VAWS user manual

Release 2.0

Geoscience Australia

Feb 02, 2018

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER

ONE

INTRODUCTION

Vulnerability and Adaptation to Wind Simulation (VAWS) is a software tool that can be used to model the vulnerability of small buildings such as domestic houses and light industrial sheds to wind. The primary use-case of VAWS is the examination of the change in vulnerability afforded by mitigation measures to upgrade a building's resilience to wind hazard.

1.1 Background

Development of VAWS commenced in 2009-2010 in a collaborative project, partly funded by the then Department of Climate Change and energy Efficiency (DCCE), between Geoscience Australia, James Cook University and JDH Consulting. The development of the current version was undertaken as part of the Bushfire and Natural Hazard Cooperative Research Centre (BNHCRC) project "Improving the Resilience of Existing Housing to Severe Wind" led by James Cook University.

1.2 Overall logic

The VAWS tool takes a component-based approach to modelling building vulnerability. It is based on the premise that overall building damage is strongly related to the failure of key connections.

The tool generates a building model by randomly selecting parameter values from predetermined probability distributions using a Monte Carlo process. Values include component and connection strengths, external pressure coefficients, shielding coefficients, wind speed profile, building orientation, debris damage parameters, and component masses.

Then, for progressive gust wind speed increments, it calculates the forces in all critical connections using influence coefficients, assesses which connections have failed and translates these into a damage scenario and costs the repair. Using the repair cost and the full replacement cost, it calculates a damage index for each wind speed.

1.3 Key features

- Component-based approach:

  A house is modelled consisting of a large number of components, and overall damage is estimated based on damage of each of the components.

- Uncertainty captured through a Monte-Carlo process:

  Various uncertainties affecting house performance are modelled through a monte-carlo process.

- Inclusion of debris and water ingress induced damages:

  In addition to the damage to the connections by wind loads, debris and water ingress induced damages are modelled.

- Internal pressurisation:

  Internal pressure coefficients are calculated at each wind speed following the procedures of AS/NZS 1170.2 (Standards Australia, 2011) using the modelled envelope failures to determine envelope permeability.

1.4 Key uncertainties

The Monte Carlo process capture a range of variability in both wind loading and component parameters. The parameter values are sampled for each model and kept the same through the wind steps.

- Wind direction

  For each house, its orientation with respect to the wind is chosen from the eight cardinal directions either randomly, or by the user.

- Gust wind profile

  Variation in the profile of wind speed with height is captured by the random sampling of a profile from a suite of user-provided profiles.

- Pressure coefficients for zone and coverage

  Pressure coefficients for different zones of the house surfaces envelope are randomly chosen from a Type III (Weibull) extreme value distribution with specified means for different zones of the house envelope, and specified coefficients of variation for different load effects.

- Construction level

  Multiple construction levels can be defined with mean and cov factors which will be used to adjust the mean and cov of distribution of connection strength.

- Strength and dead load

  Connection strengths and dead loads for generated houses are sampled from lognormal probability distributions.

1.5 Caveats and limitations

VAWS has been designed primarily as a tool for assessing vulnerability of houses to wind hazard. The simulation outcomes should be interpreted as vulnerability of a group of similar houses on average, even though an individual house is modelled. In other words, the tool is not capable of predicting performance of each individual house for a specific wind event.

CHAPTER

TWO

GETTING STARTED

This chapter provides instructions on how to install and run the code for general users. Also it provides instructions for developers on how to install, test and build the package of the code. These instructions have been tested on *Windows 7*, *Linux*, and *OS 10.11.x* and is expected to work on most of modern operating systems.

2.1 Instructions for general users

2.1.1 Installation

The VAWS code currently runs with Python 2.7 with many dependencies. It is recommended to create a Python environment dedicated to the code without disrupting the existing environment. With conda, you can manage environments easily. Instructions below are based on conda, but virutalenv can be used alternatively.

1. Install Miniconda

Download and install Miniconda(https://conda.io/miniconda.html) with Python 2.7. This step can be skipped if either Miniconda or Anaconda with Python 2.7 is already installed.

- Windows
  - Double-click the downloaded *Miniconda2-latest-Windows-x86_64.exe* file.
  - When installation is finished, from the *Start* menu, open the *Anaconda Prompt*.
- Linux
  - In Terminal window, run

```
$ bash Miniconda2-latest-Linux-x86_64.sh
```

- Mac
  - In Terminal window, run

```
$ bash Miniconda2-latest-MacOSX-x86_64.sh
```

2. Create a conda environment.

In the terminal client, enter the following command to create the environment called *vaws_env*.

```
conda create -n vaws_env python=2.7
```

3. Activate the environment.

In the terminal client, enter the following to activate the environment.

- Windows

```
activate vaws_env
```

- Linux/Mac

```
source activate vaws_env
```

4. Install the code from conda channel

In the terminal client, enter the following to install the code.

```
conda install -c crankymax vaws
```

In case you see *PackageNotFoundError: Packages missing in current channels:* then enter the following in the terminal client and try above command again.

```
conda config --add channels conda-forge
```

2.1.2 Updating

In case new version of the code is available, you may update the code. The conda environment *vaws_env* should be activated first as *2.1.1 step 3*. And then enter the following commands in the terminal to remove the old version and re-install the new version of the code.

```
conda remove vaws
conda install -c crankymax vaws
```

2.1.3 Running through GUI

To run the code, the conda environment *vaws_env* should be activated first as *2.1.1 step 3*. And then enter the following command in the terminal.

```
vaws
```

The default scenario will be loaded as shown in Fig. 2.1.

2.2 Instructions for developers

The development of the code is tracked using the git version control system. The source code is at https://github.com/GeoscienceAustralia/vaws.

2.2.1 Installation

1. Get the source code

Source code can be copied by cloning the git repository or downloading the zip file from the git repository.

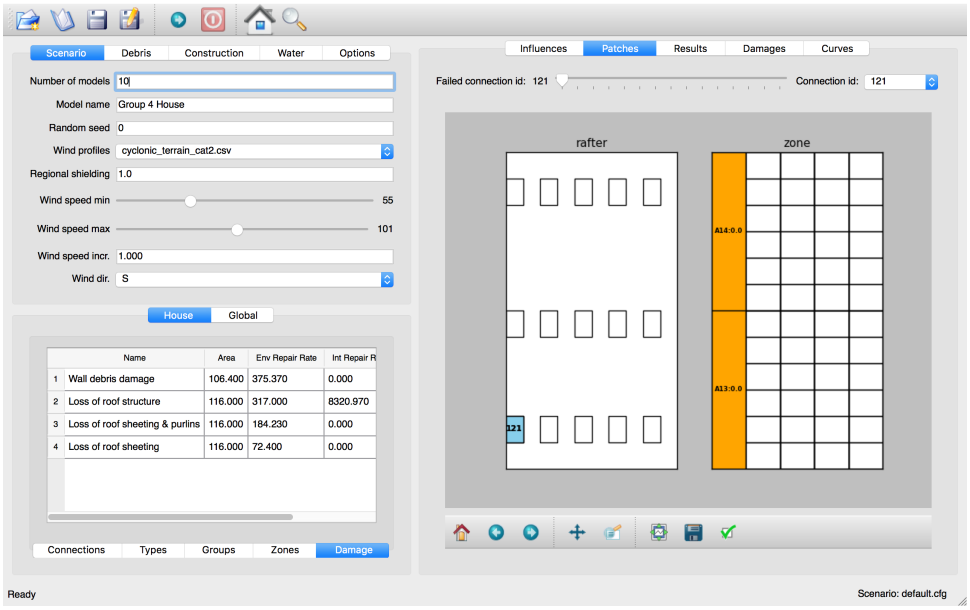

Fig. 2.1: Program main window with default scenario loaded

- If git is installed, run the following command in the terminal

```
$ git clone git@github.com:GeoscienceAustralia/vaws.git
```

- Otherwise download the zip file (https://github.com/GeoscienceAustralia/vaws/archive/master.zip) and then extract it.

This step will create directory called <vaws dir>.

2. Create a conda environment.

Make sure either miniconda or anaconda is installed. Otherwise install either Miniconda or Anaconda with Python 2.7 as *2.1.1 step 1*. Then create the environment called *vaws_env*. by entering the following command in the terminal.

- Windows

```
cd <vaws dir>
conda env create --name vaws_env --file vaws_win.yml
```

- Linux/Mac

```
cd <vaws dir>
conda env create --name vaws_env --file vaws_env.yml
```

This will create the environment called vaws_env. The *vaws_env* can be activated as *2.1.1 step 3*.

3. Create GUI

To create the GUI of the code, enter the following commands in the terminal.

- Windows

```
cd <vaws dir>\vaws\gui
build.cmd
```

- Linux/Mac

```
cd <vaws dir>/vaws/gui
./build.sh
```

4. Run the code

The code can be run in either GUI or CLI mode.

- GUI

```
cd <vaws dir>
python -m vaws.gui.main  # for default scenario
python -m vaws.gui.main -c <config_file>  # for a specific scenario
```

- CLI

```
cd <vaws dir>
python -m vaws.gui.main -c ./vaws/scenarios/default/default.cfg  # for␣
↪default scenario
python -m vaws.gui.main -c <config_file>  # for a specific scenario
```

2.2.2 Building the conda package

Steps for the conda package is described below. Please refer to (https://conda.io/docs/user-guide/tutorials/build-pkgs.html) for details.

1. Install conda-build and anaconda-client

To build the package, you need to install *conda-build* and *anaaconda-client* in the conda *root* environment not the *vaws_env* environment. And then enter the following in the terminal.

```
conda install conda-build anaconda-client
```

2. Build the package

In the terminal client, enter the following to build the package.

```
cd <vaws dir>/build
conda-build .
```

At the end of the building, you should see something like below:

```
Updating index at /foo/anaconda2/conda-bld/noarch to make package␣
↪installable with dependencies
INFO:conda_build.build:Updating index at /foo/anaconda2/conda-bld/noarch␣
↪to make package installable with dependencies
Nothing to test for: /foo/anaconda2/conda-bld/osx-64/vaws-2.0.3-py27_1.
↪tar.bz2
# Automatic uploading is disabled
# If you want to upload package(s) to anaconda.org later, type:

anaconda upload /foo/anaconda2/conda-bld/osx-64/vaws-2.0.3-py27_1.tar.bz2

# To have conda build upload to anaconda.org automatically, use
# $ conda config --set anaconda_upload yes

anaconda_upload is not set.  Not uploading wheels: []
```

3. Upload to anaconda channel

In the terminal client, enter the following to upload the package to the channel.

```
anaconda login
anaconda upload <package>
```

2.2.3 Testing the code

To test the code, the conda environment *vaws_env* should be activated first as *2.1.1 step 3*. And then enter the following command in the terminal.

```
nosetests -v vaws
```

You should see something similar to below.

```
test_distribute_damage_by_row (vaws.model.tests.test_simulation_batten.
↪TestHouseDamage) ... ok
test_calc (vaws.model.tests.test_stats.MyTestCase) ... ok
test_calc2 (vaws.model.tests.test_stats.MyTestCase) ... ok
test_calc_big_a_b_values (vaws.model.tests.test_stats.MyTestCase) ... ok
test_compute_arithmetic_mean_stdev (vaws.model.tests.test_stats.MyTestCase) ... ok
test_compute_logarithmic_mean_stdev (vaws.model.tests.test_stats.MyTestCase) ... ok
test_gev_calc (vaws.model.tests.test_stats.MyTestCase) ... ok
test_gev_calc2 (vaws.model.tests.test_stats.MyTestCase) ... ok
test_sample_logrnormal (vaws.model.tests.test_stats.MyTestCase) ... ok
test_calc_zone_pressures (vaws.model.tests.test_zone.MyTestCase) ... ok
test_get_grid (vaws.model.tests.test_zone.MyTestCase) ... ok
test_is_wall (vaws.model.tests.test_zone.MyTestCase) ... ok
test_str2num (vaws.model.tests.test_zone.MyTestCase) ... ok

----------------------------------------------------------------------
Ran 93 tests in 56.053s

OK
```

CHAPTER

THREE

INPUT DATA

The input data for a scenario consists of a configuration file and a large number of files located in three different directories. This chapter provides details of input data using the template of default scenario, which can be downloaded from https://github.com/GeoscienceAustralia/vaws/blob/master/scenarios/default. The folder structure of the default scenario is shown Listing 3.1, which consists of a configuration file (default.cfg) and input directory with three sub-directories (debris, gust_envelope_profiles, and house).

Listing 3.1: Folder structure

```
default
+-- default.cfg
+-- input
    +-- debris
    |   +-- debris.csv
    |
    +-- gust_envelope_profiles
    |   +-- cyclonic_terrain_cat2.csv
    |   +-- cyclonic_terrain_cat2.5.csv
    |   +-- cyclonic_terrain_cat3.csv
    |   +-- non_cyclonic.csv
    |
    +-- house
        +-- house_data.csv
        +-- conn_groups.csv
        +-- conn_types.csv
        +-- connections.csv
        +-- zones.csv
        +-- zones_cpe_mean.csv
        +-- zones_cpe_str_mean.csv
        +-- zones_cpe_eave_mean.csv
        +-- zones_edge.csv
        +-- coverages.csv
        +-- coverage_types.csv
        +-- coverages_cpe.csv
        +-- influences.csv
        +-- influence_patches.csv
        +-- damage_costing_data.csv
        +-- damage_factorings.csv
        +-- water_ingress_costing_data.csv
        +-- footprint.csv
        +-- front_facing_walls.csv
```

3.1 Configuration file

Each simulation requires a configuration file where basic parameter values for the simulation are provided. The configuration file can be created either by editing the template configuration file using a text editor or through GUI.

The configuration file consists of a number of sections, among which *main* and *options* are mandatory while others are optional. An example configuration file is shown in Listing 3.2.

Listing 3.2: Example configuration file: default.cfg

```
[main]
no_models = 10
house_name = Group 4 House
random_seed = 0
wind_direction = S
wind_speed_min = 55
wind_speed_max = 101
wind_speed_increment = 0.1
wind_profiles = 'cyclonic_terrain_cat2.csv'
regional_shielding_factor = 1.0

[options]
debris = True
diff_shielding = False
water_ingress = True
construction_levels = True
save_heatmaps = True

[debris]
region_name = Capital_city
staggered_sources = False
source_items = 250
building_spacing = 20.0
debris_radius = 200
debris_angle = 45
flight_time_mean = 2.0
flight_time_stddev = 0.8

[construction_levels]
levels = low, medium, high
probabilities = 0.33, 0.34, 0.33
mean_factors = 0.9, 1.0, 1.1
cov_factors = 0.58, 0.58, 0.58

[water_ingress]
thresholds = 0.1, 0.2, 0.5
speed_at_zero_wi = 50.0, 35.0, 0.0, -20.0
speed_at_full_wi = 75.0, 55.0, 40.0, 20.0

[fragility_thresholds]
states = slight, medium, severe, complete
thresholds = 0.02, 0.1, 0.35, 0.9

[heatmap]
vmin = 54.0
vmax = 95.0
vstep = 21.0
```

3.1.1 Main section

Parameters of the main section are listed in Table 3.1. In the GUI window, they are displayed in the Scenario tab as box shown in Fig. 3.1.

Table 3.1: Parameters of the main section

Name	Name in GUI	Description
no_models	Number of models	number of models
house_name	Model name	name of model
random_seed	Random seed	a number used to initialize a pseudorandom number generator
wind_profiles	Wind profiles	file name of wind profile
regional_shielding_factor	Regional shielding	regional shielding factor (default: 1.0)
wind_speed_min	Wind speed min	minimum wind speed (m/s)
wind_speed_max	Wind speed max	maximum wind speed (m/s)
wind_speed_increment	Wind speed incr.	the magnitude of the wind speed increment (m/s)
wind_direction	Wind dir.	wind direction (S, SW, W, NW, N, NE, E, SE, or RANDOM)

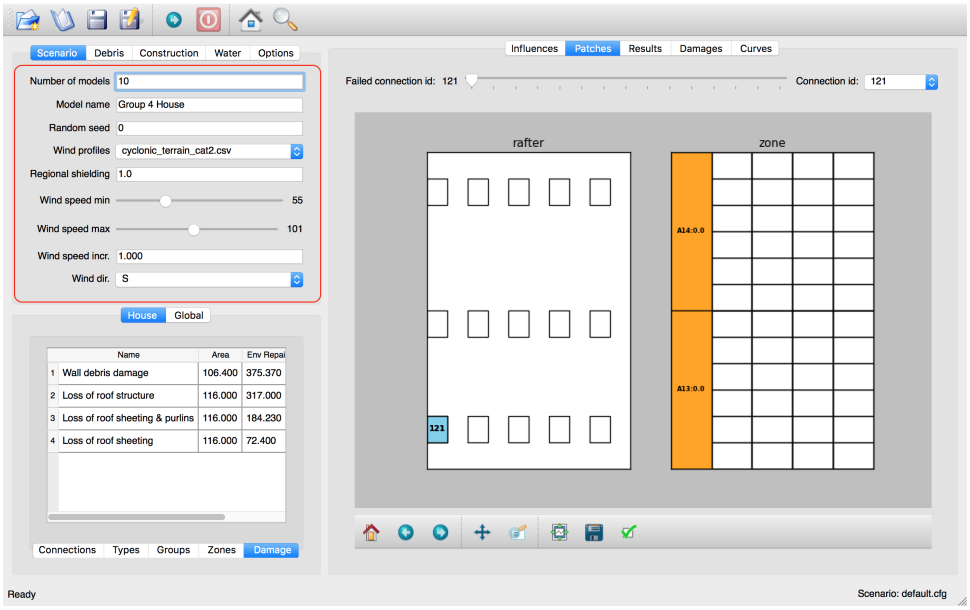

Fig. 3.1: Parameters of main section in the Scenario tab

3.1.2 Options section

Parameters of the Options section are listed in Table 3.2. Note that all the parameter values of the option section should be chosen between *True (or 1)* or *False (or 0)*. In the GUI window, they are displayed in the Debris, Water, Construction, and Options tab as listed in the Table 3.2.

Table 3.2: Parameters of options section

Name	Name in GUI	Description
debris	'Enabled' tick box in the Debris tab	if True then debris damage will be simulated.
diff_shielding	'Differential shielding' tick box in the Options tab	if True then differential shielding effect is applied.
water_ingress	'Enabled' tick box in the Water tab	if True then damage due to water ingress will be simulated.
construction_levels	'Enabled' tick box in the Construction tab	if True then construction level will be sampled.
save_heatmaps	'Save heatmaps' tick box in the Options tab	if True then heatmap plot of each model will be saved.

3.1.3 Debris section

Parameters of the debris section are listed in Table 3.3. Note that debris section is only required if *debris* is set to be *True* in the options. In the GUI window, they are displayed in the Debris tab as box shown in Fig. 3.2.

Table 3.3: Parameters of debris section

Name	Name in GUI	Description
region_name	Region	one of the region names defined in the Listing 3.3. Each region has different debris source characteristics.
building_spacing	Building spacing	distance between debris sources (m)
debris_radius	Radius	radius (in metre) of debris sources from the modelled house
debris_angle	Angle	included angle (in degree) of the sector in which debris sources exist
source_items	Source items	number of debris items per debris sources
flight_time_mean	Flight time mean	mean flight time of debris items
flight_time_stddev	Flight time std	standard deviation of flight time of debris items
staggered_sources	Staggered sources	if True then staggered sources are used. Otherwise, a grid pattern of debris sources are used.

3.1.4 Construction_levels section

Parameters of the construction_levels section are listed in Table 3.4. In the GUI window, they are dispalyed in the Construction tab as box shown in Fig. 3.3.

