DREAMIN'SG Integrated Power-Water-Transportation Simulation

User Manual v1.0

Srijith Balakrishnan
Beatrice Cassottana
Felix Kottmann

July 23, 2021

TABLE OF CONTENTS

CHAPTER

ONE

INTRODUCTION

This package is developed as part of the Disaster REsilience Assessment, Modelling, and INnovation Singapore (DREAMIN'SG) project at the Future Resilient Systems of the Singapore ETH Centre. The DREAMIN'SG project is funded by the National Research Foundation, Singapore under the Intra-CREATE grant program. In the wake of increasing threats posed by climate change on urban infrastructure [Nissen and Ulbrich (2017)], this project aims at studying the effects of policy interventions and network characteristics on the resilience of urban infrastructure networks. Given the critical nature of the above infrastructure systems, their interdependencies need to be considered in pre- and post-disaster resilience actions.

In specific, the DREAMIN'SG project envisages to build a methodology to assess and predict the resilience of urban infrastructure systems and propose new pathways to develop innovative technologies and services for its improvement. The urban infrastructure system is modeled as an interdependent power-, water-, and transportation network that interact with each other before, during and after a disaster. The researchers are developing an integrated simulation model to study the performance of the interdependent infrastructure network under various disruption and recovery scenarios. Based on the simulation-generated datasets, machine learning algorithms would be implemented to understand the causal relationship between topological and policy-related interventions and disaster risks. Building on the understanding of success features that make an urban system resilient, Design Science approaches will be used to develop new solutions that mitigate the consequences of disruptions and accommodate constraints in the analysed case studies. The results of the research will support local governments and system managers to improve infrastructure resilience against weather-related disruptions. The methodology adopted in the study is presented in Fig. 1.

The steps of the project are summarized as follows:

1. Multiple scenarios are generated by considering different disruptions, network models, technological constraints and system configurations.
2. A simulation model is created for the interdependent power grid, water distribution system,

and road transportation system.

3. Resilience is assessed based on the simulated performance of the three systems.
4. An interpretable machine learning algorithm is implemented to analyze the scenarios and extract information related to key system features that influence resilience.
5. The identified system features inform the design of new services, technologies, and products that are able to simultaneously enhance resilience and accommodate the technological constraints.

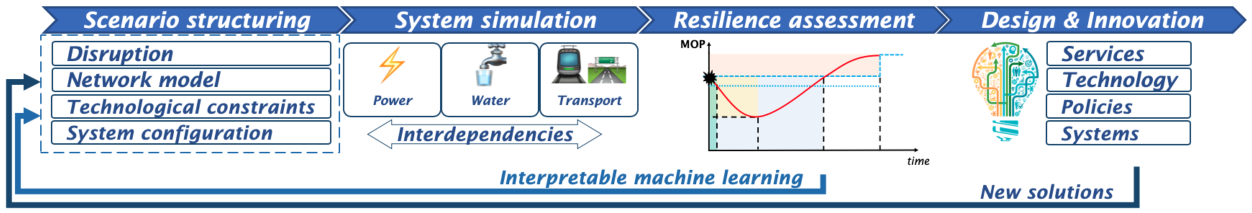


Fig. 1: Methodological framework of DREAMIN'SG project

For further information and updates on the project, please visit the DREAMIN'SG project webpage[1]. In the rest of the report, the details of the interdependent infrastructure simulation platform, including its modeling, installation, and usage are discussed.

[1]Disaster REsilience Assessment, Modelling, and INnovation Singapore https://frs.ethz.ch/research/projects/dreamin_sg.html

CHAPTER

TWO

INTEGRATED SIMULATION PLATFORM

The integrated simulation model has been developed as a Python-based package consisting of modules for simulation of system- and network-level cascading effects resulting from component failures. The overall structure of the integrated simulation platform is illustrated in Fig. 2.

