = "lp1"
```

```
loads_by_scenario.c4_hurricane = "lp1"

[components.b1_electric_generator]
type = "converter"
inflow = "natural_gas"
outflow = "electricity"
lossflow = "waste_heat"
constant_efficiency = 0.42
fragilities = ["flooding", "wind"]

[fragility.flooding]
vulnerable_to = "inundation_depth_ft"
type = "linear"
lower_bound = 4.0
upper_bound = 8.0

[fragility.wind]
vulnerable_to = "wind_speed_mph"
type = "linear"
lower_bound = 150.0
upper_bound = 220.0

[networks.nw]
connections = [
  ["utility_ng_source:OUT(0)", "b1_electric_generator:IN(0)", "natural_gas"],
  ["b1_electric_generator:OUT(0)", "b1_electricity:IN(0)", "electricity"],
  ]

[scenarios.blue_sky]
time_unit = "hours"
occurrence_distribution = {type = "fixed", value = 0, time_unit="hours"}
duration = 8760
max_occurrences = 1
network = "nw"

[scenarios.c4_hurricane]
time_unit = "days"
occurrence_distribution = {type = "fixed", value = 30, time_unit="years"}
duration = 14
max_occurrences = -1
network = "nw"
intensity.wind_speed_mph = 155.0
intensity.inundation_depth_ft = 6.0
```

The file can be called as `erin_multi.exe input.toml out.csv stats.csv`. This assumes that `erin_multi.exe` is on your path.

8.2 Excel User Interface

Using the Excel Interface, we will create the same problem specified in Figure 10.

1. Open the workbook and ensure the path to `erin_multi.exe` is set. See Figure 6. Also, ensure you have the following files in one directory as shown in Figure 11:

- `erin_gui.xlsm`
- `erin_multi.exe`
- `support.rb`
- `template.toml`
- `example-load.csv`: not required in general but we'll use it for this example

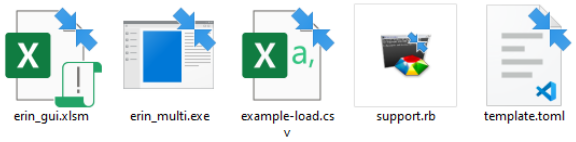

Figure 11: Required files to run the Excel UI

An easy way to get the path to `erin_multi.exe` is to find it in the file system and, while holding the SHIFT key, right click on the file and select "Copy as Path" as shown in Figure 12. The value so copied can be pasted into the cell with the path in the Settings sheet. Be sure to save the workbook once you've set the path.

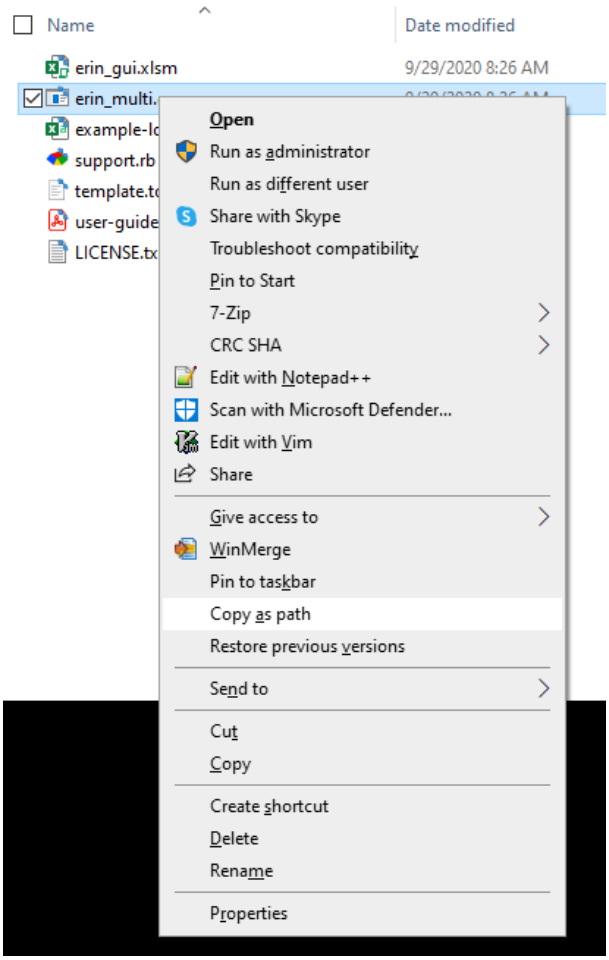

Figure 12: Easy Way to Copy the Path to `erin_multi.exe`