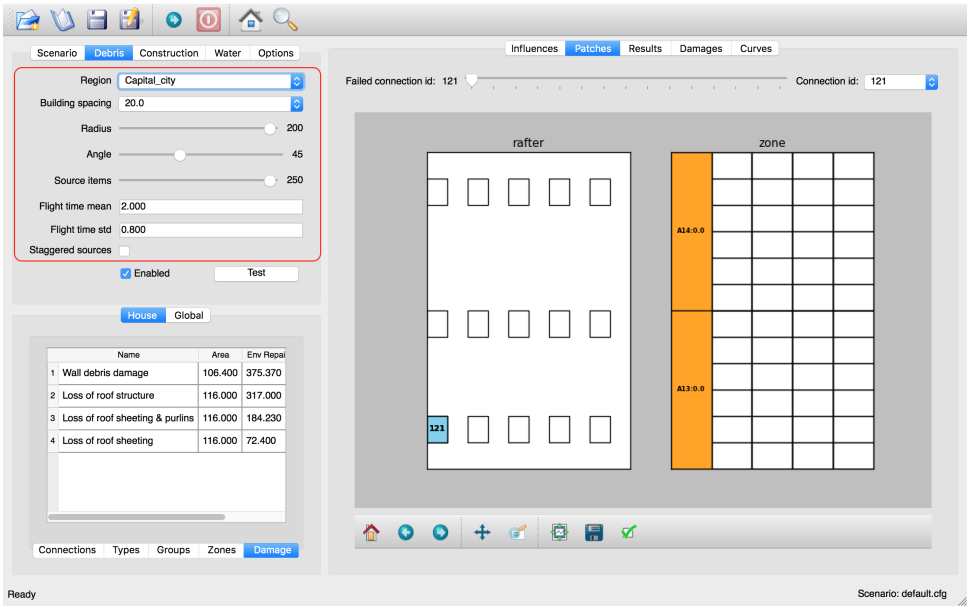

Fig. 3.2: Parameters of debris section in Debris tab

Table 3.4: Parameters of construction_level section

Name	Name in GUI	Description
levels	Levels	comma separated list of construction levels (default: low, medium, high)
probabilities	probabilities	comma separated list of probabilities of a modelled house being of a construction level (default: 0.33, 0.34, 0.33)
mean_factors	Mean factors	comma separated list of mean factors of construction levels (default: 0.9, 1.0, 1.1)
cov_factors	Cov factors	comma separated list of cov factors of construction levels (default: 0.58, 0.58, 0.58)

3.1.5 Water_ingress section

Parameters of the water_ingress section are listed in Table 3.5. In the GUI window, they are displayed in the Water tab as box shown in Fig. 3.5. The thresholds define a lower limit of envelope damage index above which the relevant water ingress vs wind speed curve is applied. The speeds at 0% water ingress and speeds at 100% water ingress define cumulative normal distribution used to relate percentage water ingress to wind speed as shown in Fig. 3.4.

Table 3.5: Parameters of water_ingress section

Name	Name in GUI	Description
thresholds	DI thresholds	comma separated list of thresholds of damage indices (default: 0.0, 0.1, 0.2, 0.5)
speed_at_zero_wi	Speeds at 0% WI	comma separated list of maximum wind speed at no water ingress (default: 40.0, 35.0, 0.0, -20.0)
speed_at_full_wi	Speeds at 100% WI	comma separated list of minimum wind speed at full water ingress (default: 60.0, 55.0, 40.0, 20.0)

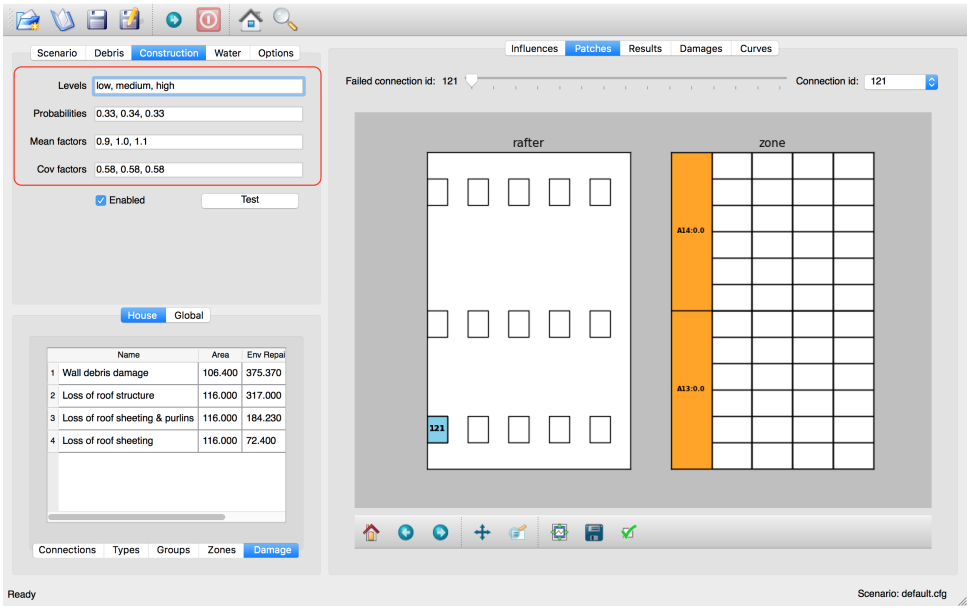

Fig. 3.3: Parameters of construction_levels section in Construction tab

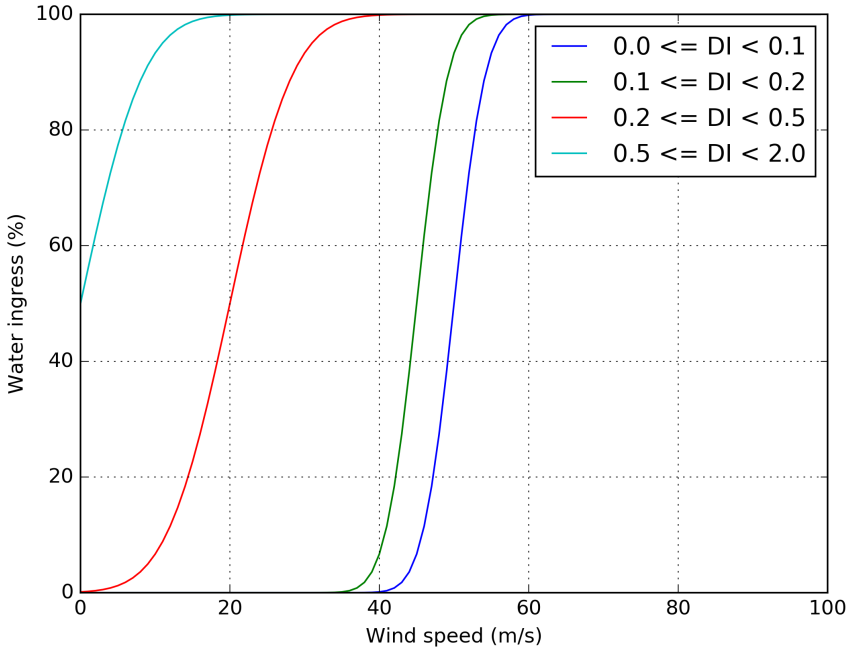

Fig. 3.4: Water ingress vs. wind speed for different ranges of damage index

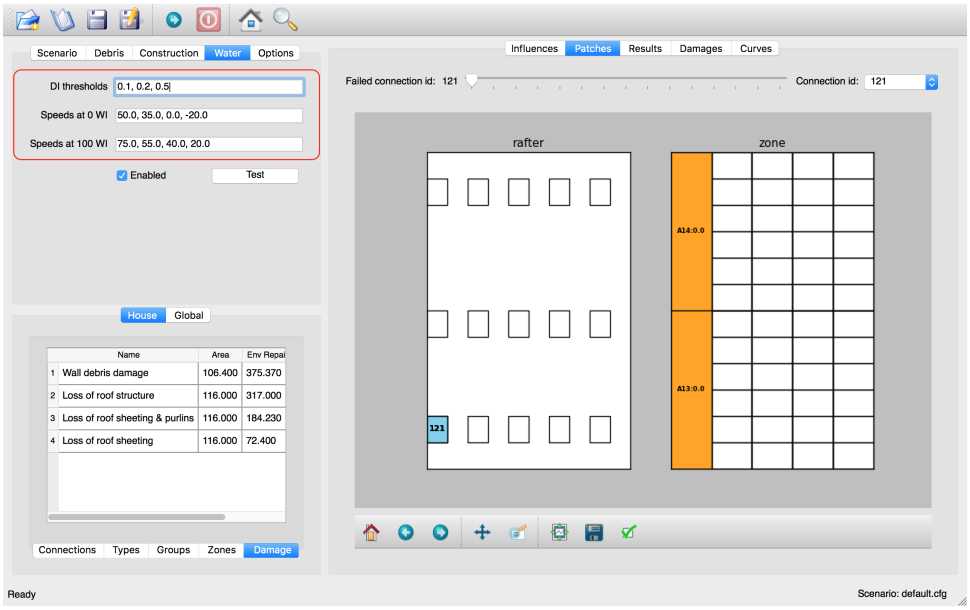

Fig. 3.5: Parameters of water_ingress section in Water tab

3.1.6 Fragility_thresholds

Parameters of the fragility_thresholds section are listed in Table 3.6. In the GUI window, they are displayed in the Options tab as box shown in Fig. 3.6. The probability of exceeding a damage state ds at a wind speed x is calculated as (3.1):

$$P\left(DS \geq ds|\, x\right) = \frac{\sum_{i=1}^{N}\left[DI_{i|x} \geq t_{ds}\right]}{N} \tag{3.1}$$

where N: number of models, $DI_{i|x}$: damage index of i th model at the wind speed x, and t_{ds}: threshold for damage state ds.

Table 3.6: Parameters of fragility_thresholds section

Name	Name in GUI	Description
states	Damage states	comma separated list of damage states (default: slight, medium, severe, complete)
thresholds	Thresholds	comma separated list of damage states thresholds(default: 0.02, 0.1, 0.35, 0.9)

3.1.7 Heatmap

Parameters of the heatmap section are listed in Table 3.7. In the GUI window, they are displayed in the Options tab as box shown in Fig. 3.7

Table 3.7: Parameters of heatmap section

Name	Name in GUI	Description
vmin	Lower limit	lower limit of wind speed for heatmap
vmax	Upper limit	upper limit of wind speed for heatmap
vstep	No. of steps	number of steps

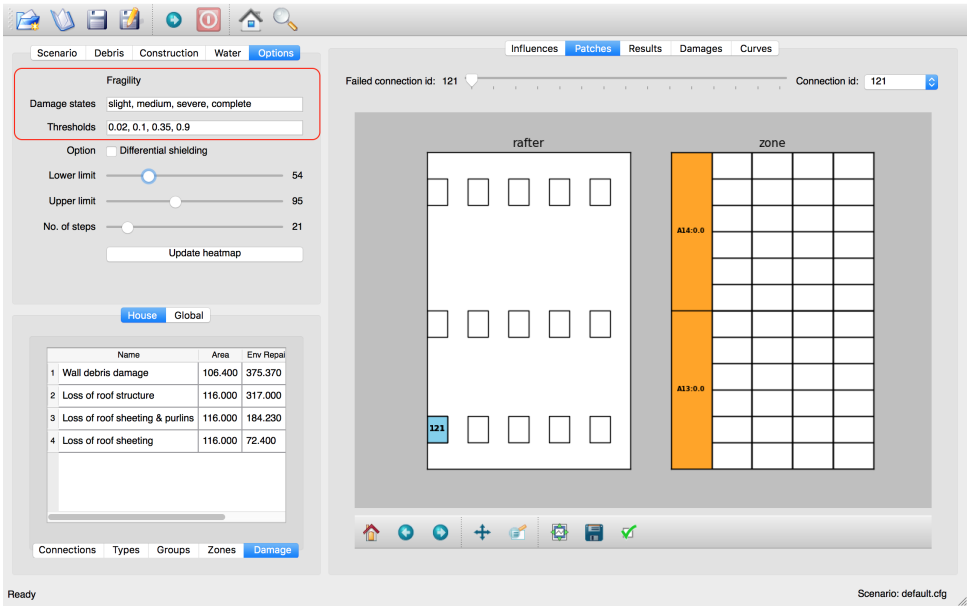

Fig. 3.6: Parameters of fragility_thresholds section in Options tab

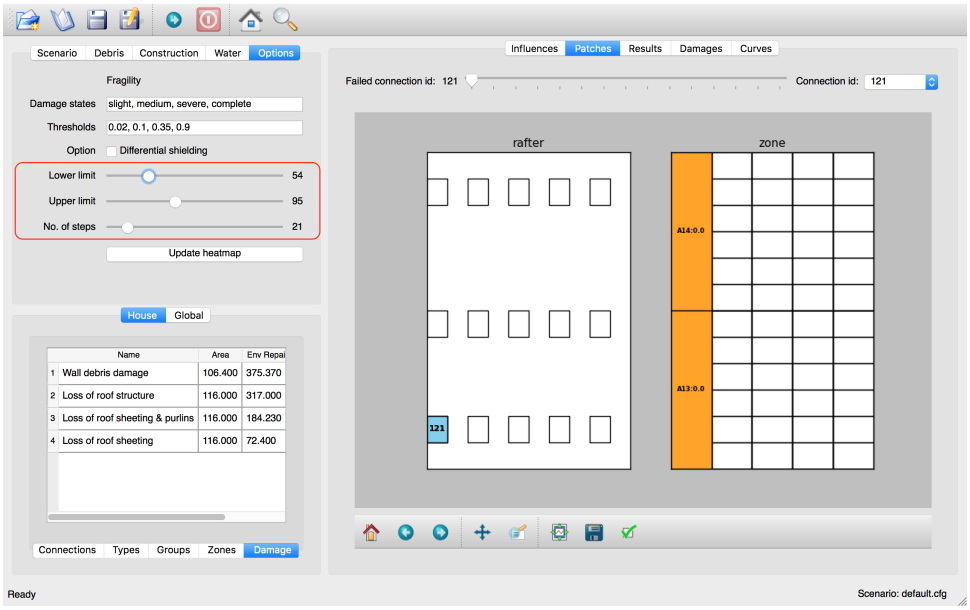

Fig. 3.7: Parameters of heatmap section in Options tab

3.2 Input file under *debris* directory

In the debris directory, *debris.csv* is located where parameter values related to windborne debris are defined. Three types of windborne debris are modelled, as listed in Table 3.8, which include *Compact*, *Rod*, and *Sheet*. Parameter values for each debris type needs to be defined by unique region name, and the defined region name should be referenced in the configuration file.

An example *debris.csv* is shown in Listing 3.3, in which debris parameters are defined for both *Capital_city* and *Tropical_town*. Note that *Capital_city* is referenced in the example configuration file Listing 3.2.

Listing 3.3: Example debris.csv

```
Region name,Capital_city,Tropical_town
Compact_ratio,20,15
Compact_mass_mean,0.1,0.1
Compact_mass_stddev,0.1,0.1
Compact_frontal_area_mean,0.002,0.002
Compact_frontal_area_stddev,0.001,0.001
Compact_cdav,0.65,0.65
Rod_ratio,30,40
Rod_mass_mean,4,4
Rod_mass_stddev,2,2
Rod_frontal_area_mean,0.1,0.1
Rod_frontal_area_stddev,0.03,0.03
Rod_cdav,0.8,0.8
Sheet_ratio,50,45
Sheet_mass_mean,3,10
Sheet_mass_stddev,0.9,5
Sheet_frontal_area_mean,0.1,1
Sheet_frontal_area_stddev,0.03,0.3
Sheet_cdav,0.9,0.9
```

Table 3.8: Debris types

Name	Examples
Compact	Loose nails screws, washers, parts of broken tiles, chimney bricks, air conditioner units
Rod	Parts of timber battens, purlins, rafters
Sheet	Roof cladding (mainly tiles, steel sheet, flashing, solar panels)

The parameter values should be provided for each of the debris types as set out in Table 3.9.

Table 3.9: Parameters for each debris item

Name	Note
ratio	proportion out of debris in percent
mass_mean	mean of mass
mass_stddev	standard deviation of mass
frontal_area_mean	mean of frontal area (m^2)
frontal_area_stddev	standard deviation of frontal area (m^2)
cdav	average drag coefficient

3.3 Input files under *gust_envelope_profiles* directory

The gust envelope profiles are defined under *gust_envelope_profiles* directory. In the configuration file, file name of the gust envelope profile needs to be referenced as shown in Listing 3.2.

Example files are provided in the default sceanrio with respect to Australian wind design categories: cyclonic_terrain_cat2.csv, cyclonic_terrain_cat2.5.csv, cyclonic_terrain_cat3.csv, and non_cyclonic.csv

An example of gust envelope profile is provided in Listing 3.4, and the corresponding plot is shown in Fig. 3.8.

Listing 3.4: Example of gust_envelope_profile

```
# Terrain Category 2
3,0.908,0.896,0.894,0.933,0.884,0.903,0.886,0.902,0.859,0.927
5,0.995,0.980,0.946,0.986,0.962,1.010,0.978,0.970,0.945,0.990
7,0.994,1.031,1.010,0.986,0.982,0.987,0.959,0.984,0.967,0.998
10,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000
12,1.056,1.025,1.032,1.033,0.998,1.043,0.997,1.008,1.005,1.027
15,1.058,1.059,1.028,1.069,1.048,1.076,1.016,1.027,1.021,1.039
17,1.092,1.059,1.079,1.060,1.042,1.053,1.046,1.045,1.047,1.102
20,1.110,1.103,1.037,1.068,1.088,1.107,1.068,1.106,1.098,1.103
25,1.145,1.151,1.069,1.091,1.089,1.196,1.126,1.113,1.099,1.142
30,1.245,1.188,1.177,1.178,1.192,1.199,1.179,1.165,1.127,1.203
```

The first row is header, and heights (in metre) are listed in the first column. Profile values along the heights are listed from the second column with comma separation. One wind profile (one column) will be randomly selected for each run of the simulation.

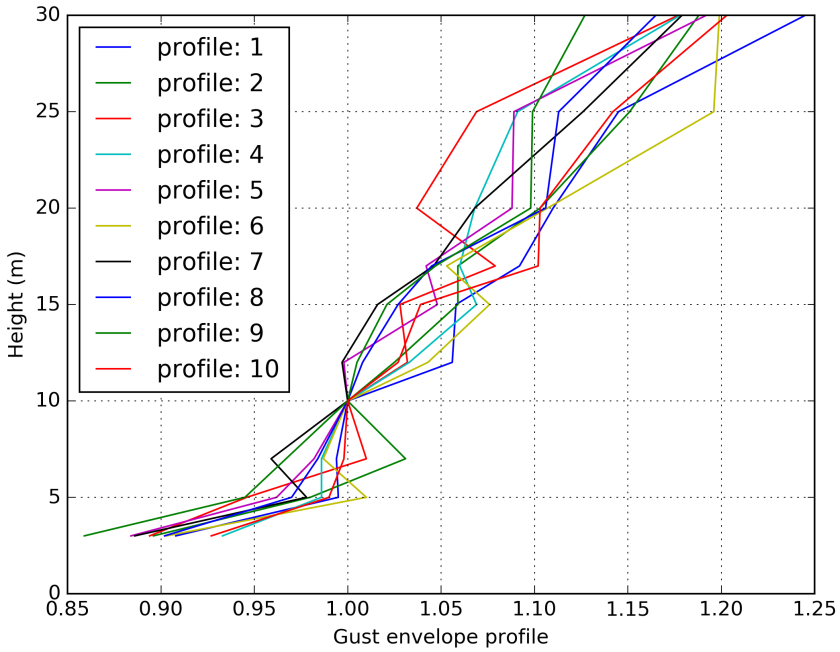

Fig. 3.8: Wind gust envelope profile along height.

3.4 Input files under *house* directory

In the house directory, a large number of files are located which are required to set parameter values of the model. The simulation model is assumed to consist of connections and zones. The connections are grouped into a number of connection types, and the connection types are further grouped into connection groups.

3.4.1 house_data.csv

This file defines parameter values for the model such as replacement cost and dimensions. An example is shown in Listing 3.5, and description of each of the parameter values are provided in Table 3.10.

Listing 3.5: Example house_data.csv

```
replace_cost,3220.93
height,4.5
length,0.9
width,9.0
cpe_cov,0.0
cpe_k,0.1
cpe_str_cov,0.0
cpe_str_k,0.1
```

Table 3.10: Parameters in the house_data.csv

Name	Type	Description
replace_cost	float	replacement cost of the model ($)
height	float	height of the model (in metre)
length	float	length of the model (in metre)
width	float	width of the model (in metre)
cpe_cov	float	cov of Cpe for sheeting and batten
cpe_k	float	shape factor of Cpe for sheeting and batten
cpe_str_cov	float	cov of Cpe for rafters and eaves
cpe_str_k	float	shape factor of Cpe for rafters and eaves

3.4.2 conn_groups.csv

The model is assumed to consist of a number of connection groups. This file defines connection groups and parameter values of the each connection group. An example is shown in Listing 3.6, and description of each of the parameter values are provided in Table 3.11.