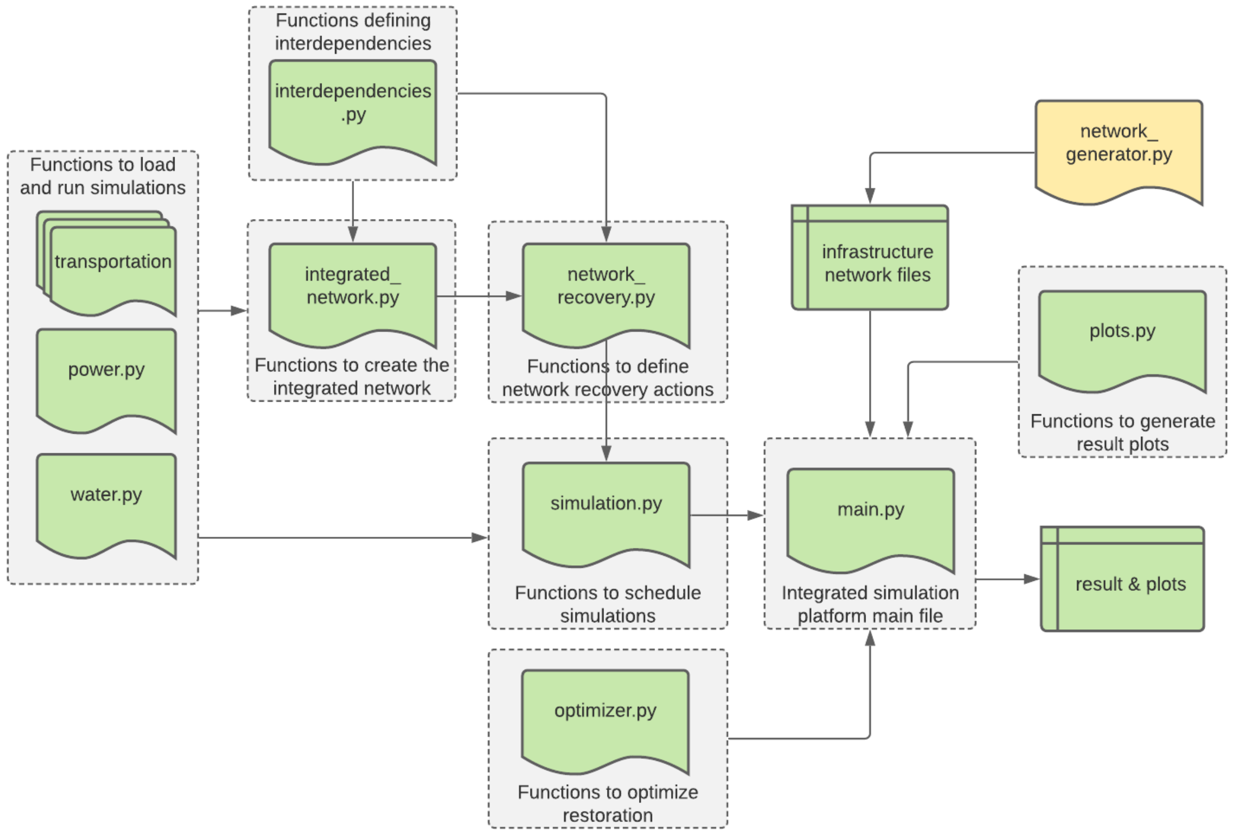

Functions defining interdependencies
interdependencies .py
network_ generator.py
Functions to load and run simulations
transportation
power.py
water.py
integrated_ network.py
Functions to create the integrated network
network_ recovery.py
Functions to define network recovery actions
infrastructure network files
plots.py
Functions to generate result plots
simulation.py
Functions to schedule simulations
main.py
Integrated simulation platform main file
result & plots
optimizer.py
Functions to optimize restoration

Fig. 2: DREAMIN'SG integrated simulation platform structure

The model is capable of initializing disaster scenarios in interdependent power, water and transportation networks and evaluating resilience strategies by generating operational performance curves (Fig. 3). The resilience strategies that can be tested include pre-disaster interventions, such as system redundancy enhancements and post-disaster recovery optimization.