2. We will start by adding two fragility curves. See Figure 13, 14.

We'll call the first fragility curve "wind" and set the "Vulnerable To" field to "`wind_speed_mph`" with a range from 150 to 220 mph. The second fragility curve we'll call "flooding" and set the "Vulnerable To" field to "`inundation_depth_ft`" with bounds of 4.0 to 8.0 feet.

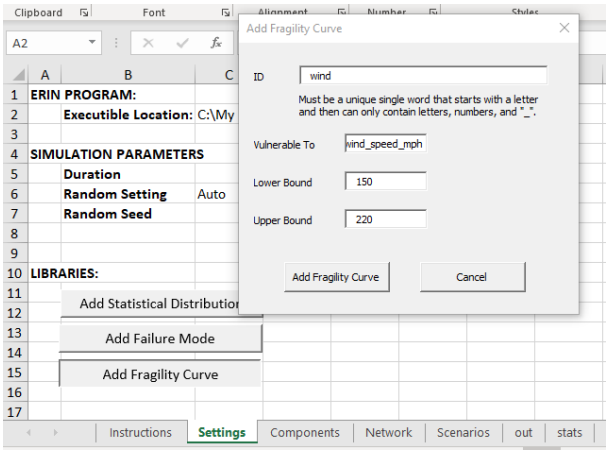

Figure 13: Add Fragility Curve #1

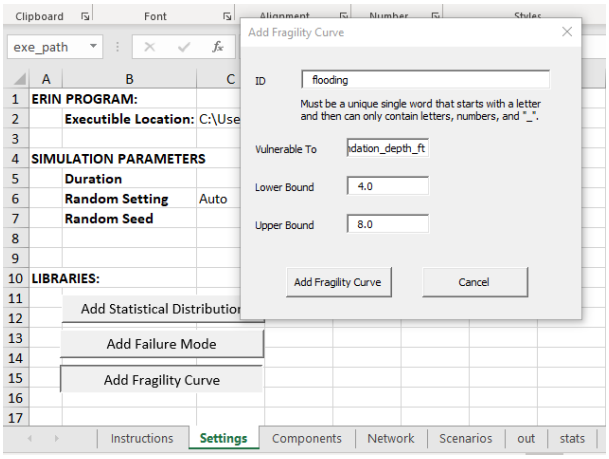

Figure 14: Add Fragility Curve #2

3. Next we'll begin filling in the Components as shown in Figure 15, 16, 17, 18, 19, 20, 21, 22, 23.

4. With all the components added, move on to the "Network" sheet. We must add a network link between the "utility" location and the "b1" (building #1) location as shown in Figure 24, 25.