Listing 3.6: Example conn_groups.csv

```
group_name,dist_order,dist_dir,damage_scenario,trigger_collapse_at,patch_dist
sheeting,1,col,Loss of roof sheeting,0.0,1
batten,2,row,Loss of roof sheeting & purlins,0.0,1
rafter,3,col,Loss of roof structure,0.0,1
```

Table 3.11: Parameters in the conn_groups.csv

Name	Type	Description
group_name	string	name of connections group
dist_order	integer	order of checking damage
dist_dir	integer	direction of damage distribution; either 'col', 'row', or ''
damage_scenario	string	damage scenario name defined in damage_costing_data.csv
trigger_collapse_at	float	proportion of damaged connections of the group at which a model is deemed to be collapsed. 0 if ignored
patch_dist	integer	1 if influence patch is applied when connection is damaged otherwise 0

3.4.3 conn_types.csv

A connection group may consists of a number of connection types which have different parameter values for strength, dead load, and costing area. This file defines connection types and parameter values of the each connection type. An example is shown in Listing 3.7, and description of each of the parameter values are provided in Table 3.12.

Listing 3.7: Example conn_types.csv

```
type_name,strength_mean,strength_std,dead_load_mean,dead_load_std,group_name,
↪costing_area
sheetinggable,1.54,0.16334,0.02025,0.0246,sheeting,0.405
sheetingeave,4.62,0.28292,0.02025,0.0246,sheeting,0.405
sheetingcorner,2.31,0.2,0.01013,0.0246,sheeting,0.225
sheeting,2.695,0.21608,0.0405,0.0246,sheeting,0.81
batten,3.6,1.26,0.089,0.0708,batten,0.81
battenend,3.6,1.26,0.089,0.0708,batten,0.405
batteneave,3.6,1.26,0.089,0.0708,batten,0.405
battencorner,3.6,1.26,0.089,0.0708,batten,0.225
endraftertopplate,19.5,5.85,0.84,0.063,rafter,1.238
endrafterridge,16.5,4.95,1.8,0.135,rafter,1.665
collarraftertopplate,19.5,5.85,1.68,0.126,rafter,1.845
collarrafterridge,16.5,4.95,1.13,0.08475,rafter,1.26
collarraftercollar,2.4,0.48,3.95,0.29625,rafter,1.665
plainraftertopplate,19.5,5.85,1.68,0.126,rafter,2.475
plainrafterridge,16.5,4.95,3.6,0.27,rafter,3.33
weakbatten,3.6,1.26,0.089,0.0708,batten,0.81
```

Table 3.12: Parameters in the conn_types.csv

Name	Type	Description
type_name	string	name of connection type
strength_mean	float	mean strength (kN)
strength_std	float	standard deviation of strength
dead_load_mean	float	mean dead load (kN)
dead_load_std	float	standard deviation of dead load
group_name	string	name of connections group
costing_area	float	costing area (m^2)

3.4.4 connections.csv

This file defines connections and parameter values of the each connection. An example is shown in Listing 3.8, and description of each of the parameter values are provided in Table 3.13.

Listing 3.8: Example connections.csv

```
conn_name,type_name,zone_loc,section,coords
1,sheetingcorner,A1,1,0,0,0.2,0,0.2,0.5,0,0.5
2,sheetinggable,A2,1,0,0.5,0.2,0.5,0.2,1,0,1
3,sheetinggable,A3,1,0,1,0.2,1,0.2,1.5,0,1.5
4,sheetinggable,A4,1,0,1.5,0.2,1.5,0.2,2,0,2
5,sheetinggable,A5,1,0,2,0.2,2,0.2,2.5,0,2.5
```

Table 3.13: Parameters in the connections.csv

Name	Type	Description
conn_name	string	name of connection
type_name	string	name of connection type
zone_loc	integer	zone name corresponding to connection location
section	integer	index of section in which damage distribution occurs
coords	float	comma separated values of x, y coordinates for plotting purpose. Provide 4 sets of x, y coordinates for a rectangular shape.

3.4.5 zones.csv

This file defines zones and parameter values of the each zone. An example is shown in Listing 3.9, and description of each of the parameter values are provided in Table 3.14.

Listing 3.9: Example zones.csv

```
name,area,cpi_alpha,wall_dir,coords,
A1,0.2025,0,0,0,0,0.2,0,0.2,0.5,0,0.5
A2,0.405,0.5,0,0,0.5,0.2,0.5,0.2,1,0,1
A3,0.405,1,0,0,1,0.2,1,0.2,1.5,0,1.5
A4,0.405,1,0,0,1.5,0.2,1.5,0.2,2,0,2
A5,0.405,1,0,0,2,0.2,2,0.2,2.5,0,2.5
```

Table 3.14: Parameters in the zones.csv

Name	Type	Description
name	string	name of zone
area	float	area of zone (m^2)
cpi_alpha	float	proportion of the zone's area to which internal pressure is applied
coords	float	comma separated list of x, y coordinates for plotting purpose. Provide 4 sets of x, y coordinates for a rectangular shape.

3.4.6 zones_cpe_mean.csv

This file defines mean cladding Cpe of each zone with regard to the eight wind directions. An example is shown in Listing 3.10, and description of each of the parameter values are provided in Table 3.15.

Listing 3.10: Example zones_cpe_mean.csv

```
name,S,SW,W,NW,N,NE,E,SE
A1,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2
A2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2
A3,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2
A4,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2
A5,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2
A6,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2,-1.2
A7,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5
A8,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5
A9,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5
A10,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5
A11,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5
A12,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5
A13,0,0,0,0,0,0,0,0
A14,0,0,0,0,0,0,0,0
```

Table 3.15: Parameters in the zones_cpe_mean.csv

Name	Type	Description
name	string	name of zones
S	float	mean cladding Cpe value in South direction
SW	float	mean cladding Cpe value in South West direction
W	integer	mean cladding Cpe value in West direction
NW	float	mean cladding Cpe value in North East direction
N	float	mean cladding Cpe value in North direction
NE	float	mean cladding Cpe value in North East direction
E	integer	mean cladding Cpe value in East direction
SE	float	mean cladding Cpe value in South East direction

3.4.7 zones_cpe_str_mean.csv

Like zones_cpe_mean.csv, mean Cpe values for zones associated with structural component (e.g., rafter) need to be provided in zones_cpe_str_mean.csv. An example is shown in Listing 3.11.

Listing 3.11: Example zones_cpe_str_mean.csv

```
name,S,SW,W,NW,N,NE,E,SE
A1,0,0,0,0,0,0,0,0
A2,0,0,0,0,0,0,0,0
A3,0,0,0,0,0,0,0,0
A4,0,0,0,0,0,0,0,0
A5,0,0,0,0,0,0,0,0
A6,0,0,0,0,0,0,0,0
A7,0,0,0,0,0,0,0,0
A8,0,0,0,0,0,0,0,0
A9,0,0,0,0,0,0,0,0
A10,0,0,0,0,0,0,0,0
A11,0,0,0,0,0,0,0,0
A12,0,0,0,0,0,0,0,0
A13,-1,-1,-1,-1,-1,-1,-1,-1
A14,-0.4,-0.4,-0.4,-0.4,-0.4,-0.4,-0.4,-0.4
```

3.4.8 zones_cpe_eave_mean.csv

Like zones_cpe_mean.csv, mean Cpe values for zones at eave need to be provided in zones_cpe_eave_mean.csv. An example is shown in Listing 3.12.

Listing 3.12: Example zones_cpe_eave_mean.csv

```
name,S,SW,W,NW,N,NE,E,SE
A1,0.7,0.7,0.7,0.7,0.7,0.7,0.7,0.7
A2,0.35,0.35,0.35,0.35,0.35,0.35,0.35,0.35
A3,0,0,0,0,0,0,0,0
A4,0,0,0,0,0,0,0,0
A5,0,0,0,0,0,0,0,0
A6,0,0,0,0,0,0,0,0
A7,0,0,0,0,0,0,0,0
A8,0,0,0,0,0,0,0,0
A9,0,0,0,0,0,0,0,0
A10,0,0,0,0,0,0,0,0
A11,-0.1,-0.1,-0.1,-0.1,-0.1,-0.1,-0.1,-0.1
A12,-0.2,-0.2,-0.2,-0.2,-0.2,-0.2,-0.2,-0.2
A13,0.07,0.07,0.07,0.07,0.07,0.07,0.07,0.07
A14,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02,-0.02
```

3.4.9 zones_edge.csv

In zones_edge.csv, for each of the eight direction, 1 is provided for zone within the region of a roof edge, otherwise 0. Zones in the edge region are considered to be subjected to differential shielding if enabled by user. An example is shown in Listing 3.13.

Listing 3.13: Example zones_edge.csv

```
name,S,SW,W,NW,N,NE,E,SE
A1,1,1,1,0,0,0,0,0
A2,1,1,1,0,0,0,0,0
```

```
A3,1,1,1,0,0,0,0,0
A4,0,1,0,0,0,0,0,0
A5,0,1,0,0,0,0,0,0
A6,0,1,0,0,0,0,0,0
A7,0,0,0,1,0,0,0,0
A8,0,0,0,1,0,0,0,0
A9,0,0,0,1,0,0,0,0
A10,0,0,1,1,1,0,0,0
A11,0,0,1,1,1,0,0,0
A12,0,0,1,1,1,0,0,0
A13,1,1,1,0,0,0,0,0
A14,0,0,1,1,1,0,0,0
```

3.4.10 coverages.csv

This file defines coverages making up the wall part of the envelope of the model. An example is shown in Listing 3.14, and description of each of the parameter values are provided in Table 3.16. The wall name is defined in Listing 3.23.

Listing 3.14: Example coverages.csv

```
name,description,wall_name,area,coverage_type
1,window,1,3.6,Glass_annealed_6mm
2,door,1,1.8,Timber_door
3,window,1,1.89,Glass_annealed_6mm
4,window,1,1.89,Glass_annealed_6mm
```

Table 3.16: Parameters in tge coverages.csv

Name	Type	Description
name	integer	coverage index
description	string	description
wall_name	integer	wall name
area	float	area (m^2)
coverage_type	string	name of coverage type

3.4.11 coverage_types.csv

This file defines types of coverages referenced in the Listing 3.14. An example is shown in Listing 3.15, and description of each of the parameter values are provided in Table 3.17.

Listing 3.15: Example coverage_types.csv

```
name,failure_momentum_mean,failure_momentum_std,failure_strength_in_mean,failure_
↪strength_in_std,failure_strength_out_mean,failure_strength_out_std
Glass_annealed_6mm,0.05,0.0,100,0.0,-100,0.0
Timber_door,142.2,28.44,100,0.0,-100,0.0
```

Table 3.17: Parameters in the coverage_types.csv

Name	Type	Description
name	string	name of coverage type
failure_momentum_mean	float	mean failure momentum (kg · m/s)
failure_momentum_std	float	standard deviation of failure momentum
failure_strength_in_mean	float	mean failure strength inward direction (kN)
failure_strength_in_std	float	standard deviation of failure strength inward direction
failure_strength_out_mean	float	mean failure strength outward direction (kN)
failure_strength_out_std	float	standard deviation of failure strength outward direction

3.4.12 coverages_cpe.csv

Like zones_cpe_mean.csv, mean Cpe values for coverages are provided in coverages_cpe.csv. An example is shown in Listing 3.16.

Listing 3.16: Example coverages_cpe.csv

```
ID,S,SW,W,NW,N,NE,E,SE
1,2.4,2.4,2.4,2.4,2.4,2.4,2.4,2.4
2,1.69,1.69,1.69,1.69,1.69,1.69,1.69,1.69
3,-1.14,-1.14,-1.14,-1.14,-1.14,-1.14,-1.14,-1.14
4,-1.45,-1.45,-1.45,-1.45,-1.45,-1.45,-1.45,-1.45
5,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9
6,-0.55,-0.55,-0.55,-0.55,-0.55,-0.55,-0.55,-0.55
```

3.4.13 influences.csv

This file defines influence coefficients relating a connection with either another connection(s) or zone(s). The wind load acting on a connection can be computed as the sum of the product of influence coefficient and either wind load on zone or load on another connection. An example is shown in Listing 3.17, and description of each of the parameter values are provided in Table 3.18. In this example, connection 1 is related to the zone A1 with coefficient 1.0, and connection 61 is related to the connection 1 with coefficient 1.0. Similarly, connection 121 is related to the zone A13 with coefficient 0.81 and the zone A14 with coefficient 0.19.

Listing 3.17: Example influences.csv

```
Connection,Zone,Coefficent
1,A1,1.0
2,A2,1.0
61,1,1
62,2,1
63,3,1
121,A13,0.81,A14,0.19
```

Table 3.18: Parameters in the influences.csv

Name	Description
Connection	name of connection
Zone	name of either zone or connection associated with the Connection
Coefficient	coefficient value

3.4.14 influence_patches.csv

This file defines influence coefficients of connections when associated connection is failed. An example is shown in Listing 3.18, and description of each of the parameter values are provided in Table 3.19. In the example, when connection 121 is failed, influence coefficients of connection 121, 122, 123 are re-defined.

Listing 3.18: Example influence_patches.csv

```
Damaged connection,Connection,Zone,Coefficient
121,121,A13,0,A14,0
121,122,A13,1,A14,0
121,123,A13,1,A14,1
122,121,A13,1,A14,0
122,122,A13,0,A14,0
122,123,A13,0,A14,1
```

Table 3.19: Parameters in the influence_patches.csv

Name	Description
Damaged Connection	name of damaged connection
Connection	name of connection for which the influence coefficients are to be updated
Zone	name of either zone or connection associated with the connection to be updated
Coefficient	new influence coefficient value

3.4.15 damage_costing_data.csv

This file defines damage scenarios referenced in Listing 3.6. An example is shown in Listing 3.19, and description of each of the parameter values are provided in Table 3.20. The damage cost for each damage scenario C is calculated as

$$C = x \times (A \times C_{\text{env}} \times R_{\text{env}} + C_{\text{int}} \times R_{\text{int}}) \tag{3.2}$$

where x: proportion of damaged area ($0 \leq x \leq 1$), A: surface area, C_{env}: costing function for envelope, R_{env}: repair rate for envelope, C_{int}: costing function for internal, and R_{int}: repair rate for internal. Two types of costing functions are defined as:

$$\begin{aligned} f_1 &= c_1 \times x^2 + c_2 \times x + c_3 \\ f_2 &= c_1 \times x^{c_2} \end{aligned} \tag{3.3}$$

Listing 3.19: Example damage_costing_data.csv

```
name,surface_area,envelope_repair_rate,envelope_factor_formula_type,envelope_
↪coeff1,envelope_coeff2,envelope_coeff3,internal_repair_rate,internal_factor_
↪formula_type,internal_coeff1,internal_coeff2,internal_coeff3,water_ingress_order
Loss of roof sheeting,116,72.4,1,0.3105,-0.8943,1.6015,0,1,0,0,0,6
Loss of roof sheeting & purlins,116,184.23,1,0.3105,-0.8943,1.6015,0,1,0,0,0,7
Loss of roof structure,116,317,1,0.3105,-0.8943,1.6015,8320.97,1,-0.4902,1.4896,0.
↪0036,3
```

Table 3.20: Parameters in the damage_costing_data.csv

Name	Description
name	name of damage scenario
surface_area	surface area (m^2)
envelope_repair_rate	repair rate for envelope ($/$m^2$)
envelope_factor_formula_type	type index of costing function for envelope
envelope_coeff1	c_1 in costing function for envelope
envelope_coeff2	c_2 in costing function for envelope
envelope_coeff3	c_3 in costing function for envelope
internal_repair_rate	repair rate for internal ($)
internal_factor_formula_type	type index of costing function for internal
internal_coeff1	c_1 in costing function for internal
internal_coeff2	c_2 in costing function for internal
internal_coeff3	c_3 in costing function for internal
water_ingress_order	order in applying cost induced by water ingress

3.4.16 damage_factorings.csv

This file defines a hierarchy of costings, where each row has a parent and child connection type group. When costing the parent group, all child costings will be factored out of the parent. This mechanism avoids double counting of repair costs. An example is shown in Listing 3.20.

Listing 3.20: Example damage_factorings.csv

```
ParentGroup,FactorByGroup
batten,rafter
sheeting,rafter
sheeting,batten
```

3.4.17 water_ingress_costing_data.csv

This file contains costing information of damage induced by water ingress for various scenarios of structural damage. Each row contains coefficients that are used by costing functions. An example is shown in Listing 3.21, and description of each of the parameter values are provided in Table 3.21. The water ingress cost WC is calculated as

$$WC = B \times C(x) \tag{3.4}$$

where x: envelope damage index prior to water ingress ($0 \leq x \leq 1$), B: base cost, and C: costing function. Like the damage costing functions, two types of costing functions are defined as (3.3).

Listing 3.21: Example water_ingress_costing_data.csv

```
name,water_ingress,base_cost,formula_type,coeff1,coeff2,coeff3
Loss of roof sheeting,0,0,1,0,0,1
Loss of roof sheeting,5,2989.97,1,0,0,1
Loss of roof sheeting,18,10763.89,1,0,0,1
Loss of roof sheeting,37,22125.78,1,0,0,1
Loss of roof sheeting,67,40065.59,1,0,0,1
Loss of roof sheeting,100,59799.39,1,0,0,1
```

Table 3.21: Parameters in the water_ingress_costing_data.csv

Name	Description
name	name of damage scenario
water_ingress	water ingress in percent
base_cost	base cost B
formula_type	type index of costing function
coeff1	c_1 in costing function
coeff2	c_2 in costing function
coeff3	c_3 in costing function

3.4.18 footprint.csv

This file contains information about footprint of the model. Each row contains x and y coordinates of the vertices of the footprint. An example is shown in Listing 3.22.

Listing 3.22: Example footprint.csv

```
footprint_coord
-6.5, 4.0
6.5, 4.0
6.5, -4.0
-6.5, -4.0
```

3.4.19 front_facing_walls.csv

This file contains wall information with respect to the eight wind direction. Each row contains wall name(s) for a wind direction. An example is shown in Listing 3.23.

Listing 3.23: Example front_facing_walls.csv

```
wind_dir,wall_name
S,1
SW,1,3
W,3
NW,3,5
N,5
NE,5,7
E,7
SE,1,7
```

CHAPTER

FOUR

USE OF THE GUI

This chapter provides an overview of the GUI and instructions on how to run simulations using the GUI.

4.1 Structure

The main window is logically separated into distinct areas of functionality as shown as Fig. 4.1.

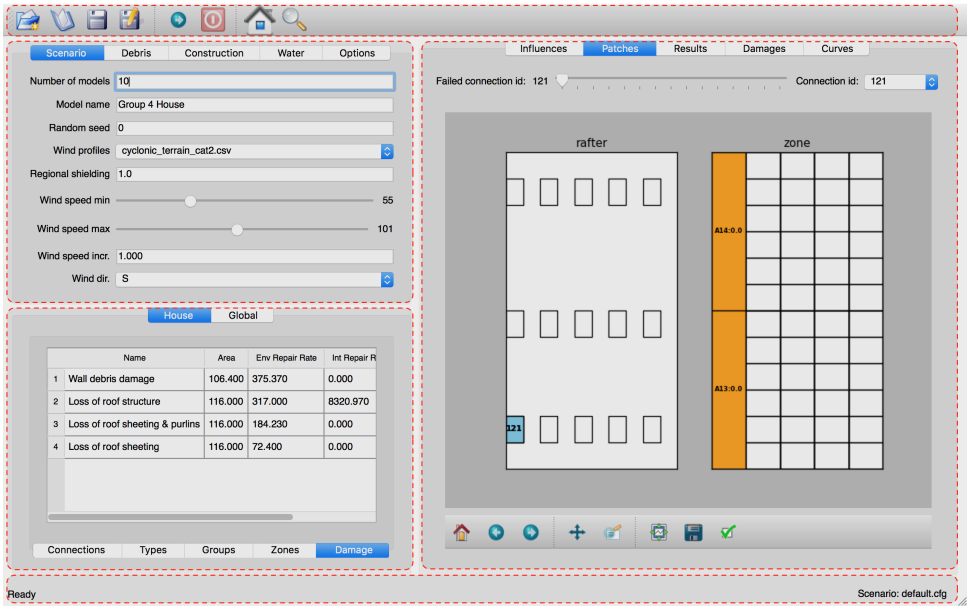

Fig. 4.1: Program main window consisting of five areas by functionality as shown as dotted box

4.1.1 Top

At the top of the main window, tool bar with buttons is located. The file menu and action corresponding to each of the buttons is set out in the Table 4.1.