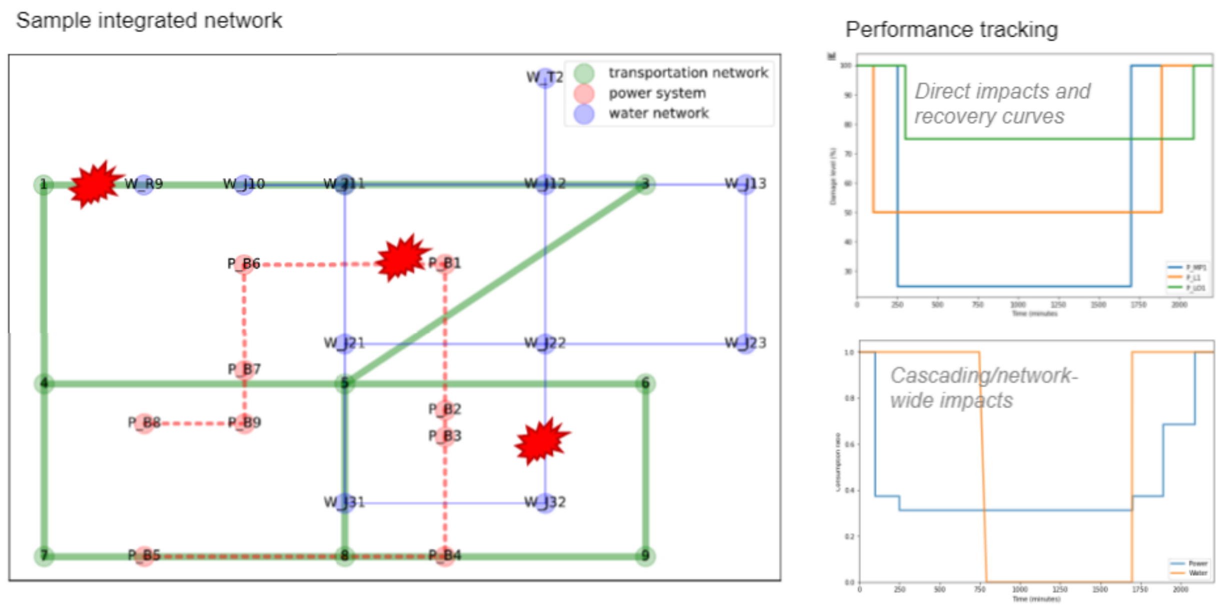


Fig. 3: Implementation of the simulation platform to generate performance curves

The model is developed by integrating existing flow-based water, power and transportation network models. The whole model can be divided into three broad modules, namely, the integrated infrastructure network module, network recovery module, and the recovery optimization module.

2.1 Integrated infrastructure network

This module houses the three infrastructure network models which are used to simulate power-, water-, and transportation networks independently. These are developed using existing Python-based packages. In order to model the power network, `pandapower` was used [Thurner *et al.* (2018)]. The water distribution network is modeled using `wntr` package [Klise *et al.* (2020)]. The traffic network is modeled using static traffic assignment method [Boyles *et al.* (2020)]. The details of the packages are presented in Table 1

The module also consists of an interdependency sub-module which serves as an interface between infrastructure network pairs. Currently the following dependencies are considered in the interdependent simulation platform.

1. Power-water dependencies include dependency of water pumps on electric motors and generators on reservoirs.

2. Dependencies also exist between traffic network and the other two infrastructure models, as the former provides access to the latter. The disruptions to transportation infrastructure components are their recovery are key considerations that affect the restoration and recovery of all other infrastructure networks. The module also stores the details of the states of all network components, including their operational status after a disaster.

Table 1: Infrastructure packages using in the simulation model

Infrastructure	Package	Capabilities
Power	`pandapower`	• Capable of generating power networks with standard components such as lines, buses, and transformers based on design data. • Capable of performing power-flow analysis.
Water	`wntr`	• Capable of generating water networks with standard components such as pipes, tanks, and nodes based on design data. • Capable of performing pressure dependent demand or demand-driven hydraulic simulations.
Transportation	static traffic assignment package	• Capable of implementing static traffic assignment and computing travel times between origin-destination pairs.

2.2 Network recovery

The recovery module consists of functions to develop an event table to schedule disruptive events and restoration actions after a disaster is initiated in the model. The simulation platform uses this table as a reference to modify the operational status of network components during a simulation, so that the consequences of disaster events and repair actions are reflected while simulating network performance. The recovery module also stores the details such as the number of repair crew for every infrastructure network, their initial locations, etc.