5. Finally, add the scenarios and intensity values as shown in Figure 26, 27, 28, 29, 30, 31, 32, 33, 34.

6. Finally, hit the run button to simulate the network.

9 When Things Don't Work: Checklist for MS Excel UI

The Microsoft Excel user interface to the *ERIN* engine is a convenient way to access the power of the engine without having to manually write an input file. However, the decoupling between

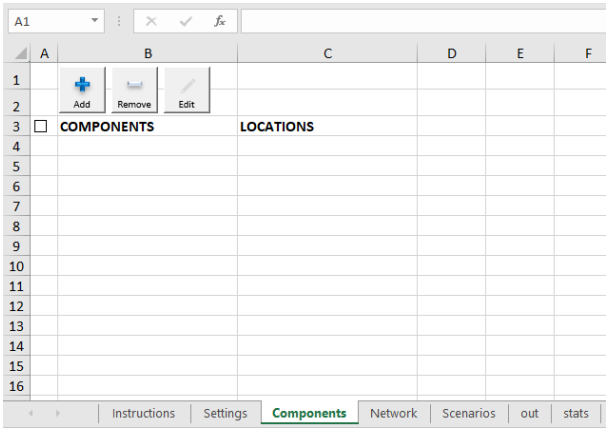

Figure 15: Add Components

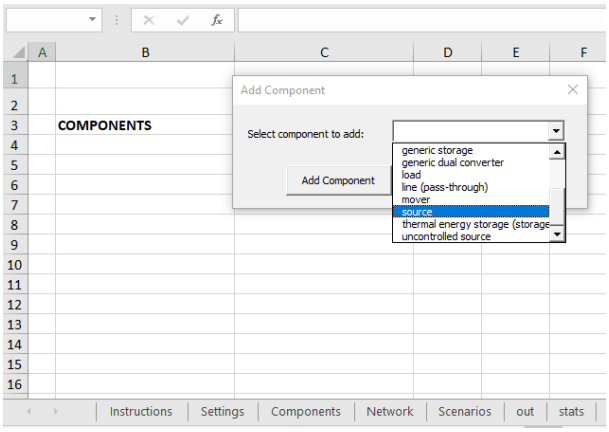

Figure 16: Add Source Components

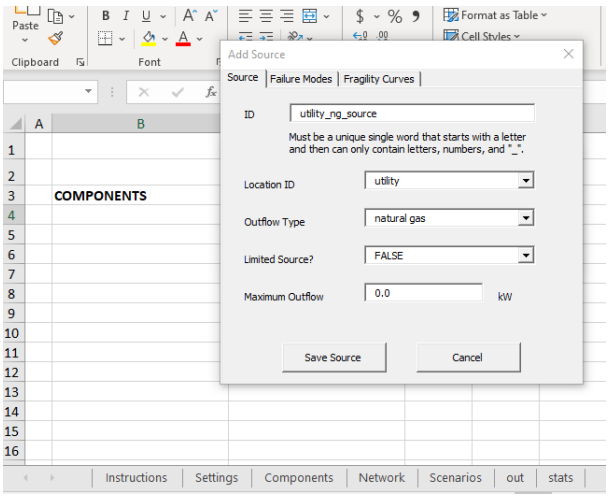

Figure 17: Add Source Component Dialogue

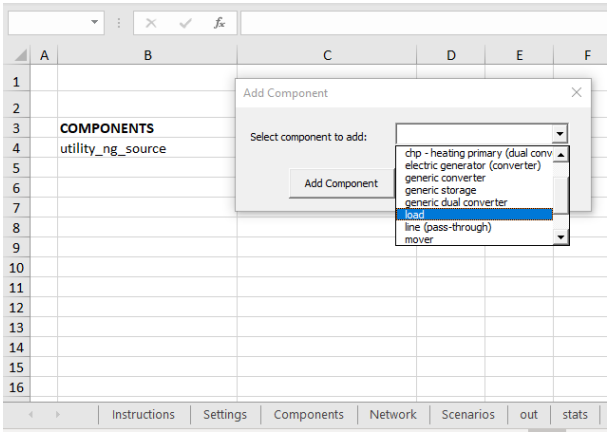