Table 4.1: Buttons in the toolbar

Button	Menu	Action
	File -> New	Create a new scenario with default setting
	File -> Open Scenario	Open an existing configuration file
	File -> Save	Save current scenario
	File -> Save As	Save current scenario to a new configuration file
	Simulator -> Run	Run the scenario
	Simulator -> Stop	Stop the simulation when in progress
	Model -> House Info	Show the current house information including wall coverages

4.1.2 Top left

The top left panel contains simulation settings across five tabs: Scenario, Debris, Construction, Water, and Options, where parameter values for a simulation can be set. The details of each of the tab can be found in *3.1*.

There are three test button across the tabs.

The Test button in the Debris tab demonstrates debris generation function at a selected wind speed. Once the wind speed is determined, then a window showing debris traces from sources are displayed as shown as Fig. 4.2. The vulnerability curve used in the test function is selected from the curves shown in Fig. 4.3, whereas other parameter values can be changed through the GUI.

Table 4.2: Parameter values for vulnerability curves (4.2) used in the debris test

name	α	β
Capital_city	0.1585	3.8909
Tropical_town	0.1030	4.1825

The Test button in the Construction tab shows distribution of connection strength of the selected connection type. Example of sampled strength of batten type is shown in Fig. 4.4.

The Test button in the Water tab shows relationship between percentage of water ingress and wind speed for a range of damage index as shown in Fig. 4.5.

4.1.3 Bottom left

The bottom left panel contains data browser of house and global data. This panel contains two tabs at the top: House and Global. The House tab has five tabs at the bottom: Connections, Types, Groups, Zones, and Damage, as shown in Fig. 4.6. Table 4.3 sets out corresponding input file and section for each of the tabs.

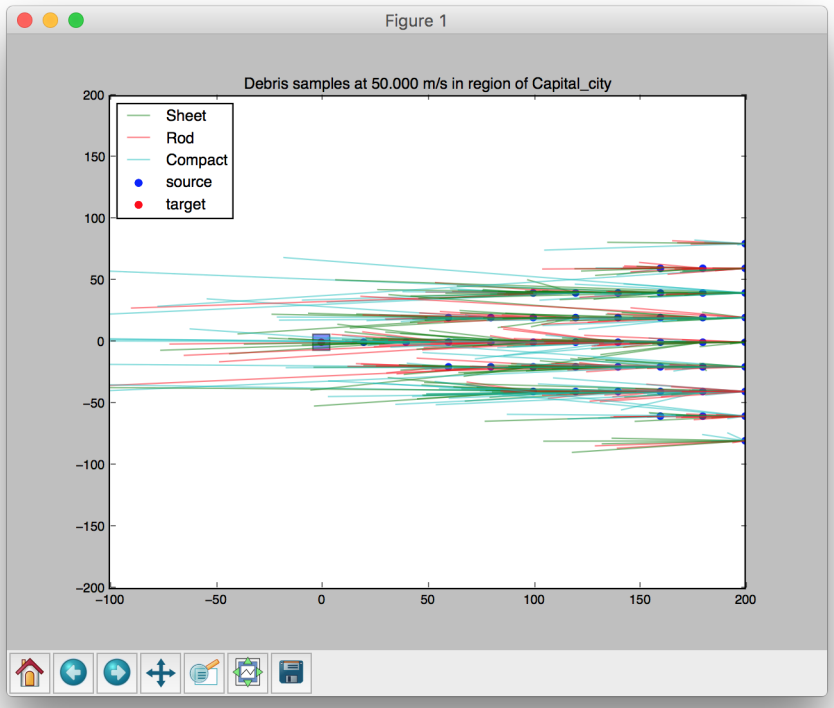

Fig. 4.2: Test of debris generation function: debris generated at 50 m/s in region of Capcital_city

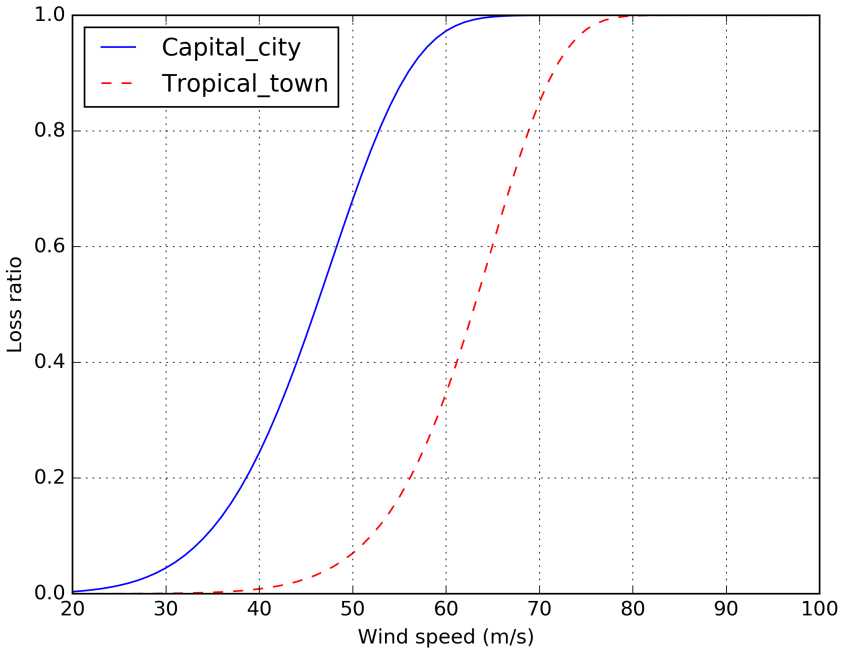

Fig. 4.3: Vulnerability curves implemented in the debris test function using the parameter values listed in Table 4.2.

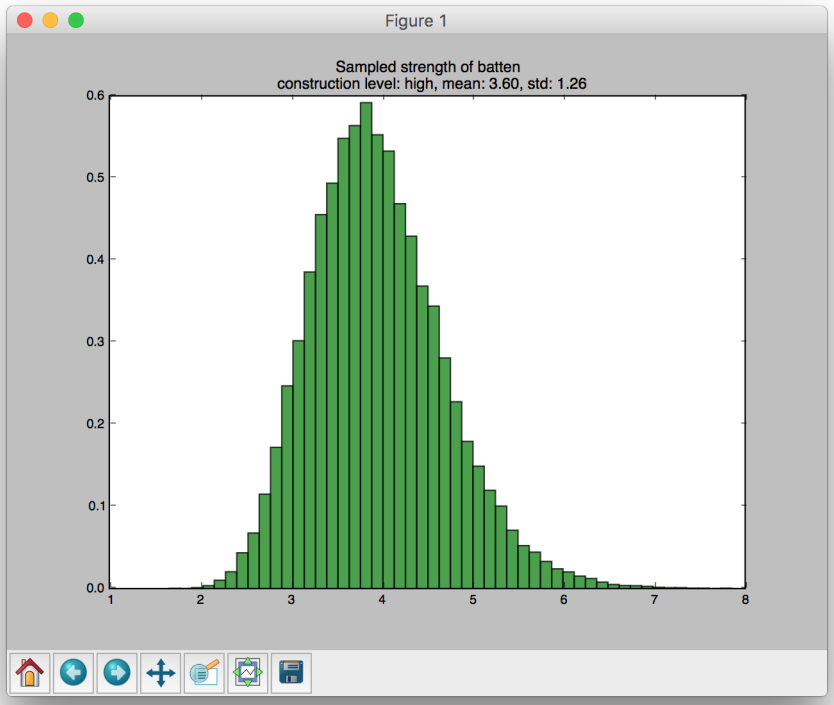

Fig. 4.4: Distribution of sampled strength of the selected connection type

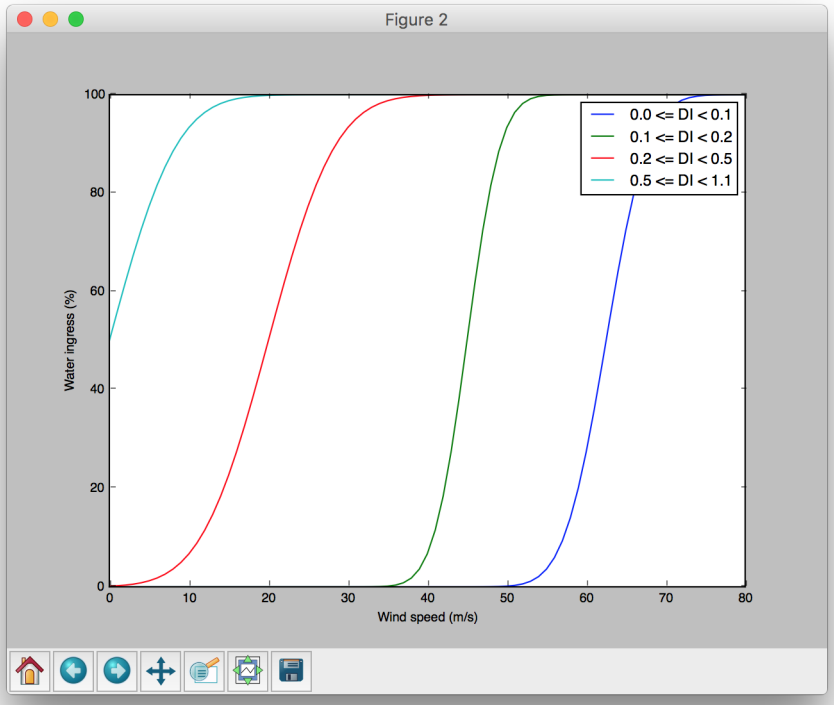

Fig. 4.5: Relationship between percentage of water ingress and wind speed

Table 4.3: House data

Tab name	Input file	Section
Connections	connections.csv	*3.4.4*
Types	conn_types.csv	*3.4.3*
Groups	conn_groups.csv	*3.4.2*
Zones	zones.csv	*3.4.5*
Damage	damage_costing_data.csv	*3.4.15*

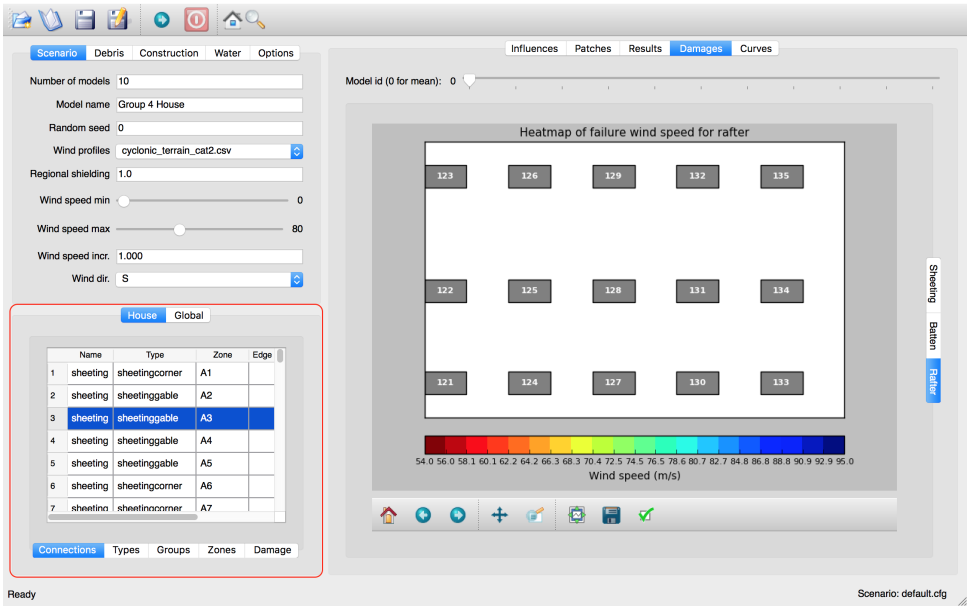

Fig. 4.6: House data tab showing connections information

The Global tab has two tabs at the bottom: Boundary Profile and Debris, as shown in Fig. 4.7. In the Boundary Profile tab, gust envelope profiles of selected wind profiles is displayed. Details about the gust envelop profiles can be found in *3.3*. In the Debris tab, parameter values for debris model listed in the debris.csv (*3.1.3*) is displayed. Note that the contents of both tabs are to be changed dynamically upon different selection of wind profile file (Wind Profiles) and debris region (Region).

4.1.4 Bottom right

The bottom right panel shows input data of influence coefficients and simulation results. This panel consists of five tabs: Influences, Patches, Results, Damages, and Curves, among which Results, Damages, and Curves are empty until a simulation is completed.

Influences tab

Once connection id is set by the slider at the top, then the selected connection (coloured in skyblue) and its associated either zone or connection (coloured in orange) with influence coefficient are shown as Fig. 4.8.

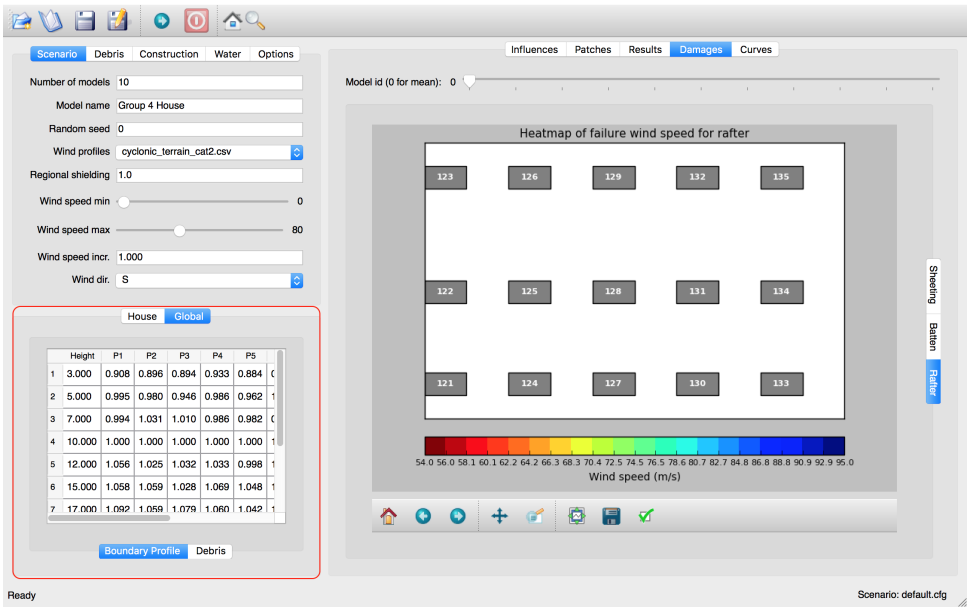

Fig. 4.7: Global data tab showing boundary profiles information

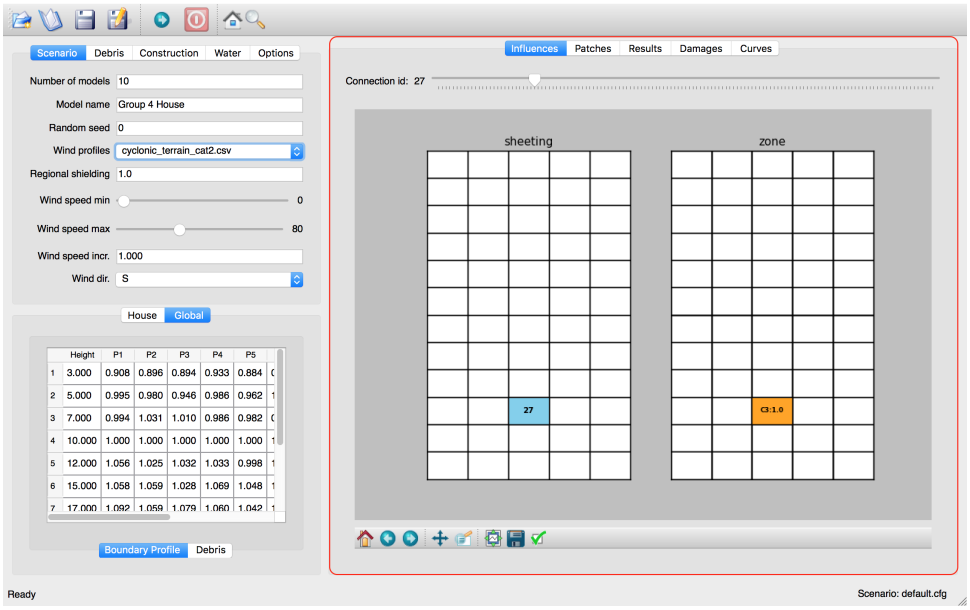

Fig. 4.8: Display of influence coefficient of connection id 27, which is 1.0 with Zone C3.

Patches tab

The Patches tab shows the influence coefficient of connection when associated connection is failed. Once failed connection (coloured in gray) and connection id (coloured in skyblue) are set, then associated either zone or connection (coloured in orange) with influence coefficient is shown as Fig. 4.9.

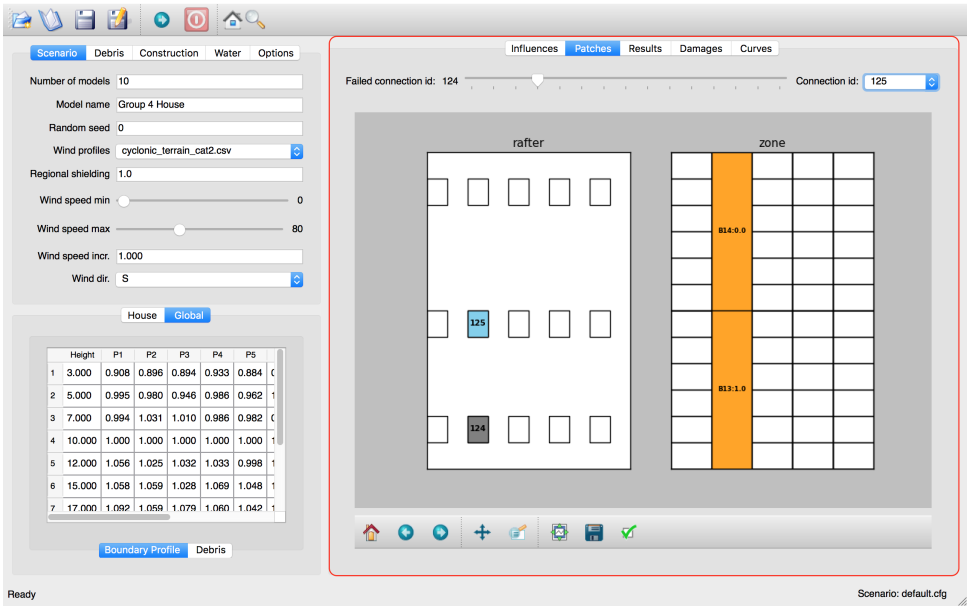

Fig. 4.9: Display of influence coefficient of connection 125 when connection 124 is failed.

Results tab

The Results tab shows the results of simulation in four sub-windows: Zones, Connections, Type Strengths, and Type Damage.

The Zones window shows sampled Cpe values for each of the zones for each realisation of the simulation models as shown as Fig. 4.10. The first string at the *House* column refers to model index, and the string before and after slash refer to wind direction and construction quality level, respectively.

Likewise, the Connections window shows the results of each connections such as sampled strength and dead load as shown as Fig. 4.11.

The Type Strengths window show distribution of connection strength by connection type as shown as Fig. 4.12.

The Type Damage window shows distribution of speeds at which connection fails by connection type as shown as Fig. 4.13.

Damages tab

The Damages tab shows heatmap by connection type group such as Sheeting, Batten, and Rafter as shown in Fig. 4.14. The heatmap averaged across models is shown by default, and heatmap for individual model can be displayed by moving the slide at the top.