2.3 Recovery optimization

This module determines the order in which the repair actions are carried out. Currently, the approach of the optimization module leverages on the methodology of Model Predictive Control (MPC) [Camacho and Bordons (2007)]. In this approach, first, out of N repair steps, the solution considering only k steps, called the prediction horizon, is computed. Next, the first step of the obtained solution is applied to the system and then the process is being repeated for the remaining

N-1 components until all components have been scheduled for repair. In the context of the integrated infrastructure simulation, the optimizer module evaluates repair sequences of the length of the prediction horizon for each infrastructure (assuming that each of the infrastructure has a separate recovery crew) based on a resilience metric. In the model, the integral loss of service (ILOS) is used as the resilience metric. The ILOS is calculated as follows:

$$ILOS = w_P \sum_k \Delta Power \times dt_P(k) + w_W \sum_k \Delta Water \times dt_w(k) + w_T \sum_k \Delta Transport \times dt_T(k) \tag{2.1}$$

where, $\Delta Power$, $\Delta Water$ and $\Delta Transport$ are the respective demands not served, $dt_P(k)$, $dt_W(k)$ and $dt_T(k)$ the repair-time specific time steps between the repair actions and w_P, w_W, w_T the weights. The optimal repair sequence is found by minimizing the ILOS. At this stage, the optimal repair action in each prediction horizon is computed using a brute-force approach where the ILOS is evaluated for each of the repair sequences.

2.4 Work in progress

Currently, the network data, interdependency data and infrastructure disruption data are to be manually fed into the model to run the network simulations. However, efforts are being made to include separate modules for network generation and hazard initiation in the simulation platform, to enhance the scope of the model. The following improvements will also be made to the model:

1. Realistic policies for network recovery.
2. Additional interdependencies.
3. More efficient repair optimization algorithms

CHAPTER

THREE

MODEL DATA REQUIREMENTS

Currently, network generation and hazard generation are not automated in the integrated simulation model. Therefore, in order to use the model, three types of data are to be manually fed into the model, namely the water, power and transportation networks, interdependency data, and infrastructure disruption data. There datasets need to be in specific formats which are compatible with the model. In this chapter, the details such as file formats and information to be included in the input files are discussed in detail.

3.1 Infrastructure networks

The infrastructure network data must be compatible with the respective infrastructure model packages enlisted in Table 1. In addition, disruptions to certain components belonging to the infrastructure systems are not supported as of now. The individual networks must be constructed taking into the above aspects into consideration.

3.1.1 Water distribution system

Since the integrated simulation model handles the water distribution network models using `wntr` package, the input file must be in `.inp` format. Some examples of water network files can be found in `wntr` Github repository[1]. The water network can be either built using EPANET or `wntr` package. Currently, the water network simulation is performed using `WNTRSimulator` in the `wntr` package. Therefore, there are certain exceptions which must be considered while generating the water network file. These exceptions can be found in `wntr`'s software framework and limitations[2] page.

[1] https://github.com/USEPA/WNTR/tree/main/examples/networks

[2] Software framework and limitations. https://wntr.readthedocs.io/en/latest/framework.html#discrepancies

In addition, the integrated simulation model identifies the component details such as infrastructure type, component type, etc. using the component names. The name of the components must follow the nomenclature. The details of nomenclature and whether a component can be failed in the model is presented in Table 2. For example, a water pump can be named as `W_WP1`. Integers must be followed by the prefix to name components belonging to same category.

Table 2: Nomenclature and disruption functionality of supported water system components

Prefix	Component name	Disruption supported
W_WP	Pump	Yes
W_R	Reservoir	No
W_P	Pipe	Yes
W_J	Junction	No
T	Tank	Yes

3.1.2 Power system

The power system is modeled using `pandapower` package, and therefore the network file must be in `*.json` format. Several tutorials for generating power system networks using `pandapower` are available in the package's tutorial page[3]. Similar to water network components, naming of power system components must also follow the stipulated nomenclature presented in Table 3. Currently, power networks are modeled as three-phase systems. Therefore, single-phase components in `pandapower` are not supported. Integers must be followed by the prefix to name components belonging to same category.

Table 3: Nomenclature and disruption functionality of supported power system components

Prefix	Component name	Disruption supported
P_B	Bus	Yes
P_LOA	Asymmetric load	Yes
P_LOMP	Motor as load	Yes
P_AL	Asymmetric load	Yes
P_AS	Asymmetric static generator	No
P_SW	Switch	Yes
P_EG	External Grid	Yes
P_L	Line	Yes
P_TF	Transformer	Yes

[3]https://github.com/e2nIEE/pandapower/tree/develop/tutorials

3.1.3 Transportation system

The transportation system is simulated using the static traffic assignment model developed by Prof. Stephen Boyles of Department of Civil, Architectural and Environmental Engineering, The University of Texas at Austin. The transportation network data must be in the TNTP data format. More details on the format and example networks can be found in Ben Stabler's Github page[4]. The naming of the transportation network components must follow the nomenclature presented in Table 4. Integers must be followed by the prefix to name components belonging to same category.