Figure 18: Add Load Component

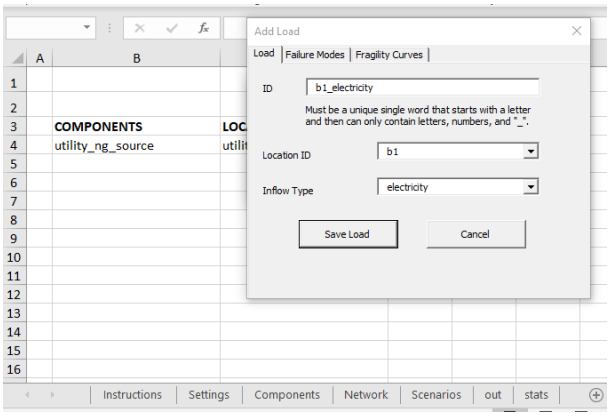

Figure 19: Add Load Component Dialogue

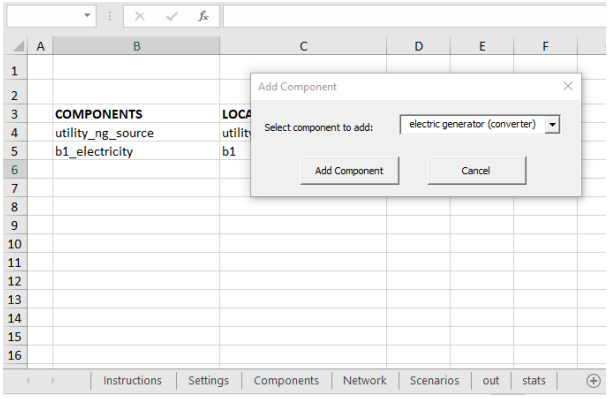

Figure 20: Add Converter Component

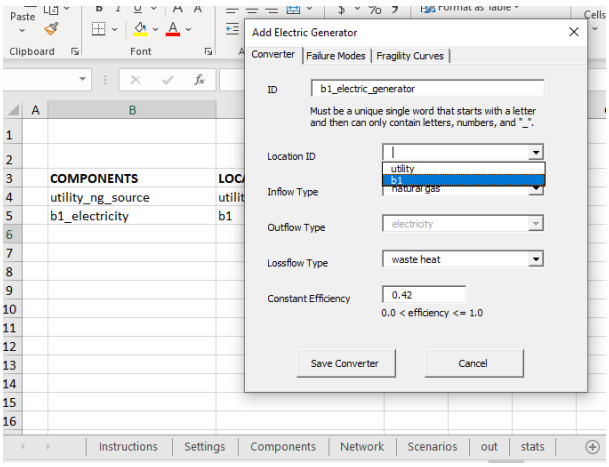

Figure 21: Add Converter Component Dialogue

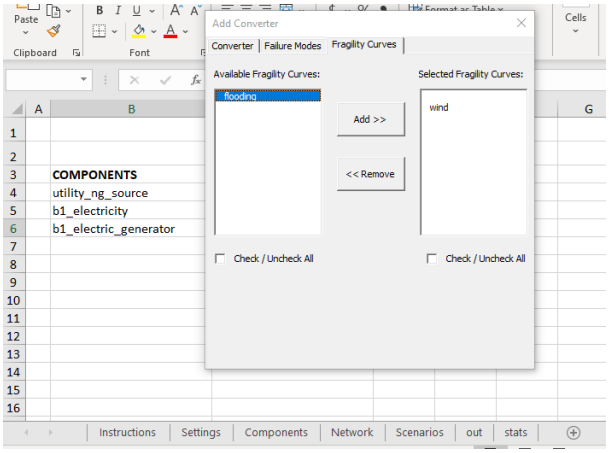

Figure 22: Add Fragility Curves to Converter Components

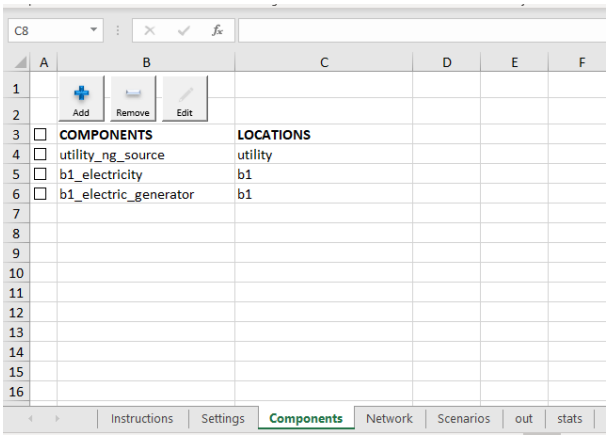

Figure 23: All Components Added

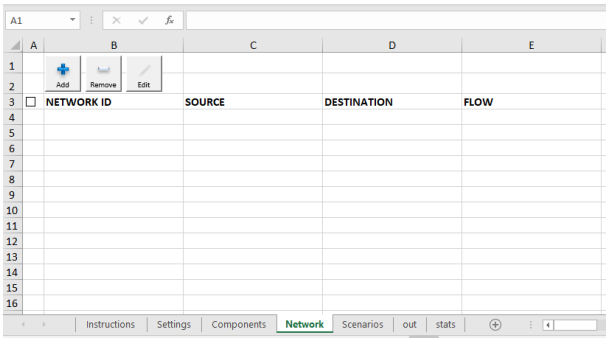

Figure 24: The Network Tab

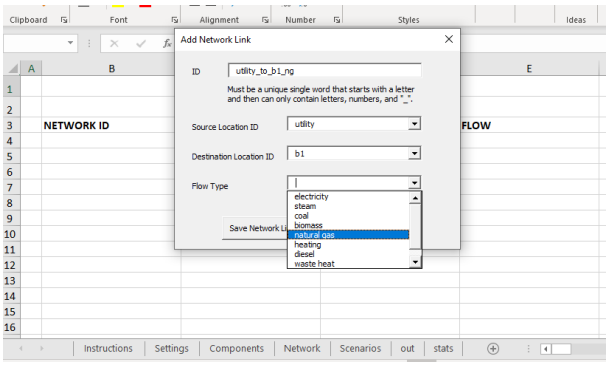

Figure 25: Adding a Network Link from `utility` to `b1`

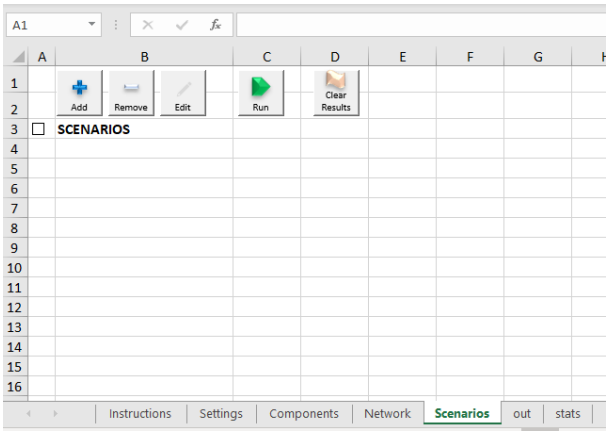