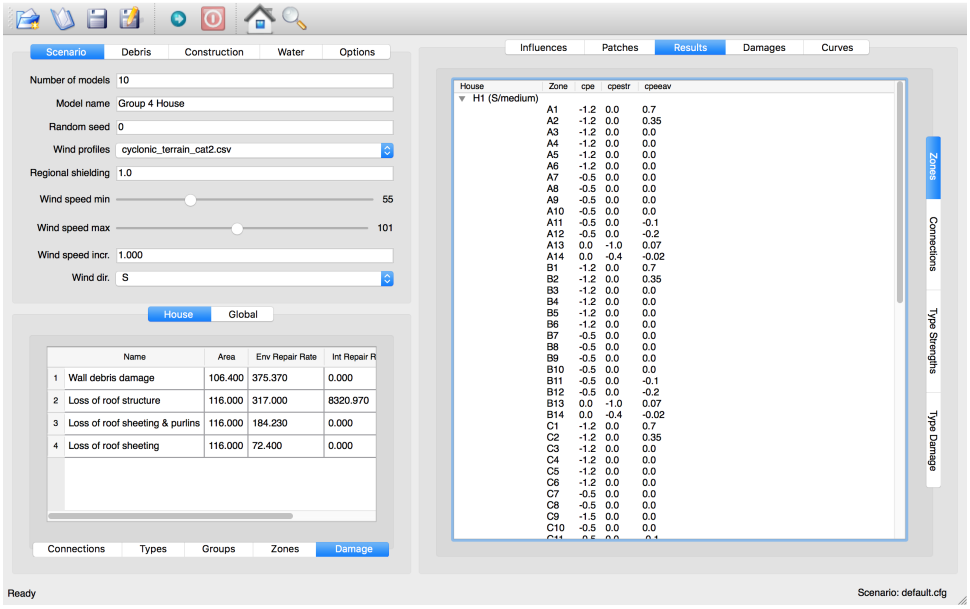

Fig. 4.10: Display of Cpe values for each zone

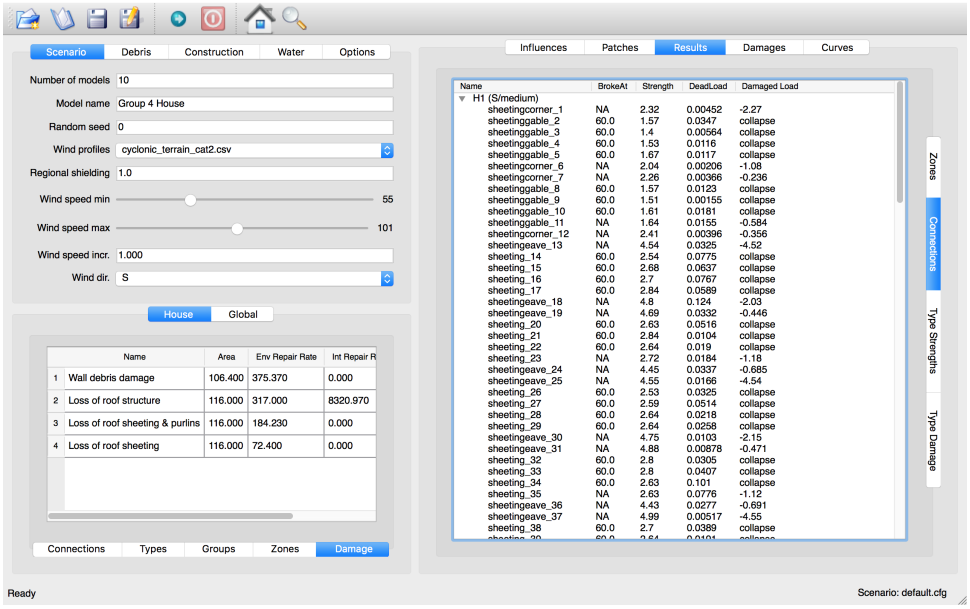

Fig. 4.11: Display of strength, dead load, and failure wind speed for each connection

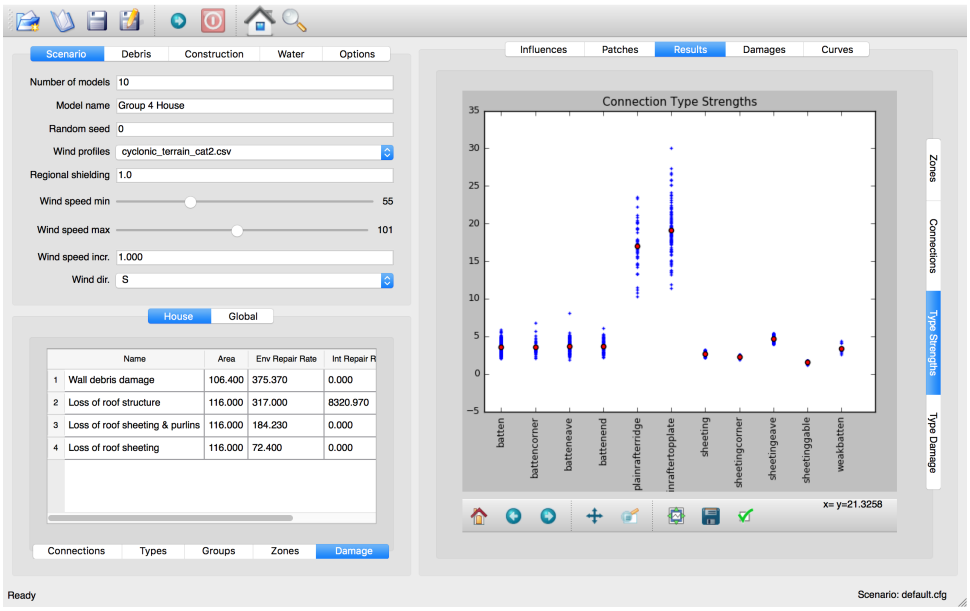

Fig. 4.12: Display of distribution of sampled connection strength by connection type

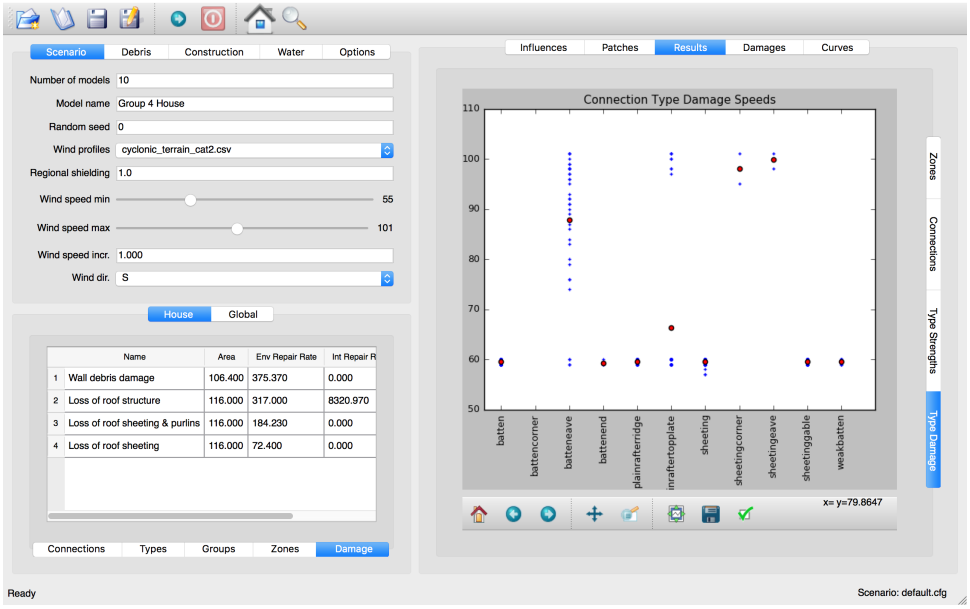

Fig. 4.13: Display of distribution of failure wind speed by connection type

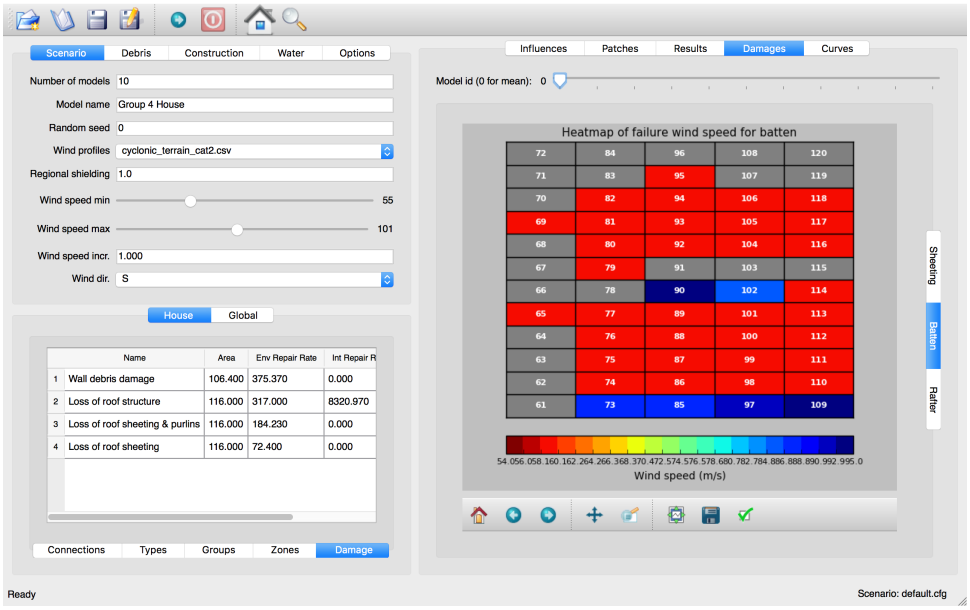

Fig. 4.14: Heatmap of failure wind speed averaged across models for batten group

Curves tab

The Curves tab shows curves in four sub-windows: Vulnerability, Fragility, Water Ingress, and Debris. The Vulnerability window shows a scatter plot of damage indices at each wind speed along with two fitted vulnerability curves, one of which is fitted to cumulative lognormal distribution function as (4.1) and the other one is to cumulative Weibull distribution function as (4.2). The estimated parameter values are displayed at the top.

$$F_X(x; m, \sigma) = \Phi\left(\frac{\ln(x/m)}{\sigma}\right) \tag{4.1}$$

where Φ: the cumulative distribution function of the standard normal distribution, m: median, and σ: logarithmic standard deviation.

$$F(x; \alpha, \beta) = 1 - \exp\left[-\left(\frac{x}{e^{\beta}}\right)^{\frac{1}{\alpha}}\right] \tag{4.2}$$

An example plot is shown in Fig. 4.15.

The Fragility window shows fragility curves for discrete damage states, which are fitted to cumulative lognormal distribution function, as shown in Fig. 4.16.

The Water Ingress window shows a scatter plot of the costing associated with water ingress along wind speed, as shown in Fig. 4.17.

The Debris window shows 1) number of generated debris items, 2) number of impacted debris items, and 3) proportion of model breached by debris along the range of wind speed, as shown in Fig. 4.18.

4.1.5 Bottom

At the bottom of the main window, configuration file name and status of current simulation are displayed as shown in Fig. 4.19.

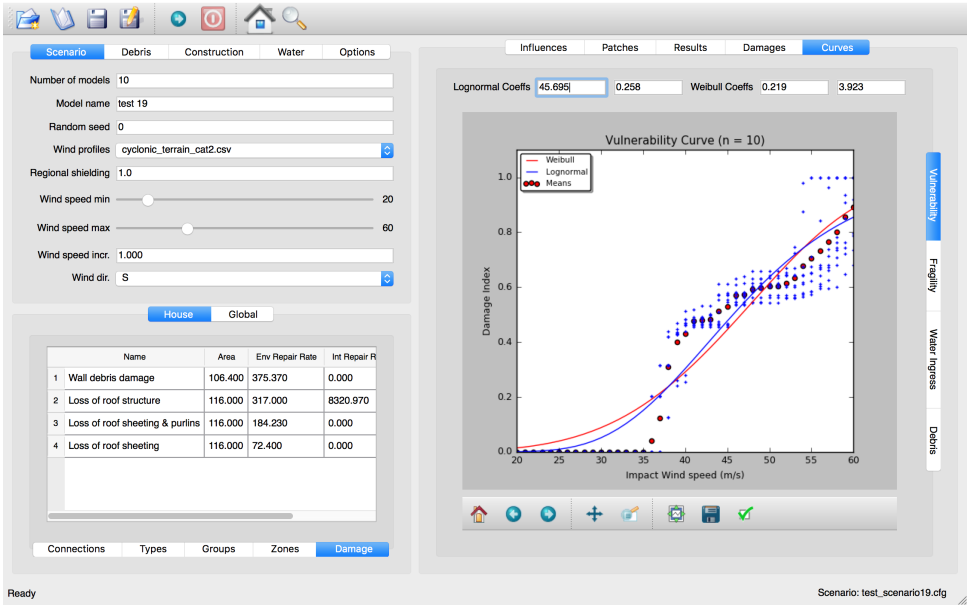

Fig. 4.15: Plot in the Vulnerability window

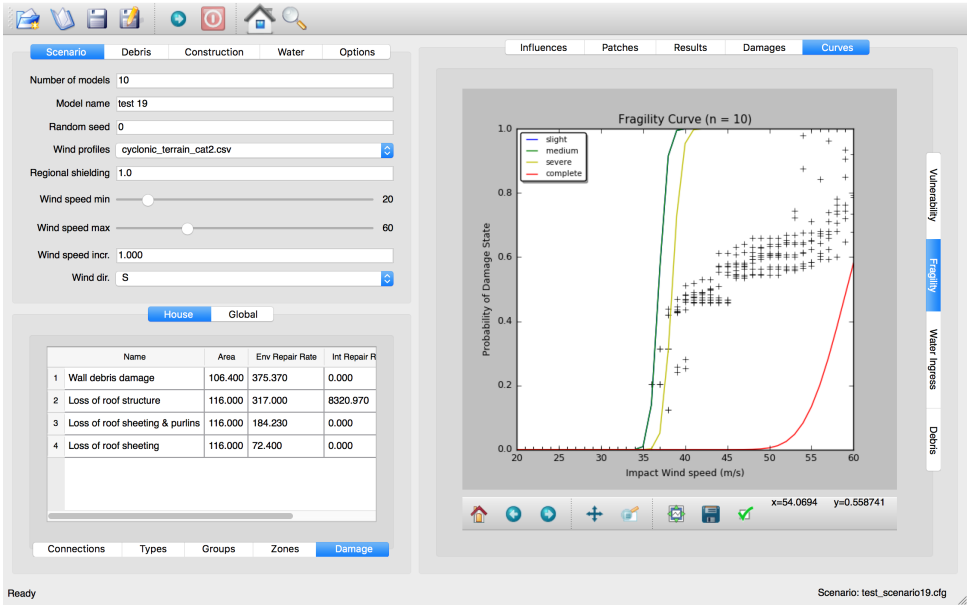

Fig. 4.16: Plot in the Fragility window

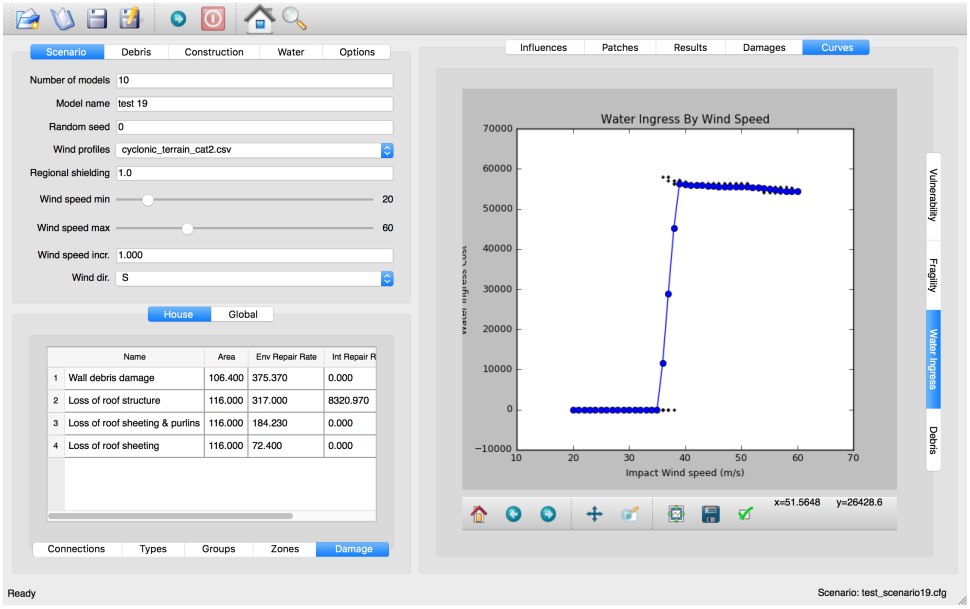

Fig. 4.17: Plot in the Water Ingress window

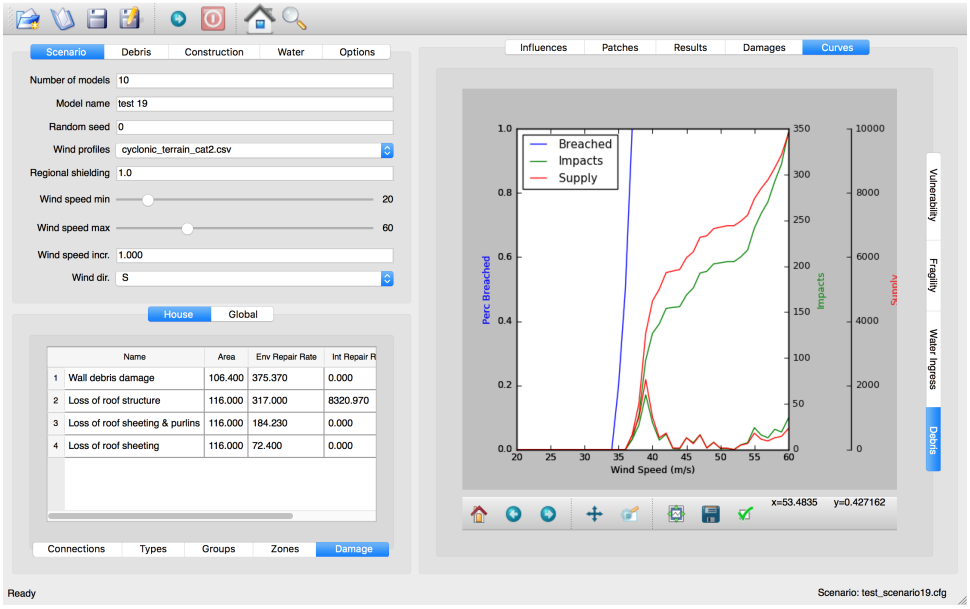

Fig. 4.18: Plot in the Debris window

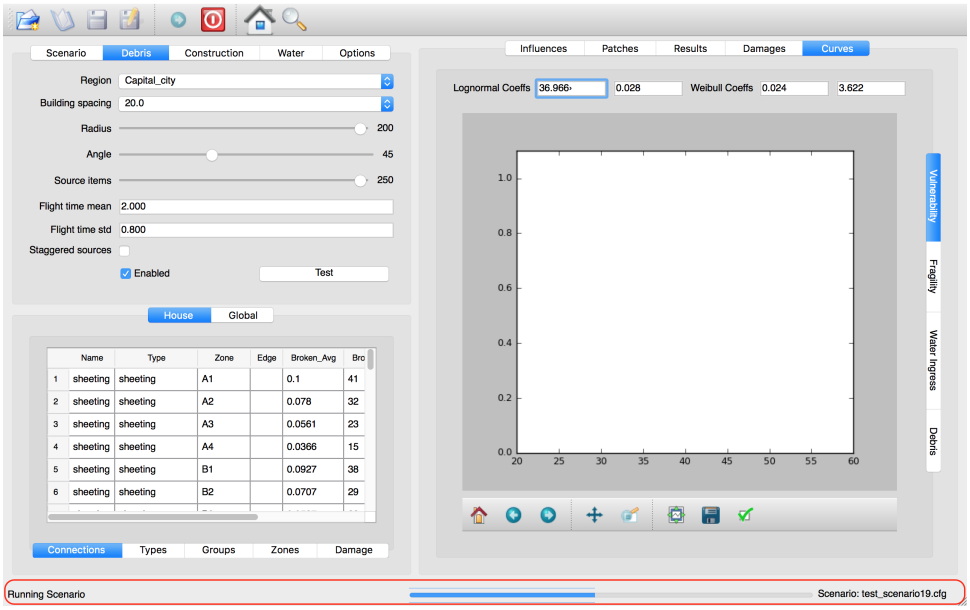

Fig. 4.19: Display of configuration file and status of simulation

4.2 Running simulations

CHAPTER

FIVE

PROGRAM LOGIC

This chapter describes the logic of the program.