Table 4: Nomenclature and disruption functionality of supported transportation system components

Prefix	Component name	Disruption supported
T_J	Junction	No
T_L	Link	Yes

3.2 Infrastructure dependencies

Data related to water-power dependencies[5] between infrastructure components must be provided separately in a `.csv` file. The file must contain `water_id` and `power_id` fields which represent the water- and power component names (Table 5). The model will determine the type of the water- and power components and construct the dependencies accordingly.

Table 5: Format of dependency data file

water_id	power_id name
W_WP9	P_MP1
W_R9	P_G3

3.3 Infrastructure disruption data

The third data input is related to disrupted components. Similar to the dependency input file, the disruption data also needs to be provided in `*.csv` format. The file should have three fields, namely `time_stamp` (time of disruption in seconds), `components` (name of the component that is included in any of the three networks), and `fail_perc` (percentage of damage). An example for the dependency data is presented in Table 6.

[4]https://github.com/bstabler/TransportationNetworks

[5]The dependencies of power and water networks on transportation network are identified internally by the model and therefore that information need not be provided in the dependency data file.

Table 6: Disruptive event table generated from dependencies input file

	time_stamp	components	fail_perc
0	15000	P_MP1	75
1	6000	T_L2	50

While the percentage of damage of component is not used in the current model, this may be incorporated in the model to find the repair duration estimate in the future versions.

CHAPTER

FOUR

INSTALLATION

4.1 Stable release

To install dreaminsg-integrated-model, run this command in your terminal[1]:

```
$ pip install dreaminsg_integrated_model
```

This is the preferred method to install dreaminsg-integrated-model, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

4.2 From sources

The sources for dreaminsg-integrated-model can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/srijithabalakrishnan/dreaminsg_integrated_
↪model
```

Or download the tarball:

```
$ curl -OJL https://github.com/srijithabalakrishnan/dreaminsg_integrated_
↪model/tarball/master
```

You may need to create a new Python environment that has all required packages and dependencies installed before start using the package. Run the following comment.

[1]Currently, the package is not uploaded to PyPI repository. However, it will be done in the near future.

```
$ conda env create --name ENV_NAME --file=environment.yml
```

Once you have a copy of the source, set the project folder as working directory and you can install the package with:

```
$ conda activate redcar
$ pip install -editable .
```

CHAPTER

FIVE

GETTING STARTED

This chapter provides a step-by-step tutorial for running a model predictive control (MPC) method to identify the optimal repair strategy after a disruptive event.

It is suggested to create a Jupyter notebook inside the notebooks folder as below:

```
$ cd notebooks
$ conda activate redcar
(redcar)$ conda install ipykernel
(redcar)$ ipython kernel install -user -name=redcar
```

The browser would open showing available notebook kernels. Select `redcar` from options to open a Jupyter notebook that can run the integrated simulation model.

5.1 Load required modules and libraries

Now run the following IPython extensions to reload all modules within the simulation package and set the Jupyter notebook.

```
%load_ext autoreload
%autoreload 2

from IPython.core.display import display, HTML
display(HTML("<style>.container  width:100% !important; </style>"))
```

If the simulation package was built from source (by downloading from Github), it is required to load the required modules and external libraries manually as follows:

```
from pathlib import Path

from dreaminsg_integrated_model.src.network_recovery import *
import dreaminsg_integrated_model.src.simulation as simulation
from dreaminsg_integrated_model.src.network_sim_models.integrated_network␣
↪import *
from dreaminsg_integrated_model.src.network_sim_models.interdependencies␣
↪import *
from dreaminsg_integrated_model.src.optimizer import *

import dreaminsg_integrated_model.src.plots as model_plots
```

5.2 Load individual networks and create an IntegratedNetwork object

In order to create the integrated infrastructure model, first we need to initiate the `IntegratedNetwork` object from the `integrated_network` module. Then the individual models should be loaded one by one to the `IntegratedNetwork` object. For this study, we use a simplified integrated network consisting of a few nodes and links.