Figure 26: The Scenario Sheet

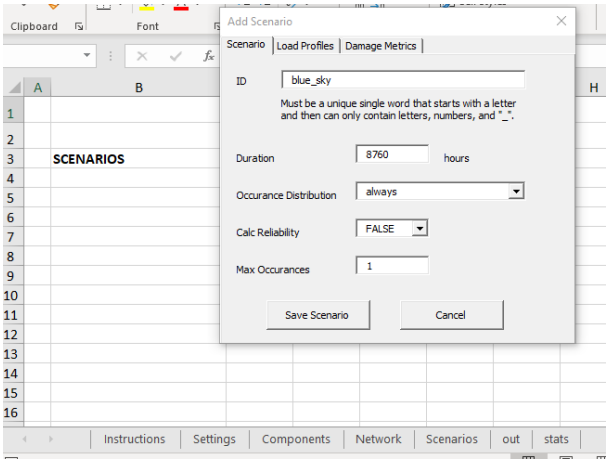

Figure 27: Adding the Blue Sky Scenario

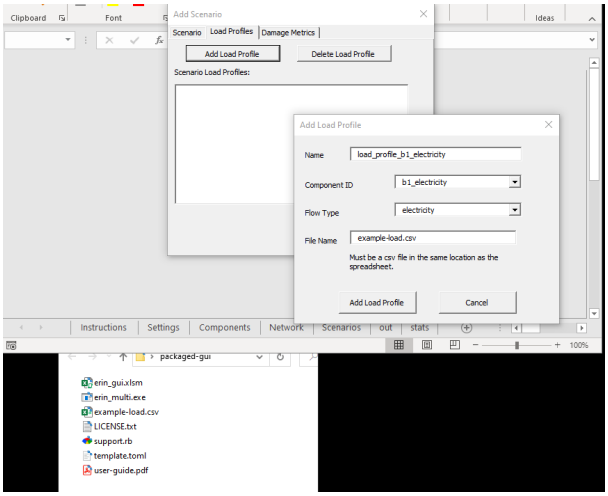

Figure 28: Adding the Load Profile for Blue Sky

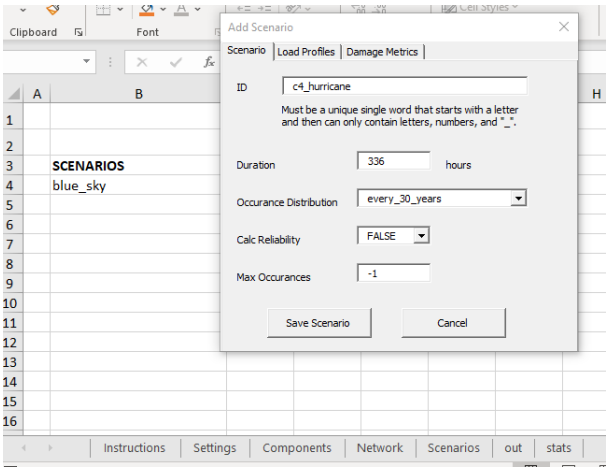

Figure 29: Adding the Hurricane Scenario

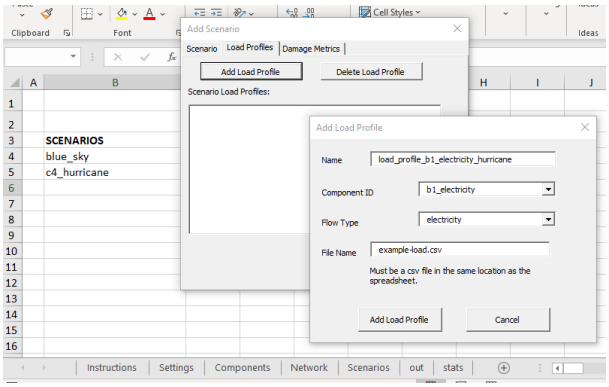

Figure 30: Adding the Load Profile for Hurricane

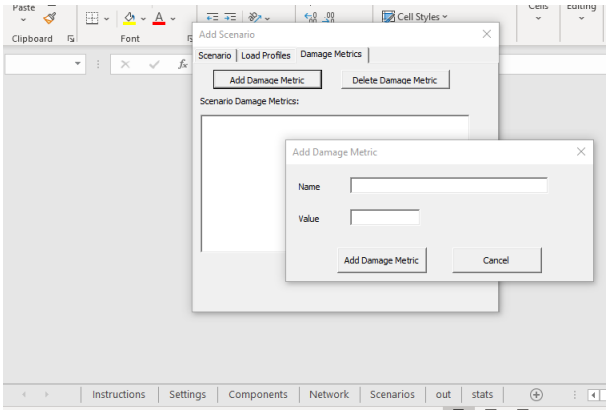

Figure 31: The Damage Metric UI

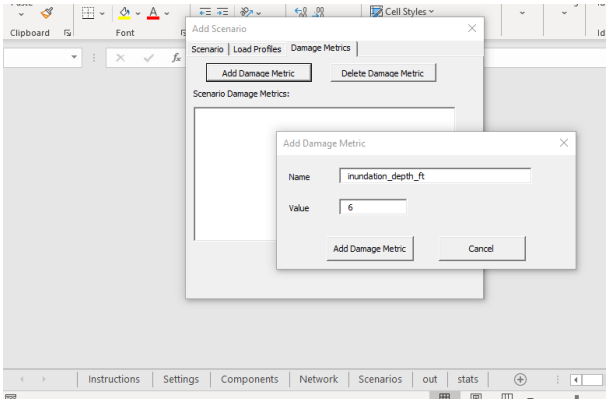

Figure 32: Adding Inundation Depth in Feet

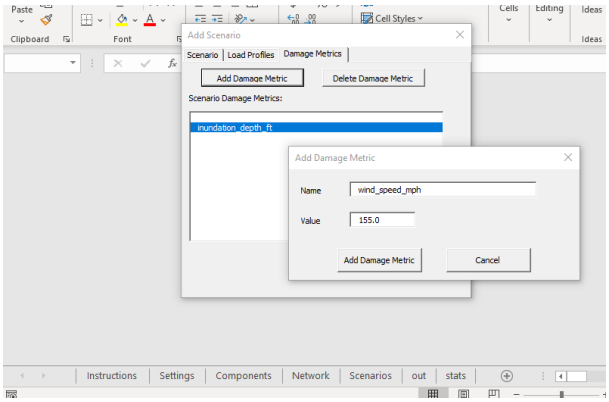

Figure 33: Adding Wind Speed in MPH

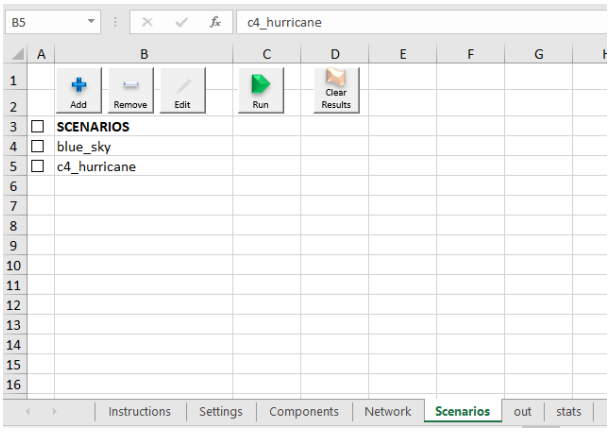

Figure 34: The Finished Scenarios

Excel and the *ERIN* engine adds just enough complexity to the picture that it can be difficult to determine what's wrong when things don't work as they should. This section provides a series of checks to help you methodically identify the source of any potential problems and either correct it yourself or help us to quickly pinpoint the issue. These instructions were written assuming you are using a recent version of Microsoft Windows as that is the only operating system supported by the MS Excel User Interface.

- **Are you using MS Windows?** Unfortunately, the link between Microsoft Excel and the *ERIN* engine is only supported on Microsoft Windows at this time. If you are not using Microsoft Windows, you can still use the simulation engine directly or through the *Modelkit* template generation capability.
- **Do you have Microsoft Excel installed?** This may be obvious, but ensure that you actually have Microsoft Excel installed. We are currently using Microsoft Excel for Office 365. Any recent version of Microsoft Excel should work, but if you do have to submit a bug report, please note the version of Microsoft Excel you are using. Unfortunately, other spreadsheet programs that can open and edit Microsoft Excel files such as Open Office or Libre Office do not work.