5.1 Overall logic

The program is built around the following high level sequence:

1. Create a group of models by random sampling

- For each model (*House*)
  - sample wind direction (*House.set_wind_orientation()*)
  - sample wind profile (*House.set_wind_profile()*)
  - sample construction quality level (*House.set_construction_level()*)
  - set up coverages given the wind direction (*House.set_coverages()*)
  - set up connections (*House.set_connections()*)
  - set up zones (*House.set_zones()*)
  - set up debris generation model (*House.set_debris()*)

2. Calculate damage indices of the models over a range of wind speeds

- For each wind speed
  - calculate damage index for each model (*HouseDamage.run_simulation()*)
    - assign the increment in the mean damage index from the previous wind step as an input to the debris generation model (*Debris.no_items_mean*)
    - calculate free stream wind pressure (qz), optionally applying a regional shielding factor (*HouseDamage.compute_qz_ms()*)
    - calculate zone pressures (*Zone.calc_zone_pressure()*)
    - check damage of envelope coverages by wind load (*Coverage.check_damage()*)
    - calculate connection loads (*Connection.compute_load()*)
    - check damage of each connection by connection group (*ConnectionTypeGroup.check_damage()*)
    - check damage and compute damaged area by connection group (*ConnectionTypeGroup.compute_damaged_area()*)

    * update influence by connection group (*ConnectionTypeGroup.update_influence()*)
    * check for total house collapse event (*HouseDamage.check_house_collapse()*)
    * compute damage index of the model (*HouseDamage.compute_damage_index()*)
    * generate debris and update Cpi in case of internal pressurisation event (*HouseDamage.check_internal_pressurisation()*)
  - calculate increment in mean damage index of the group of models (*update_bucket()*)
3. Fit fragility and vulnerability curves and save outputs (*save_results_to_files()*)

5.2 Detailed logic

This section provides detailed descriptions of each module.

5.2.1 House module

wind direction

The wind direction is set up at the time of model creation, and kept constant during the simulation over a range of wind speeds. If wind_direction (Table 3.1) is 'RANDOM', then wind direction is randomly sampled among the eight directions.

wind profile

A set of gust envelope wind profiles is read from wind_profiles (Table 3.1), and one profile is randomly chosen for each model and kept constant during the simulation over a range of wind speeds. And mzcat value at the model height is then calculated by interploation using the sampled profile over height.

construction quality level

A set of mean and cov factors for connection strength is defined for each construction quality level with likelihood as listed in Table 3.4. Construction level for each model is determined from a random sampling, and the corresponding mean and cov factors are later multiplied to arithmetic mean and standard deviation of strength as (5.1):

$$\begin{aligned} \mu_{adj} &= \mu \times f_{\mu} \\ \sigma_{adj} &= \sigma \times f_{\mu} \times f_{\text{cov}} \end{aligned} \tag{5.1}$$

where μ_{adj} and σ_{adj}: adjusted mean and standard deviation of connection strength reflecting construction quality level, respectively, μ and σ: mean and standard deviation of connection strength, f_{μ} and f_{cov}: mean and cov factors for connection strength.

regional shielding factor

If the value of the regional shielding factor is greater than 0.85, then Ms is set to be 1.0, and no adjustment of wind speed is required. When the value is less or equal to 0.85 then Ms is sampled from a probability mass function, which has 1.0, 0.85, and 0.95 with likelihood of 0.63, 0.15, and 0.22, respectively. And the sampled value of Ms is used to adjust wind speed as (5.2):

$$V_{adj} = V \times Ms/R \tag{5.2}$$

where V_{adj}: adjusted wind speed reflecting the regional shielding factor, V: wind speed, Ms:, R: regional shielding factor.

connection load

The load applied for each of connections are calculated as (5.3):

$$L_i = D_i + \sum_{j=1}^{N_z} (I_{ji} \times A_j \times P_j) + \sum_{j=1}^{N_c} (I_{ji} \times L_j) \tag{5.3}$$

where L_i: applied load for i th connection, D_i: dead load of i th connection, N_z: number of zones associated with the i th connection, N_c: number of connections associated with the i th connection, A_j: area of j th zone, P_j: wind pressure on j th zone, I_{ji}: influence coefficient from j th either zone or connection to i th connection.

5.2.2 Debris damage module

The methdology of modelling damage from wind-borne debris implemented in the code is described in the *[JDH2010d]* and *[JDH2010d]*. The debris damage module consists of four parts: 1) debris generation, 2) debris trajectory, 3) debris impact, and 4) debris damage costing.

debris generation

The mean number of debris items to be generated (N_{mean}) is calculated by (5.4).

$$N_{mean} = \text{nint}\left(\Delta DI \times N_{items}\right) \tag{5.4}$$

where N_{items}: number of debris items per source defined in *3.1.3*, ΔDI: increment in damage index from previous wind step, and nint: nearest integer function.

The number of generated debris items is assumed to follow the Poisson distribution with parameter $\lambda = N_{mean}$. For each debris source, the number of generated debris items is randomly sampled from the distribution, and debris type is randomly chosen as many as number of items with probability proportional to the ratio of each type defined in Table 3.9. The debris types are provided in *3.2*:

debris trajectory

For each generated debris item, mass, frontal area, and flight time are sampled from the lognormal distribution with parameter values provided in *3.1.3* and *3.2*. The flight distance is calculated based on the methodology presented in the Appendix of Lin and Vanmarcke (2008). Note that the original fifth

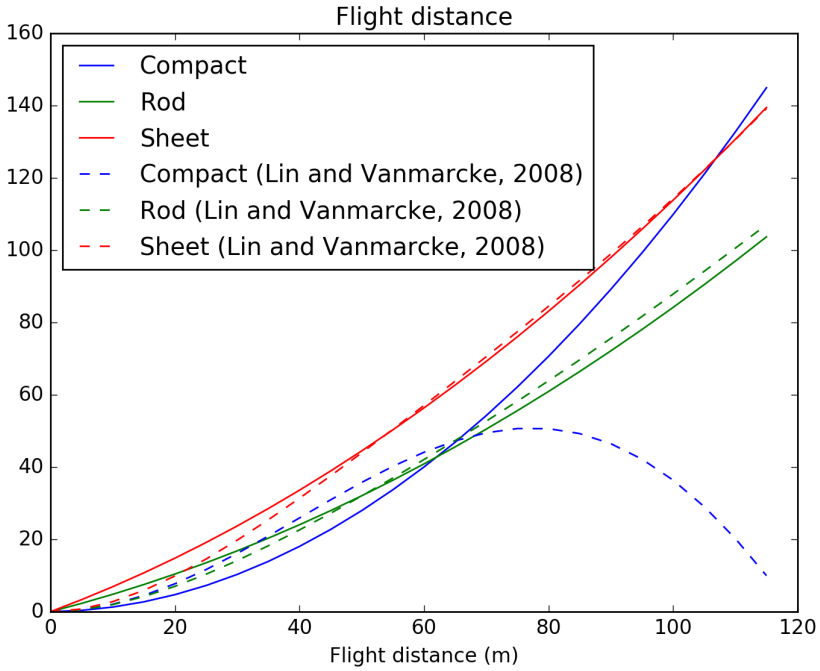

Fig. 5.1: Flight distance of debris item

polynomial functions are replaced with quadratic one with the coefficients as listed in Table 5.1. The computed flight distance by debris type using the fitth and quadratic polynomials is shown in Fig. 5.1.

Table 5.1: Coefficients of quadratic function for flight distance computation by debris type

Debris type	linear coeff.	quadratic coeff.
Compact	0.011	0.2060
Rod	0.2376	0.0723
Sheet	0.3456	0.072

The probability distribution of point of landing of the debris in a horizontal plane is assumed to follow a bivariate normal distribution as (5.5).

$$f_{x,y} = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left[-\frac{(x-d)^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right] \tag{5.5}$$

where x and y are the coordinates of the landing position of the debris, σ_x and σ_y: standard deviation for the coordinates of the landing position, and d: expected flight distance. The value of σ_x and σ_y are set to be $d/3$ and $d/12$, respectively.

If the line linking the source to the landing point intersects with the footprint of the model, then it can be assumed that an impact has occurred.

Following Lin and Vanmarcke 2008, the ratio of horizontal velocity of the windborne debris object to the wind gust velocity is modelled as a random variable with a Beta distribution as (5.6).

$$\frac{u_m}{V_s} \sim Beta(\alpha, \beta) \tag{5.6}$$

where u_m: the horizontal velocity of the debris object, V_s: the local (gust) wind speed, α and β are two

parameters of the Beta distribution and estimated as (5.7).

$$\begin{aligned} \alpha &= E \times \nu \\ \beta &= \nu \times (1 - E) \end{aligned} \tag{5.7}$$

where E: the expected value and $\nu = \alpha + \beta$.

The expected value (E) and the parameter (ν) are assumed to be as (5.8).

$$\begin{aligned} E &= 1 - \exp\left(-b\sqrt{x}\right) \\ \nu &= \max\left[\frac{1}{E}, \frac{1}{1-E}\right] + 3.0 \end{aligned} \tag{5.8}$$

where x: the flight distance, b: a dimensional parameter calucalted as (5.9). If E is 1, then α and β are assigned with 3.996 and 0.004, respectively.

$$b = \sqrt{\frac{\rho_a C_{D,av} A}{m}} \tag{5.9}$$

where ρ_a: the air density, $C_{D,av}$: an average drag coefficient, A: the frontal area, and m: the mass of the object.

The momentum ξ is calculated using the sampled value of the ratio, $\frac{u_m}{V_s}$ as (5.10).

$$\xi = \left(\frac{u_m}{V_s}\right) \times m \times V_s \tag{5.10}$$

debris impact : original method

Based on the methodology presented in HAZUS and Lin and Vanmacke (2008), the number of impact N is assumed to follow a Poisson distribution as (5.11).

$$\begin{aligned} N &\sim \text{Pois}(\lambda) \\ \lambda &= N_v \cdot q \cdot F_\xi(\xi > \xi_d) \end{aligned} \tag{5.11}$$

where N_v: number of impacts at a single wind speed, q: proportion of coverage area out of the total area of envelope, F_ξ: the cumulative distribution of momentum, and xi_d: threshold of momentum or energy for damage of the material of the coverage.

The probability of damage can be calculated based on the Poisson distribution as (5.12).

$$P_D = 1 - P(N = 0) = 1 - \exp\left[-\lambda\right] \tag{5.12}$$

In the original method, N_v is estimated from the all debris sources, and for each coverage, q and $F_\xi(\xi > \xi_d)$ are estimated. If the material of the coverage is glass, then P_D is computed and compared against a random value sampled from unit uniform distribution to determine whether the coverage is failed or not. For coverage with non-glass material, a random value of number of impact is sampled from the Poisson distribution with λ, which is regarded as the damaged area assuming that affected area by debris impact is 1.

debris impact : alternative method

In the alternative method, instead of using the Poisson distribution, it is possible to check whether impact occurs or not each debris item through simulation. For each debris impact, target coverage is determined

based on the area proportion. And the momentum of the debris is computed as (5.10), and then compared against the capacity of the coverage, which is also sampled from the assumed distribution. Once the momentum is greater than the capacity, then the frontal area of the debris is regarded as damaged area for non-glass coverage, while whole coverage area is for glass one.

The following plots show the differences in results between the two approaches:

1. Simulation using the Tropical_town vulnerability

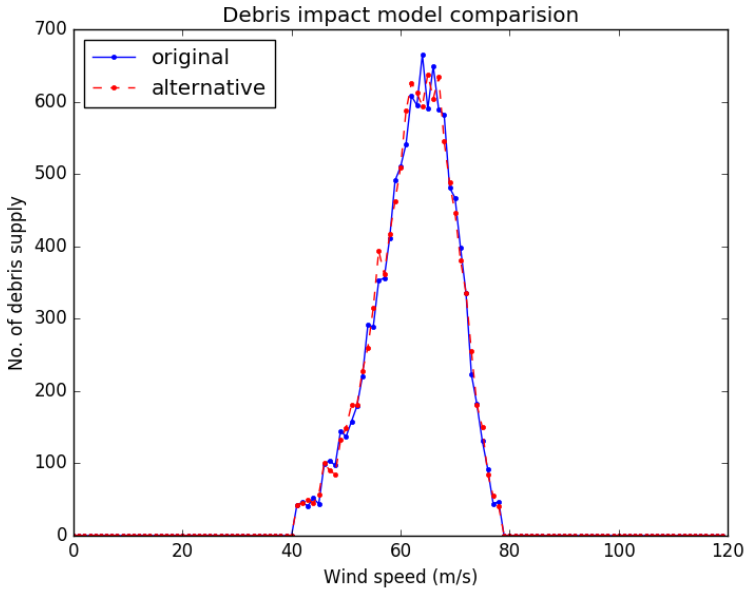

Fig. 5.2: No. of supply using the Tropical_town vulnerability

Note that original_sampled represents the number of impacts sampled using the Poisson distribution,which is later used in estimating damaged area.

As explained earlier, the damaged area from the original method is assumed to be 1.0 times number of impacts, which is way larger than estimated frontal area of the debris, which explains the difference.

2. simulation using the Capital_city vulnerability
3. Use of amplification factor

To reduced the difference between the two approaches, amplification factor is used, which is multiplied to the frontal area of debris item for the damaged area. For this exercise, amplification factor is set to be 5.

The debris sources are generated by calling `Debris.create_sources()`, which requires a number of parameters as shown in the Fig. 5.10.

Depending on the value for *staggered_sources*, Fig. 5.11 and Fig. 5.12 are displayed.

5.2.3 Water ingress

The damage cost induced by water ingress is estimated over the following three steps:

1. estimate amount of water ingress (`compute_water_ingress_given_damage()`)