```
# initiate the integrated network object
simple_network = IntegratedNetwork()

# specify locations of network files
MAIN_DIR = Path('..')
network_dir = 'in2'

water_file = MAIN_DIR/f'dreaminsg_integrated_model/data/networks/
↪network_dir/water/Example_water2.inp'
power_file = MAIN_DIR/f'dreaminsg_integrated_model/data/networks/
↪network_dir/power/Example_power.json'
transp_folder = MAIN_DIR/f'dreaminsg_integrated_model/data/networks/
↪network_dir/transportation/'

# load all infrastructure networks
simple_network.load_networks(water_file, power_file, transp_folder)
```

Next, the individual infrastructure networks will be converted into a NetworkX object for plotting purposes.

```
simple_network.generate_integrated_graph(plotting = True)
```

The above method will create the object and outputs the integrated graph (Fig. 4) consisting of the topologies of all the three infrastructure networks that were loaded.

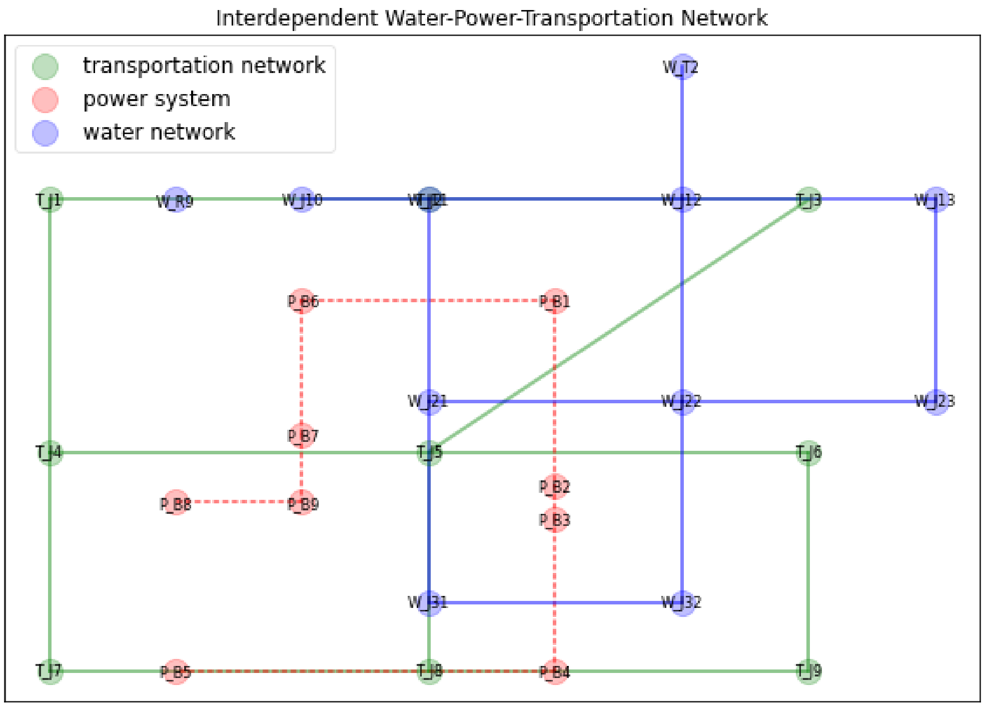


Fig. 4: The integrated infrastructure network in the current example

In the next step, we need to build the interdependencies within the integrated network. In this model, three types of interdependencies are considered.

1. Water pump on electric motor (water system on power system dependency)
2. Power generator on reservoir (power system on water system dependency)
3. All power and water system components on nearest transportation network node (power and water systems on transportation system dependency)

The information related to the first two types of dependencies must be provided in the form of csv file whereas the third set of dependencies will be automatically identified by the simulation model.

```
# location of the dependency file
dependency_file = MAIN_DIR/f"dreaminsg_integrated_model/data/networks/
↪network_dir/dependecies.csv"
```

(continues on next page)

(continued from previous page)

```
simple_network.generate_dependency_table(dependency_file =␣
↪dependency_file)
```

The dependencies are referenced using two pandas dataframes in the model.

- `wp_table` stores water - power dependencies.
- `access_table` stores transportation access dependencies.

In order to view the `wp_table`, the following line of code may be implemented. It will return the table shown in Table 7.