- **Have you installed *Modelkit Catalyst*?** *Modelkit Catalyst*, a free download available from Big Ladder Software, is a required pre-requisite for using the MS Excel User Interface. If you haven't already installed it, please download it from https://bigladdersoftware.com/projects/modelkit/.
- **Is *Modelkit Catalyst* working correctly?** To determine if *Modelkit* is installed correctly, go to the Windows start menu and type `cmd`. This should bring up the option to launch the "Command Prompt". Launch the "Command Prompt" and type `modelkit`. You should see a help text return to you giving an overview of the *Modelkit* commandline interface options. If you do not see this, right-click on your start menu button and select "settings" from the menu. Type "Add or remove programs" in the search dialogue. If you do not see "Modelkit Catalyst" in the list of installed programs, you have not yet installed *Modelkit* so stop here and download and install *Modelkit Catalyst*. If you do see "Modelkit Catalyst", try uninstalling and reinstalling the program and try using the instructions again for the MS Excel UI example. If you are still having problems, please submit a bug report for support.
- **Is the *ERIN* simulation engine working correctly?** Next, we need to confirm that the *ERIN* simulation engine is working correctly. Unzipped a *fresh* folder from the original file zip archive to ensure there are no changes you made that are creating issues. You can do this by right clicking on the zip archive and selecting "Extract All...". Next, open "File Explorer". If you are unfamiliar with how to do this, click on the Start menu and type "File Explorer" and open the application called "File Explorer". In File Explorer, navigate to that extracted (i.e., unzipped) folder. This folder should contain the `iea-annex-73-tool.xlsm`, `erin_multi.exe`, and several other files including this User's Guide. In the address bar of File Explorer, type `CMD` to start a new command shell in that folder (or use any other method of opening a command shell at the folder of the unzipped contents). In the "Command Prompt" that is opened, type `erin_multi.exe`. You should see a usage message that also lists the version of the simulator. If you do not see this, please submit a bug report for support.
- **Did you enable macros when opening the Microsoft Excel workbook?** Visual Basic for Applications Macros are required when using the Microsoft Excel User Interface. When you opened the `xlsm` file, you should have been prompted with "SECURITY WARNING: Macros have been disabled." You will need to hit the button to "Enable Content" in order to use the interface.
- **Is the Microsoft Excel User Interface workbook in the same folder as `erin_multi.exe`, `support.rb`, and `template.toml`?** If you are working from a freshly

extracted (i.e., unzipped) folder, all the required files should be in the same folder together. If you are not, please do start again from a freshly extracted folder to help us methodically debug the issue.

- **Have you added the complete path to *ERIN* simulation engine in cell C2 of the "Settings" sheet?** Even if you have already done this, it can't hurt to double check and cut and paste the path again. Again, we recommend that you use the *freshly* extracted Excel workbook at this point (versus one you've already been typing in). The best way to get the path to the `erin_multi.exe` simulation engine is to go to the freshly extracted folder using File Explorer, hold down the SHIFT key while right clicking on "`erin_multi.exe`", and select the option "Copy as path". Then, open the Microsoft Excel User Interface and paste (CTRL+V) the path into cell C2 of the "Settings" sheet. Note: even if you have spaces in your path, do not add quotes around your path. The tool automatically surrounds the path in "double quotes" and if you add your own double quotes in cell C2, that will actually cause an error. The path you paste in must be unquoted.
- **Lastly, have you followed the details of the example tutorial *exactly*?** There are known issues where you can formulate an incorrect problem using the Excel User Interface. Common issues include:
  - not having a proper load header: did you use the `example-load.csv` file as a reference?
  - specifying an immediately occurring scenario with no occurrence limit: an occurrence distribution of "ALWAYS" must be accompanied by a Max Occurrence of 1 or more.
  - is there a network link between sources and load locations?

If, after exhausting all of the above checks, you are still having issues, please do not hesitate to contact us for support.