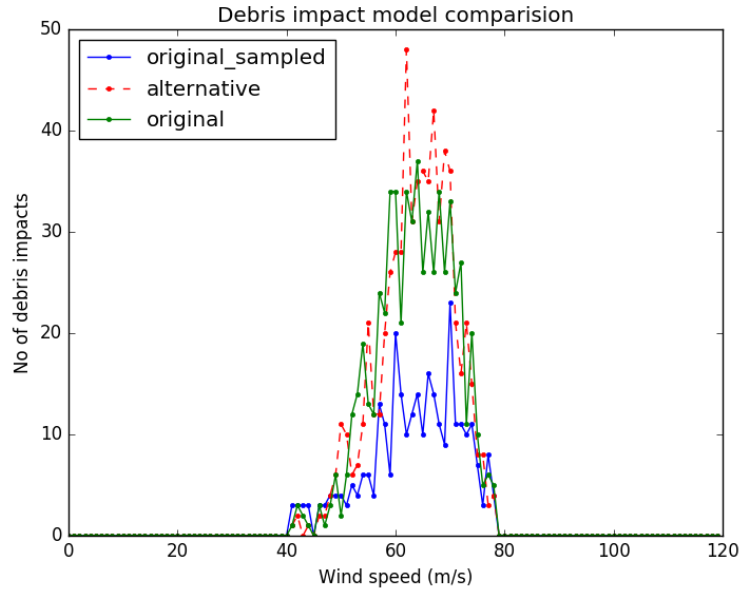

Fig. 5.3: No. of impacts (or touched) using the Tropical_town vulnerability

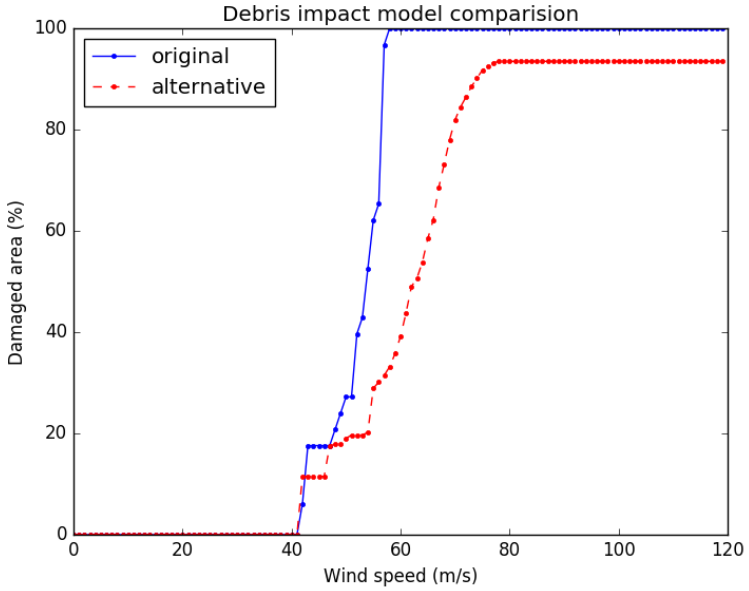

Fig. 5.4: Damaged area using the Tropical_town vulnerability

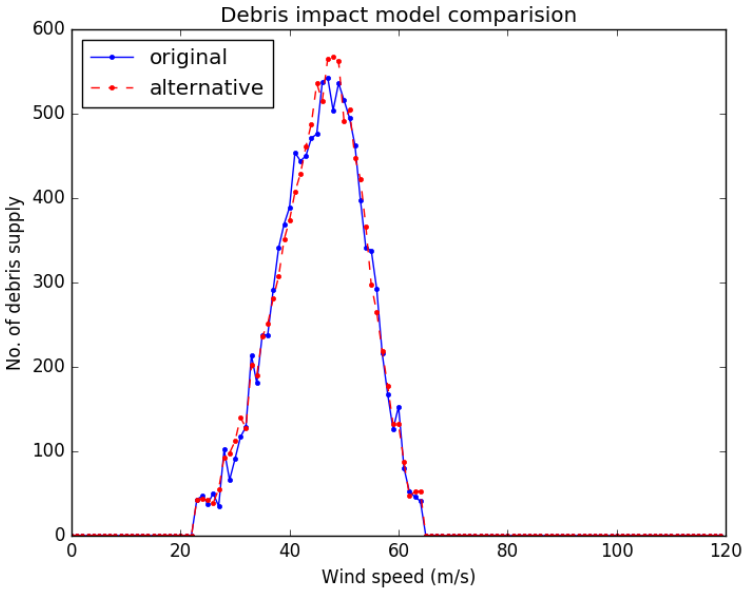

Fig. 5.5: No. of supply using the Capital_city vulnerability

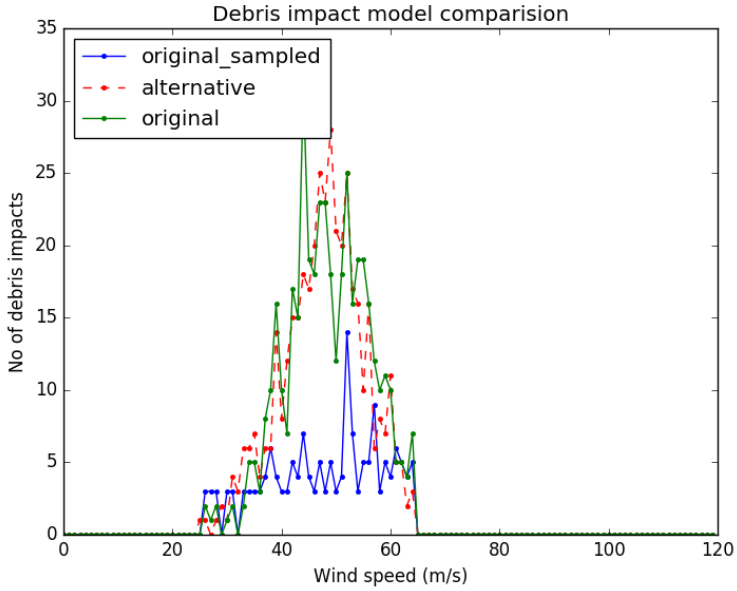

Fig. 5.6: No. of impacts (or touched) using the Capital_city vulnerability

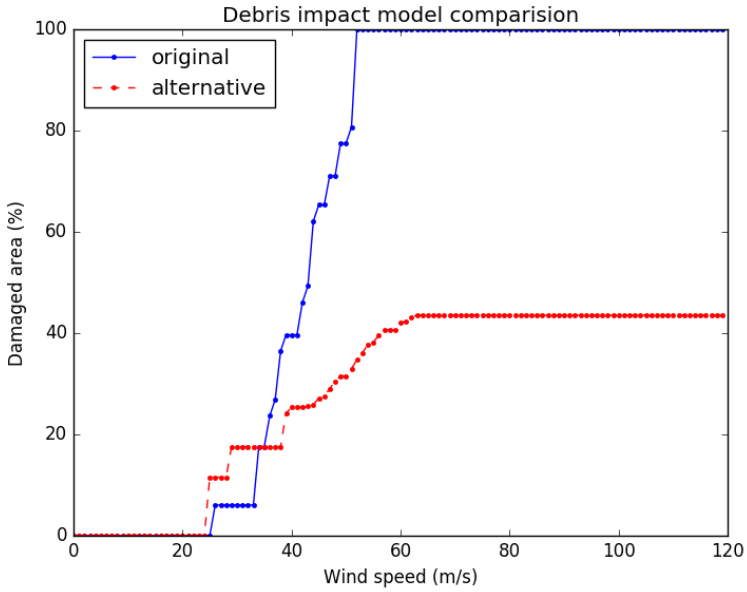

Fig. 5.7: Damaged area using the Capital_city vulnerability

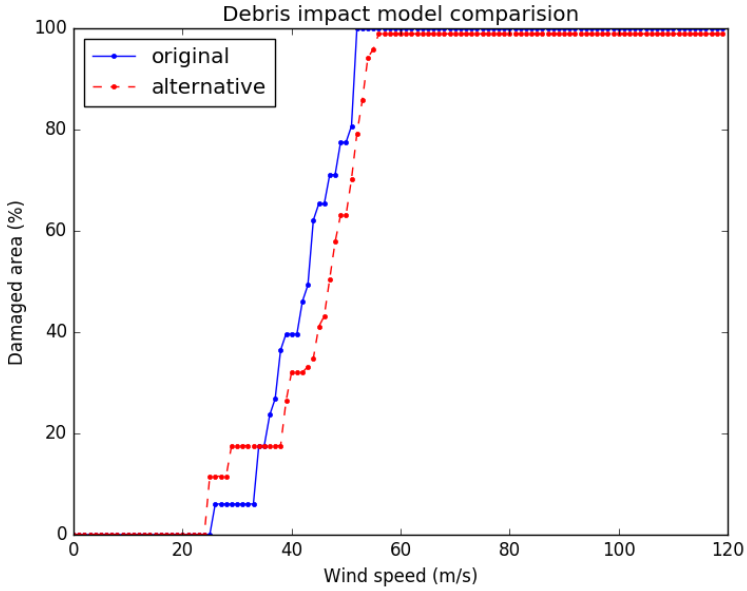

Fig. 5.8: Damaged area using the Capital_city vulnerability: amplified with 5

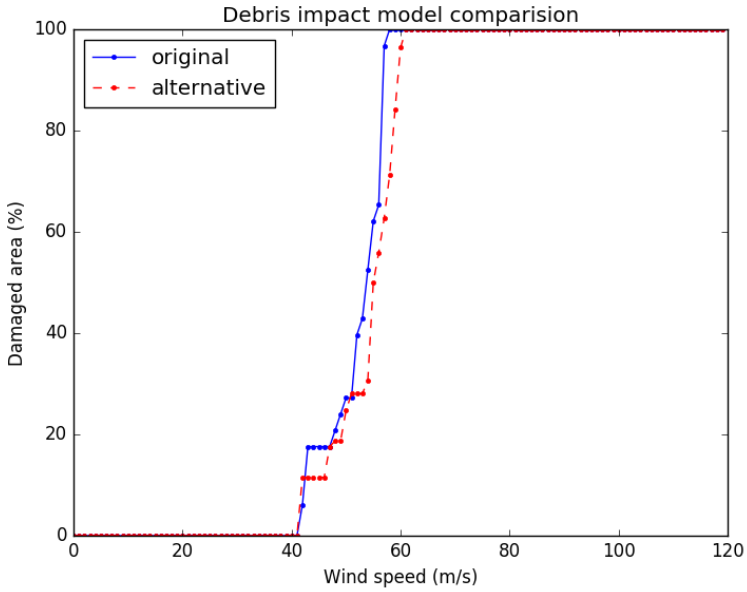

Fig. 5.9: Damaged area using the Tropical_town vulnerability: amplfied with 5

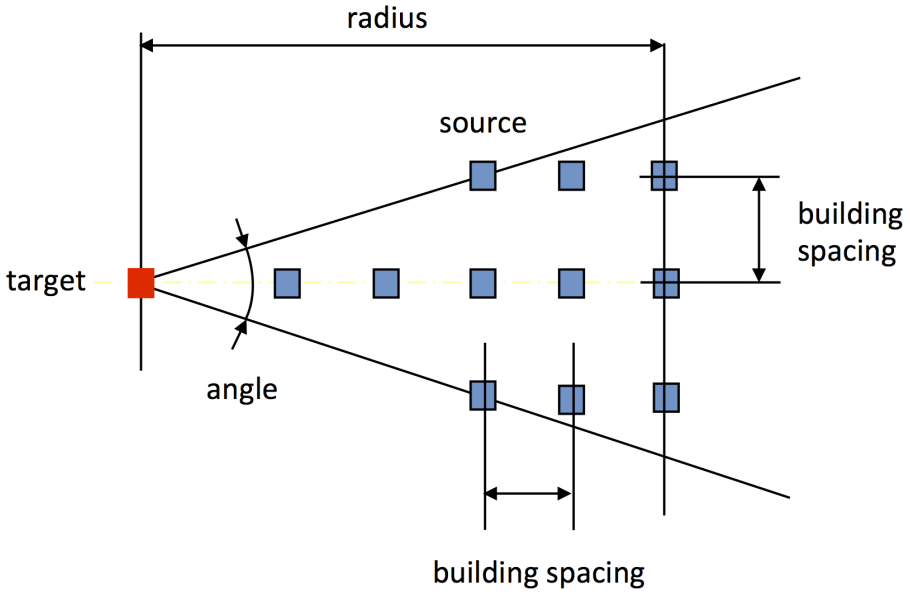

Fig. 5.10: Distribution of debris sources with parameters

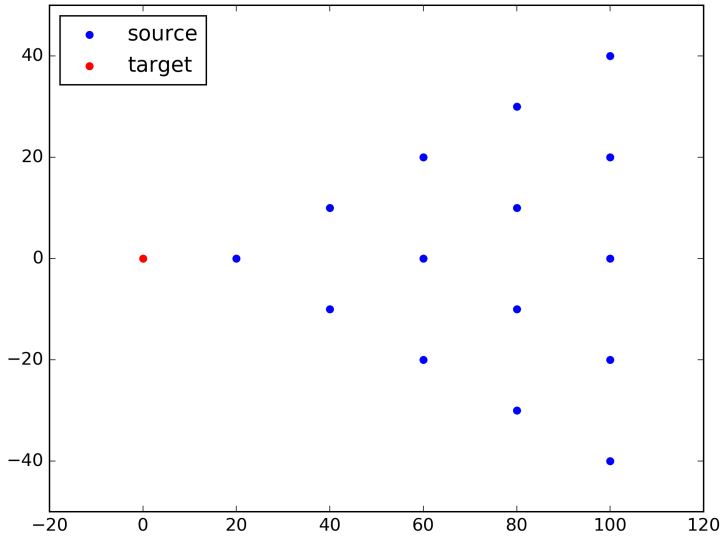

Fig. 5.11: Distribution of debris source buildings generated with debris_radius = 100.0 (m), debris_angle = 45.0 (deg), debris_space = 20.0 (m), and staggered_sources = *True*.

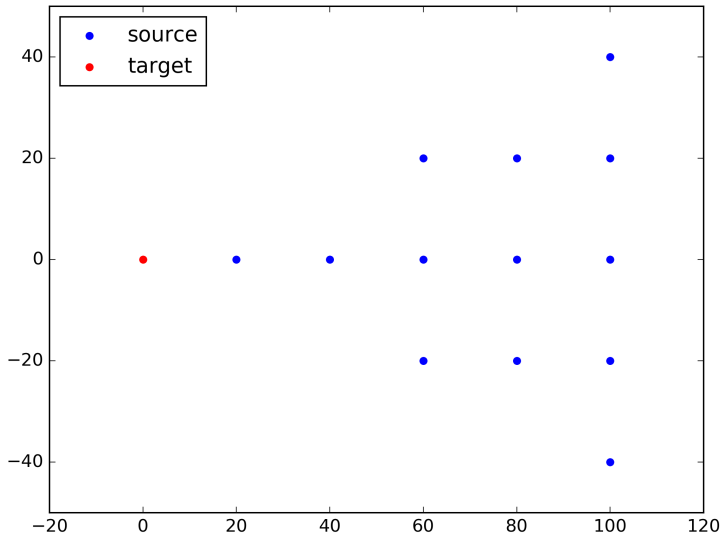

Fig. 5.12: Distribution of debris source buildings generated with debris_radius = 100.0 (m), debris_angle = 45.0 (deg), debris_space = 20.0 (m), and staggered_sources = *False*.

The amount of water ingress is estimated based on the relationship between water ingress and wind speed, which is listed in Table 3.5. The estimated damage index prior to water ingress is used to choose the right curve as shown in Fig. 3.4.

2. determine damage scenario (*HouseDamage.determine_scenario_for_water_ingress_costing(*

The damage scenario for water ingress is determined based on the order of damage scenario as listed in Table 3.20. One damage scenario is selected by the order among the damage scenarios with which damage area of connection associated is greater than zero. When the damage index is zero (or no connection damage yet), then damage scenario of 'WI only' is used.

3. calculate cost for water ingress damage (*HouseDamage.compute_water_ingress_cost()*)

The cost for water ingress damage is estimated using the data provided in *3.4.17*. The example plot for the scenario of *loss of roof sheeting* is shown in Fig. 5.13. The cost for water ingress damage is estimated using the curve for water ingress closest to the estimated amount of water ingress.

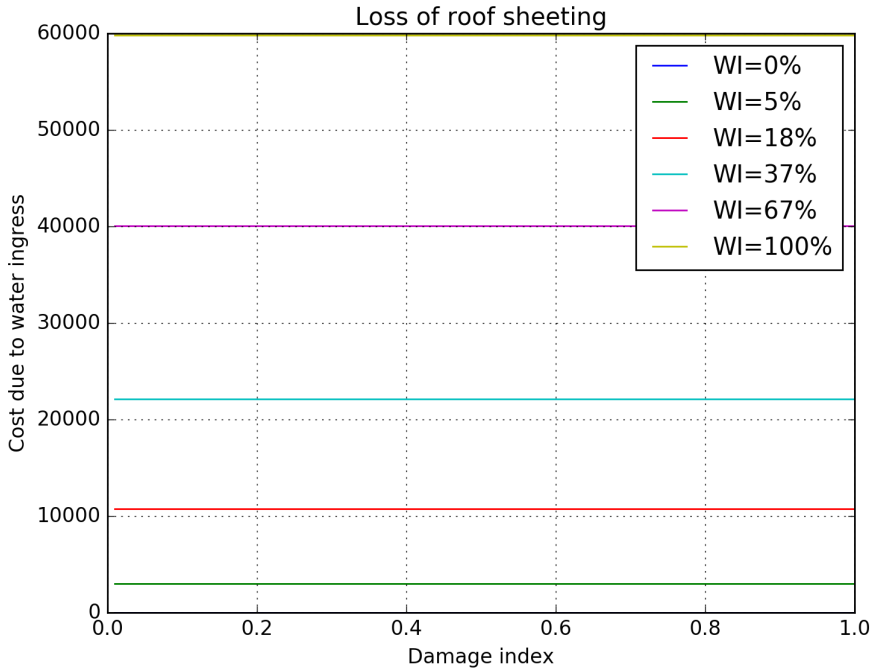

Fig. 5.13: Relationship between cost due to water ingress damage and damage index

5.2.4 Cpe

The external pressure coefficient, C_{pe} is used in computing zone pressures, and is sampled from Type III extreme value distribution (`stats.sample_gev()`) which has the cumulative distribution function and probability density as (5.13) and (5.14), respectively.

$$F(s;k) = \exp(-(1-ks)^{1/k}) \tag{5.13}$$

$$f(s;a,k) = \frac{1}{a}(1-ks)^{1/k-1}\exp(-(1-ks)^{1/k}) \tag{5.14}$$

where $s = (x - u)/a$, u: location factor ($\in \mathbb{R}$), a: scale factor (> 0), and k: shape factor ($k \neq 0$).

The mean and standard deviation are calculated as (5.15):

$$\begin{aligned} \mathrm{E}(X) &= u + \frac{a}{k}\left[1 - \Gamma(1+k)\right] \\ \mathrm{SD}(X) &= \frac{a}{k}\sqrt{\Gamma(1+2k) - \Gamma^2(1+k)} \end{aligned} \tag{5.15}$$

The u and a can be estimated given $c_v \left(= \frac{SD}{E}\right)$ and k values as (5.16):

$$\begin{aligned} a &= \mathrm{E}\,\frac{c_v}{B} \\ u &= \mathrm{E} - a \times A \end{aligned} \tag{5.16}$$

where $A = (1/k)\left[1 - \Gamma(1+k)\right]$ and $B = (1/k)\sqrt{\Gamma(1+2k) - \Gamma^2(1+k)}$.

5.2.5 Calculate damage index

The damage index is calculated over the following steps:

1. calculate sum of damaged area by connection group (`HouseDamage.compute_area_by_group()`)
2. Apply damage factoring (`HouseDamage.apply_damage_factoring()`)

In order to avoid double counting of repair cost, damage cost associated with child group(s) will be factored out of damage cost of the parent group as explained in *3.4.16*.

3. Calculate sum of damaged area by damage scenario (`HouseDamage.compute_area_by_scenario()`)

A damage scenario is assigned to each connection type group as explained in *3.4.2*.

4. calculate total damage cost and damage index prior to water ingress (DI_p) as (5.17):

$$DI_p = \frac{\sum_{i=1}^{S} C_i}{R} \tag{5.17}$$

where S: number of damage scenario, C_i: damage cost for i th damage scenario, and R: total replacement cost.

5. Calculate cost by water ingress damage, C_{wi} if required as explained in *5.x*
6. calculate damage index as (5.18):

$$DI = \frac{\sum_{i=1}^{S} C_i + C_{wi}}{R} \tag{5.18}$$

5.2.6 Cpi

The internal pressure coefficient, C_{pi} is determined based on Table 5.2 and Table 5.3 depending on the existence of dominant opening by either coverage failure or debris breach, which are revised from Tables 5.1(A) and 5.1(B) of AS/NZS 1170.2:2011, respectively.

Table 5.2: C_{pi} for buildings without dominant openings

condition	C_{pi}
All walls equally breached	-0.3
Two or three windward walls equally breached	0.2
Two or three non-windward walls equally breached	-0.3

Table 5.3: C_{pi} for buildings with dominant openings

ratio of dominant opening to total open area (r)	dominant opening on windward wall	dominant opening on leeward wall	dominant opening on side wall
$r < 0.5$	-0.3	-0.3	-0.3
$0.5 \leq r < 1.5$	0.2	-0.3	-0.3
$1.5 \leq r < 2.5$	0.7 C_{pe}	C_{pe}	C_{pe}
$2.5 \leq r < 6.0$	0.85 C_{pe}	C_{pe}	C_{pe}
$r \geq 6.0$	C_{pe}	C_{pe}	C_{pe}

5.2.7 Zone module

zone pressure

The zone pressure is calculated as (5.19):

$$P = q_z \times (C_{pe} - \alpha_{C_{pi}} \times C_{pi}) \times d_s \tag{5.19}$$

where q_z:, C_{pe}:, C_{pi}, $\alpha_{C_{pi}}$, and d_s:

differential shielding

If the value of diff_shielding is True, then differential shielding effect is considered in calculating zone pressure. The differential shielding is computed as follows:

```
front_facing = self.is_roof_edge[wind_dir_index]
if building_spacing == 40 and ms >= 1.0 and front_facing == 0:
    dsd = ms ** 2.0
elif building_spacing == 20 and front_facing == 1:
    dsd = ms ** 2.0
    if ms <= 0.85:
        dsn = 0.7 ** 2.0
    else:
        dsn = 0.8 ** 2.0
```

update influence coefficient

The influence coefficient is used to associate one connection with another either zone or connection with regard to load distribution. For instance, if connection 1 has influences of connection 2 and 3 with coefficient 0.5 and 0.5, respectively, then the load on connection 1 is equal to the sum of 0.5 times load on connection 2 and 0.5 times load on connection 3, as shown in (5.3).

Once a connection is failed, then load on the failed connection needs to be distributed to other connections accordingly, which means that influence coefficient needs to be revised.

Given the failure of connection of either sheeting and batten connection type group, the influence coefficient will be distributed evenly to the next connection of the same type to the distribution direction (*dist_dir* listed in Table 3.11). Note that *patch_dist* of both sheeting and batten connection group are set to be *False*.

Unlike sheeting and batten, a connection of Rafter group fails, then influence coefficients associated with the failed connection are replaced with a new set of influence coefficients, which is termed "patch". In the current implementation, the patch is defined for a single failed connection. Thus the failure order of the connections may make difference in the resulting influences as shown in Table 5.4.

Table 5.4: Example of how patch works

Failed connection	Connection	Patch (connection: influence coeff.)
1	3	1:0.0, 2:0.5, 3:0.5
2	3	1:0.5, 2:0.0, 3:1.0
1 and then 2	3	1:0.0, 2:0.0, 3:1.0
2 and then 1	3	1:0.0, 2:0.0, 3:0.5

coverage

The coverages are making up the wall part of the envelope of the model. Two failure mechanism are implemented: 1) failure by wind load and 2) failure by windborne debris. The coverage failure by wind load is very similar to the failure of connection by wind, in which wind load is first calculated as (5.20) and then compared against its strengths in both directions to check the failure.

$$L = 0.9 \times q_z \times (C_{pe} - C_{pi}) \times A \tag{5.20}$$

vulnerability

$$P = q_z \times (C_{pe} - \alpha_{C_{pi}} \times C_{pi}) \times d_s \tag{5.21}$$

connection Fragility "implies easily damaged or broken, but is often used to

connection type Fragility "implies easily damaged or broken, but is often used to

connection type group Fragility "implies easily damaged or broken, but is often used to

coverage Fragility "implies easily damaged or broken, but is often used to

C_{pe} external pressure coefficient

C_{pi} internal pressure coefficient

damage index Fragility "implies easily damaged or broken, but is often used to

debris Fragility "implies easily damaged or broken, but is often used to

fragility Fragility "implies easily damaged or broken, but is often used to describe the probability of a stated level of damage for a specific hazard, e.g. an earthquake." Fragility measures probability.

fragility function

fragility curves Fragility function is a mathematical function that expresses the relationship between the probability of occurrence of some undesirable event and some measure of environmental excitation. In our case the the undesirable event is a facility or component reaching or exceeding some clearly defined limit state, and the environmental excitation is an earthquake of a defined intensity measure.

influence coefficient coefficient relating a connection to either zone or connection with regard to load distribution

patch Fragility "implies easily damaged or broken, but is often used to

q_z free stream wind pressure

vulnerability Vulnerability refers to the concept of susceptibility of damage for a given entity. The entity can be a civil structure, a critical infrastructure facility, a component within such a facility, or a subset of population within a defined geographical area, etc. Vulnerability measures loss.

vulnerability function Vulnerability function is a mathematical function that depicts loss as a function of environmental excitation. It has several synonyms: vulnerability curves, damage functions, loss functions.

wind profile Fragility "implies easily damaged or broken, but is often used to

zone Fragility "implies easily damaged or broken, but is often used to

CHAPTER

SIX

VAWS PACKAGE

6.1 vaws.gui package

6.1.1 vaws.gui.house module

class vaws.gui.house.**HouseViewer**(*cfg*, *parent=None*)

Bases: PyQt4.QtGui.QDialog, vaws.gui.house_ui.Ui_Dialog, *vaws.gui.mixins.PersistSizePosMixin*

accept()

reject()

vaws.gui.house.**bin**(*QTextStream*) → QTextStream

vaws.gui.house.**hex**(*QTextStream*) → QTextStream

vaws.gui.house.**oct**(*QTextStream*) → QTextStream

6.1.2 vaws.gui.main module

class vaws.gui.main.**MyForm**(*parent=None*, *init_scenario=None*)

Bases: PyQt4.QtGui.QMainWindow, vaws.gui.main_ui.Ui_main, *vaws.gui.mixins.PersistSizePosMixin*

closeEvent(*event*)

determine_capacities(*bucket*)

file_load(*fname*)

heatmap_house_change()

init_debris_region()

init_influence_and_patch()

init_terrain_category()

okToContinue()

onSliderChanged(*label*, *x*)

open_scenario(*config_file=None*)

runScenario()

save_as_scenario()

save_scenario()

showHouseInfoDlg()

stopScenario()

testConstructionLevels()

testDebrisSettings()

testWaterIngress()

updateBreachPlot(*bucket*)

updateComboBox()

updateConnectionGroupTable()

updateConnectionTable()

updateConnectionTable_with_results(*bucket*)

updateConnectionTypePlots(*bucket*)

updateConnectionTypeTable()

updateDamageTable()

updateDebrisRegionsTable()

updateDisplaySettings()

updateFragCurve(*_array*)

updateHeatmap(*bucket*, *house_number=0*)

updateHouseResultsTable(*bucket*)

updateInfluence()

updatePatch()

updateStrengthPlot(*bucket*)

updateTerrainCategoryTable()

updateTypeDamagePlot(*bucket*)

updateVulnCurve(*_array*)

updateWaterIngressPlot(*bucket*)

updateZonesTable()

update_config_from_ui()

update_house_panel()

update_ui_from_config()

vaws.gui.main.**progress_callback**(*percent_done*)

vaws.gui.main.**run_gui**()