```
simple_network.dependency_table.wp_table.head()
```

Table 7: Water-power dependencies

	water_id	power_id	water_type	power_type
0	W_WP9	P_MP1	Pump	Motor
1	W_R9	P_G3	Reservoir	Generator

Similarly, the `access_table` can be printed which will return a table as shown in Table 8.

```
simple_network.dependency_table.access_table.head()
```

Table 8: Transportation access dependencies

	origin_id	transp_id	origin_cat	origin_type	access_dist
0	P_B8	T_J4	power	Bus	125.00
1	P_B7	T_J5	power	Bus	103.08
2	P_B5	T_J7	power	Bus	100.00
3	P_B4	T_J8	power	Bus	100.00
4	P_B6	T_J2	power	Bus	180.28

5.3 Set disrupted components

The information regarding the dursuptive events are also stored in the `IntegratedNetwork` object. The data related to disrupted components and the level of damage must be provided in csv file format.

```
scenario_file = MAIN_DIR/"dreaminsg_integrated_model/data/
↪disruptive_scenarios//.csv".format('test1', 'motor_failure_net1')

simple_network.set_disrupted_components(scenario_file=scenario_file)
```

In this example, two components in the integrated network are failed, namely an electric motor connected to water pump (`P_MP1`) and a transportation link (`T_L2`). The disruptive events table can be returned using the following code.

```
simple_network.get_disruptive_events()
```

The returned pandas dataframe would like the one below (Table 6). It shows the time (`time_stamp`) in seconds at which the component (`components`) failed and the intensity of damage in percentage (`fail_perc`).

If there are any disrupted water pipe components, leak nodes are to be added to simulate pipe leaks. This is performed using an inbuilt method within the `IntegratedNetwork` object.

```
simple_network.pipe_leak_node_generator()
```

5.4 Set initial locations of restoration crew

To perform the restoration and recovery of damaged or failed components after disruptive event occurs, each of the infrastructure agency has a repair crew. Once the disaster hits, each of these crews are sent to the locations of the damaged infrastructure components based on a recovery strategy. To set the initial locations of the water- power-, and transportation system repair crews, the following method is initiated.

```
simple_network.set_init_crew_locs(init_power_loc='T_J8',␣
↪init_water_loc='T_J8', init_transpo_loc='T_J8')
```

In the current example, for simplicity and demonstration purpose, the initial locations of all the three repair crews are set as `'T_J8'`, which is a transportation node (junction).

5.5 Optimization of restoration actions

The next step is to create a NetworkRecovery object. The NetworkRecovery object stores recovery related information including the start times and end times of various repair actions.

```
network_recovery = NetworkRecovery(simple_network, sim_step=60)
```

Now, a simulation object is created which include methods to perform actions to simulate direct and indirect impacts in the network due to the external event as well as recovery actions. The simulation time step parameter is set to be the initial time-step of the `wntr` water network model, which is 60 seconds.

```
# initial_sim_step which will be updated during the simulation
sim_step = simple_network.wn.options.time.hydraulic_timestep

# NetworkSimulation object
bf_simulation = simulation.NetworkSimulation(network_recovery, sim_step)
```

Finally, it is time to identify the optimal repair order and simulate the interdependent effects based on the repair schedules. For this a `BruteForceOptimizer` object is created with a prediction horizon of one and the model predictive control (MPC) optimization algorithm is applied on the `NetworkSimulation` object. This is implemented using the following lines of code.

```
bf_optimizer = BruteForceOptimizer(prediction_horizon = 1)
bf_optimizer.find_optimal_recovery(bf_simulation)
```

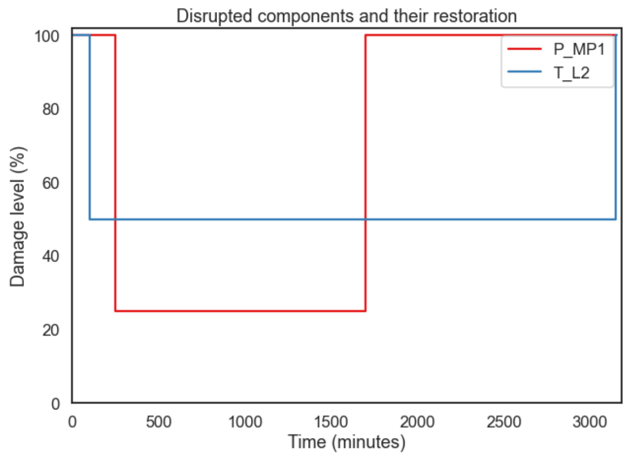
5.6 Plot the results

The direct and the network-wide effects can be plotted as performance curves as follows:

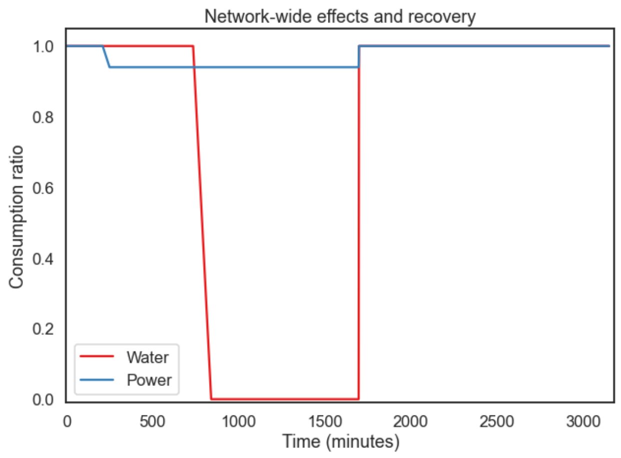
```
# plot repair curves
model_plots.plot_repair_curves(network_recovery, scatter = False)

# plot interdependent effects
time_tracker, power_consump_tracker, water_consump_tracker = bf_optimizer.
↪get_trackers()
model_plots.plot_interdependent_effects(power_consump_tracker,␣
↪water_consump_tracker, time_tracker, scatter=False,)
```

The following plots will be generated by the codes (Fig. 3). The first code shows the level of damage to each of the directly affected component (any performance value less than 100% assumes that the component service is disrupted). The second plot show the interdependent effects of the initial disruption and the subsequent recovery actions. Every restoration strategy produces unqiue performance plots. For optimal resilience strategy, the area under the curve must be maximized (i.e., minimum ILOS)



(a) Disrupted components and their restoration



(b) Network-wide effects

Fig. 5: Output plots from the integrated simulation model

5.7 Save the results

The data related to interdependent effects can be stored to local directory as follows:

```
location = MAIN_DIR/"dreaminsg_integrated_model/data/disruptive_scenarios/
↪test1"
bf_simulation.write_results(time_tracker, power_consump_tracker,␣
↪water_consump_tracker, location, plotting=False)
```

CHAPTER

SIX

SUPPORT

The model is still in development stage. To report any issues in the model, suggest changes, or obtain support with installation or use, please write to srijith.balakrishnan[at]sec.ethz.ch

BIBLIOGRAPHY

[Boyles *et al.* (2020)] **Boyles, S. D.**, **N. E. Lownes**, and **A. Unnikrishnan**, *Transportation Network Analysis*, volume 1. 2020, 0.85 edition.

[Camacho and Bordons (2007)] **Camacho, E. F.** and **C. Bordons**, Introduction to model predictive control. *In Advanced Textbooks in Control and Signal Processing*, 9781852336943. Springer International Publishing, 2007, 1–11.

[Klise *et al.* (2020)] **Klise, K.**, **D. Hart**, **M. Bynum**, **J. Hogge**, **T. Haxton**, **R. Murray**, and **J. Burkhardt** (2020). Water network tool for resilience (wntr) user manual. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

[Nissen and Ulbrich (2017)] **Nissen, K. M.** and **U. Ulbrich** (2017). Increasing frequencies and changing characteristics of heavy precipitation events threatening infrastructure in Europe under climate change. *Natural Hazards and Earth System Sciences*, **17**(7), 1177–1190. ISSN 16849981.

[Thurner *et al.* (2018)] **Thurner, L.**, **A. Scheidler**, **F. Schafer**, **J. H. Menke**, **J. Dollichon**, **F. Meier**, **S. Meinecke**, and **M. Braun** (2018). Pandapower - An open-source Python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Transactions on Power Systems*, **33**(6), 6510–6521. ISSN 08858950.