6.1.3 vaws.gui.matplotlibwidget module

MatplotlibWidget

Example of matplotlib widget for PyQt4

Copyright © 2009 Pierre Raybaut This software is licensed under the terms of the MIT License

Derived from 'embedding_in_pyqt4.py': Copyright © 2005 Florent Rougon, 2006 Darren Dale

class vaws.gui.matplotlibwidget.**MatplotlibWidget**(*parent=None*, *title=''*, *xlabel=''*, *ylabel=''*, *xlim=None*, *ylim=None*, *xscale='linear'*, *yscale='linear'*, *width=4*, *height=3*, *dpi=100*, *hold=False*)

Bases: PyQt4.QtGui.QWidget

class vaws.gui.matplotlibwidget.**MatplotlibWidget2**(*parent=None*, *title=''*, *xlabel=''*, *ylabel=''*, *xlim=None*, *ylim=None*, *xscale='linear'*, *yscale='linear'*, *width=4*, *height=3*, *dpi=100*, *hold=False*)

Bases: matplotlib.backends.backend_qt4agg.FigureCanvasQTAgg

MatplotlibWidget inherits PyQt4.QtGui.QWidget and matplotlib.backend_bases.FigureCanvasBase

parent (None): parent widget title (''): figure title xlabel (''): X-axis label ylabel (''): Y-axis label xlim (None): X-axis limits ([min, max]) ylim (None): Y-axis limits ([min, max]) xscale ('linear'): X-axis scale yscale ('linear'): Y-axis scale width (4): width in inches height (3): height in inches dpi (100): resolution in dpi hold (False): if False, figure will be cleared each time plot is called

figure: instance of matplotlib.figure.Figure axes: figure axes

self.widget = MatplotlibWidget(self, yscale='log', hold=True) from numpy import linspace x = linspace(-10, 10) self.widget.axes.plot(x, x**2) self.wdiget.axes.plot(x, x**3)

minimumSizeHint()

sizeHint()

6.1.4 vaws.gui.mixins module

class vaws.gui.mixins.**PersistSizePosMixin**(*name*)

Bases: object

initSizePosFromSettings()

storeSizePosToSettings()

vaws.gui.mixins.**finiTable**(*t*)

vaws.gui.mixins.**setupTable**(*t*, *l=None*)

6.1.5 vaws.gui.output module

output.py - output module, postprocess and plot display engine

class vaws.gui.output.**PlotFlyoverCallback**(*axes*, *data*, *statusbar*, *num_cols*, *num_rows*)

Bases: object

vaws.gui.output.**draw_influence**(*cfg*, *infl_dic*, *dic_ax*, *conn_name*)

vaws.gui.output.**format_coord**(*x*, *y*)

vaws.gui.output.**plot_damage_show**(*fig*, *grouped*, *values_grid*, *xlim_max*, *ylim_max*, *v_min*, *v_max*, *v_step*, *house_number=0*)

vaws.gui.output.**plot_fitted_curve**(*mp_widget*, *v*, *di*, *label='Fitted Curve'*, *alpha=1.0*, *col='b'*)

vaws.gui.output.**plot_fragility_show**(*mp_widget*, *num_iters*, *Vmin*, *Vmax*)

vaws.gui.output.**plot_influence**(*fig*, *cfg*, *conn_name*, *file_name=None*)

vaws.gui.output.**plot_influence_patch**(*fig*, *cfg*, *failed_conn_name*, *conn_name*, *file_name=None*)

vaws.gui.output.**plot_model_curve**(*mp_widget*, *v*, *di*, *label='Model Curve'*)

vaws.gui.output.**plot_wind_event_damage**(*mp_widget*, *v*, *di*)

vaws.gui.output.**plot_wind_event_mean**(*mp_widget*, *v*, *di*)

vaws.gui.output.**plot_wind_event_show**(*mp_widget*, *num_iters*, *Vmin*, *Vmax*)

vaws.gui.output.**set_axis_etc**(*ax*, *title*, *xlim_max*, *ylim_max*)

6.1.6 vaws.gui.windsim_rc module

vaws.gui.windsim_rc.**qCleanupResources**()

vaws.gui.windsim_rc.**qInitResources**()

6.2 vaws.model package

6.2.1 vaws.model.config module

Config module

This module defines basic parameter values for simulation.

vaws.model.config.**OUTPUT_DIR**
"output"

vaws.model.config.**INPUT_DIR**
"input"

vaws.model.config.**DEBRIS_DATA**
"input/debris"

vaws.model.config.**GUST_PROFILES_DATA**
"input/gust_envelope_profiles"

vaws.model.config.**HOUSE_DATAL**
"input/house"

vaws.model.config.**FILE_HOUSE_DATA**
'house_data.csv'

vaws.model.config.**FILE_CONN_GROUPS**
'conn_groups.csv'

vaws.model.config.**FILE_CONN_TYPES**
'conn_types.csv'

vaws.model.config.**FILE_CONNECTIONS**
'connections.csv'

vaws.model.config.**FILE_ZONES**
'zones.csv'

vaws.model.config.**FILE_COVERAGE_TYPES**
'coverage_types.csv'

vaws.model.config.**FILE_COVERAGES**
'coverages.csv'

vaws.model.config.**FILE_COVERAGES_CPE**
'coverages_cpe.csv'

vaws.model.config.**FILE_INFLUENCES**
'influences.csv'

vaws.model.config.**FILE_INFLUENCE_PATCHES**
'influence_patches.csv'

vaws.model.config.**FILE_DAMAGE_COSTING_DATA**
'damage_costing_data.csv'

vaws.model.config.**FILE_DAMAGE_FACTORINGS**
'damage_factorings.csv'

vaws.model.config.**FILE_WATER_INGRESS_COSTING_DATA**
'water_ingress_costing_data.csv'

vaws.model.config.**FILE_FOOTPRINT**
'footprint.csv'

vaws.model.config.**FILE_FRONT_FACING_WALLS**
'front_facing_walls.csv'

 # debris data

vaws.model.config.**FILE_DEBRIS**
'debris.csv'

 # results

vaws.model.config.**FILE_RESULTS**
'results.h5'

vaws.model.config.**FILE_CURVE**
'results_curve.csv'

Note:

- regional shielding may be indirectly disabled by setting 'regional_shielding_factor' to 1.0
- differential shielding may be indirectly disabled by setting 'building_spacing' to 0

class vaws.model.config.**Config**(*cfg_file=None*)
Bases: object

Config class to set configuration for simulation

wind_dir
list

wind_dir
list

debris_types_keys
list

debris_types_atts
list

dominant_opening_ratio_thresholds
list

cpi_table_for_dominant_opening
dict

 # model dependent attributes

house_bucket =
list

att_non_float =
list

house_damage_bucket =
list

debris_bucket =
list

 # model and wind dependent attributes

list_components =
list

group_bucket =
list

connection_bucket
list

zone_bucket
list

coverage_bucket
list

dic_obj_for_fitting
dict

att_time_invariant
list

att_non_float = ['construction_level']

att_time_invariant = ['strength', 'strength_negative', 'strength_positive', 'dead_load', 'cpe', 'cpe_str

connection_bucket = ['damaged', 'capacity', 'load', 'strength', 'dead_load']

coverage_bucket = ['strength_negative', 'strength_positive', 'load', 'breached', 'breached_area', 'capa

cpi_table_for_dominant_opening = {0: {'windward': -0.3, 'leeward': -0.3, 'side2': -0.3, 'side1': -0.

debris_bucket = ['no_items', 'no_touched', 'damaged_area']

debris_types_atts = ['mass', 'frontal_area', 'cdav', 'ratio']

debris_types_keys = ['Rod', 'Compact', 'Sheet']

dic_obj_for_fitting = {'weibull': 'vulnerability_weibull', 'lognorm': 'vulnerability_lognorm'}

dominant_opening_ratio_thresholds = [0.5, 1.5, 2.5, 6.0]

static **get_diff_tuples**(*row*, *key1*, *key2*)

static **get_lognormal_tuple**(*row*, *dic_*, *key*)

group_bucket = ['damaged_area']

house_bucket = ['profile', 'wind_orientation', 'construction_level', 'mzcat', 'str_mean_factor', 'str_cov

house_damage_bucket = ['qz', 'ms', 'cpi', 'collapse', 'di', 'di_except_water', 'repair_cost', 'water_ingr

list_components = ['group', 'connection', 'zone', 'coverage']

process_config()

static **read_column_separated_entry**(*value*)

read_config()

read_construction_levels(*conf*, *key*)

static **read_damage_costing_data**(*filename*)

Parameters **filename** –

Returns:

Note: damage costing data may contain

static **read_damage_factorings**(*filename*)
Read damage factorings

Parameters **filename** – damage_factorings.csv

Returns: dict

read_debris(*conf*, *key*)

classmethod **read_file_connections**(*file_connections*, *types*)

classmethod **read_file_zones**(*file_zones*)

read_fragility_thresholds(*conf*, *key*)

static **read_front_facing_walls**(*filename*)

read_heatmap(*conf*, *key*)

static **read_influence_patches**(*filename*)
Read influence patches

Parameters **filename** – influence_patches.csv

Returns: dict

static **read_influences**(*filename*)
Read influence coefficients

Parameters **filename** – influences.csv

Returns: dict

read_main(*conf*, *key*)

Parameters

- **conf** –
- **key** –

Returns:

read_options(*conf*, *key*)

read_water_ingress(*conf*, *key*)
read water ingress related parameters :param conf: :param key:

Returns:

TODO:

static **read_water_ingress_costing_data**(*filename*)

static **return_centroid**(*row*)

static **return_norm_cdf**(*row*)

Parameters **row** –

Returns:

save_config(*filename=None*)

set_connections(*types*, *list_groups*)

set_construction_levels()

set_costings(*df_groups*)

set_coverages()

set_debris_types()

set_flight_time_log()

set_fragility_thresholds()

set_groups()

set_house()

set_influences_and_influence_patches()

set_output_files()

set_region_name(*value*)

set_types()

set_wawter_ingress()

set_wind_dir_index()

set_wind_profiles()

set_wind_speeds()

set_zones()

wind_dir = ['S', 'SW', 'W', 'NW', 'N', 'NE', 'E', 'SE', 'RANDOM']

zone_bucket = ['pressure', 'cpe', 'cpe_str', 'cpe_eave']

6.2.2 vaws.model.connection module

Connection module

This module contains Connection class, ConnectionTypeGroup class, and Influence class.

class vaws.model.connection.**Connection**(*connection_name=None*, ***kwargs*)
Bases: object

check_damage(*wind_speed*)

Parameters **wind_speed** –

Returns:

compute_load()
Returns: load

influence_patch

influences

sample_dead_load(*rnd_state*)

Parameters **rnd_state** –

Returns: sample of dead load following log normal dist.

sample_strength(*mean_factor*, *cov_factor*, *rnd_state*)
Return a sampled strength from lognormal distribution

Parameters

- **mean_factor** – factor adjusting arithmetic mean strength
- **cov_factor** – factor adjusting arithmetic cov
- **rnd_state** –

Returns: sample of strength following log normal dist.

update_influence(*source_connection*, *influence_coeff*)

Parameters

- **source_connection** – source connection (sheeting)
- **infl_coeff** –

Returns: load

class vaws.model.connection.**ConnectionTypeGroup**(*group_name=None*, ***kwargs*)

Bases: object

check_damage(*wind_speed*)

Parameters **wind_speed** – wind speed

Returns:

compute_damaged_area()

Returns: prop_damaged_group, prop_damaged_area

connection_by_grid

connections

damage_grid

grid_idx_by_dist_dir = {'col': 0, 'row': 1}

update_influence(*house_inst*)

Parameters **house_connections** –

Returns:

Notes

influence_patch will be applied regardless of patch_dist influence coeff will be distributed only if patch_dist == 0 dist_dir == 'col': coeff will be distributed over the same char. dist_dir == 'row': coeff will be distributed over the same number row: chr, col: number

static update_influence_by_patch(*damaged_connection*, *house_inst*)

Parameters **damaged_connection** – damaged_connection

Returns:

class vaws.model.connection.**Influence**(*name=None*, *coeff=None*)

Bases: object

source

6.2.3 vaws.model.coverage module

Coverage Module

This module contains Coverage class.

class vaws.model.coverage.**Coverage**(*coverage_name=None*, ***kwargs*)

Bases: *vaws.model.zone.Zone*

breached_area

check_damage(*qz*, *cpi*, *wind_speed*)

sample_strength(*rnd_state*)

6.2.4 vaws.model.curve module

Curve module

This module contains functions related to fragility and vulnerability curves.

vaws.model.curve.**fit_fragility_curves**(*cfg*, *df_dmg_idx*)

Parameters

- **cfg** –
- **df_dmg_idx** –

Returns: dict with keys of damage state

vaws.model.curve.**fit_vulnerability_curve**(*cfg*, *df_dmg_idx*)

Parameters

- **cfg** –
- **df_dmg_idx** –

Returns: dict of fitted_curve

vaws.model.curve.**vulnerability_lognorm**(*x*, *med*, *std*)

Return vulnerability in lognormal CDF

Parameters

- **x** – 3sec gust wind speed at 10m height
- **med** – exp(mean of log x)
- **std** – standard deviation of log x

Returns: lognorm.cdf(x, std, loc=0, scale=med)

vaws.model.curve.**vulnerability_weibull**(*x*, *alpha_*, *beta_*)

Return vulnerability in Weibull CDF

Parameters

- **x** – 3sec gust wind speed at 10m height
- **_alpha** – parameter value used in defining vulnerability curve
- **_beta** – ditto

Returns: weibull_min.cdf(x, shape, loc=0, scale)

Note:

weibull_min.pdf = c/s * (x/s)(c-1) * exp(-(x/s)**c)** c: shape, s: scale, loc=0

weibull_min.cdf = 1 - exp(-(x/s)**c)

while Australian wind vulnerability is defined as

DI = 1 - exp(-(x/exp(beta))**(1/alpha))

therefore:

s = exp(beta) c = 1/alpha

---

vaws.model.curve.**vulnerability_weibull_pdf**(*x*, *alpha_*, *beta_*)

Return PDF of vulnerability curve in Weibull

Parameters

- **x** – 3sec gust wind speed at 10m height
- **alpha** – parameter value used in defining vulnerability curve
- **beta** – ditto

Returns: weibull_min.cdf(x, shape, loc=0, scale)

6.2.5 vaws.model.damage_costing module

Damage Costing Module - costing profiles for a "type of damage"

- loaded from the database
- imported from house
- referenced by damage module to cost damages

class vaws.model.damage_costing.**Costing**(*costing_name=None*, ***kwargs*)

Bases: object

assign_costing_function(*key*)

compute_cost(*x*)

dic_costing = {1: 'type1', 2: 'type2'}

static type1(*x*, *c1*, *c2*, *c3*)

Parameters

- **x** – damage ratio between 0 and 1
- **c1** – coefficients
- **c2** –
- **c3** –

Returns: c1*x**2 + c2*x + c3

static type2(*x*, *c1*, *c2*, *c3*)

Parameters

- **x** – damage ratio between 0 and 1
- **c1** – coefficients

- **c2** –
- **c3** –

Returns: c1*x**c2

class vaws.model.damage_costing.**WaterIngressCosting**(*costing_name=None*, ***kwargs*)

Bases: object

assign_costing_function()

compute_cost(*damage_index*)

dic_costing = {1: 'type1', 2: 'type2'}

vaws.model.damage_costing.**compute_water_ingress_given_damage**(*damage_index*, *wind_speed*, *water_ingress*)

Parameters

- **damage_index** –
- **wind_speed** –
- **water_ingress** – pd.DataFrame

Returns:

6.2.6 vaws.model.debris module

Debris Module - adapted from JDH Consulting and Martin's work

- **given sim:wind_speed, sim:wind_dir,** scen:h:target_height, scen:h:target_width, scen:building_spacing, scen:debris_angle, scen:debris_radius, scen:flighttime_mean, scen:flighttime_stddev
- generate sources of debris.
- generate debris items from those sources.
- track items and sample target collisions.
- handle collisions (impact)

class vaws.model.debris.**Debris**(*cfg*)

Bases: object

amplification_factor = 5.0

angle_by_idx = {0: 90.0, 1: 45.0, 2: 0.0, 3: -45.0, 4: 90.0, 5: 45.0, 6: 0.0, 7: -45.0}

check_debris_impact(*frontal_area*, *item_momentum*)

Parameters

- **frontal_area** –
- **item_momentum** –

Returns self.breached

compute_debris_momentum(*cdav*, *frontal_area*, *flight_distance*, *mass*, *wind_speed*)
calculate momentum of debris object

Parameters

- **cdav** – average drag coefficient
- **frontal_area** –
- **flight_distance** –
- **mass** –
- **wind_speed** –

Returns: momentum of debris object

Notes

The ratio of horizontal velocity of the windborne debris object to the wind gust velocity is related to the horizontal distance travelled, x as below

um/vs approx.= 1-exp(-b*sqrt(x))

where um: horizontal velocity of the debris object vs: local gust wind speed x: horizontal distance travelled b: a parameter, sqrt(rho*CD*A/m)

The ratio is assumed to follow a beta distribution with mean, and

compute_flight_distance(*debris_type_str*, *flight_time*, *frontal_area*, *mass*, *wind_speed*, *flag_poly=2*)
calculate flight distance based on the methodology in Appendix of Lin and Vanmarcke (2008)

Parameters

- **debris_type_str** –
- **flight_time** –
- **frontal_area** –
- **mass** –
- **wind_speed** –
- **flag_poly** –

Returns:

Notes

The coefficients of fifth order polynomials are from

Lin and Vanmarcke (2008), while quadratic form are proposed by Martin.

coverages

static create_sources(*radius*, *angle*, *bldg_spacing*, *flag_staggered*, *restrict_y_cord=False*)
define a debris generation region for a building :param radius: :param angle: (in degree) :param bldg_spacing: :param flag_staggered: :param restrict_y_cord:

FIXME !! NO VALUE is assigned to restrict_yord

Returns:

flight_distance_coeff = {2: {'Compact': [0.011, 0.206], 'Rod': [0.2376, 0.0723], 'Sheet': [0.3456, 0.0

flight_distance_power = {2: [1, 2], 5: [2, 3, 4, 5]}

footprint

g_const = 9.81

generate_debris_item(*wind_speed*, *source*, *debris_type_str*)

Parameters

- **wind_speed** –
- **source** –
- **debris_type_str** –

Returns:

no_items_mean

rho_air = 1.2

rnd_state

run(*wind_speed*)

Parameters **wind_speed** –

Returns:

6.2.7 vaws.model.house module

House module

This module contains House class.

class vaws.model.house.**House**(*cfg*, *rnd_state*)
Bases: object

assign_costing(*key*)

Parameters **key** –

Returns:

assign_cpi()

assign_windward(*wall_name*)

link_connection_to_influence(*_connection*)

read_house_data()

set_connection_property(*_connection*)

Parameters **_connection** – instance of Connection class

Returns:

set_connections()

set_construction_level()
Returns: construction_level, str_mean_factor, str_cov_factor

set_coverages()

set_debris()

set_wind_orientation()

set_wind_profile()
Returns: profile, mzcat

set_zones()

6.2.8 vaws.model.house_damage module

class vaws.model.house_damage.**HouseDamage**(*cfg*, *seed*)
Bases: object

apply_damage_factoring(*area_by_group*)

check_house_collapse(*wind_speed*)

Parameters **wind_speed** –

Returns: collapse of house

check_internal_pressurisation(*wind_speed*)

Parameters **wind_speed** –

Returns self.cpi self.cpi_wind_speed

compute_area_by_group()

compute_area_by_scenario(*revised_area*, *total_area_by_group*)

compute_damage_index(*wind_speed*)

Parameters **wind_speed** –

Returns repair cost / replacement cost

Return type damage_index

Note:

1.compute sum of damaged area by group

2.revised damage area by group by applying damage factoring

3.calculate sum of revised damaged area by damage scenario

4.apply costing modules

---

compute_qz_ms(*wind_speed*)
calculate qz, velocity pressure given wind velocity qz = 0.6*10-3*(Mz,cat*V)**2 :param wind_speed: wind velocity (m/s)

Returns qz update ms

compute_water_ingress_cost(*damage_name*, *water_ingress_perc*)

determine_scenario_for_water_ingress_costing(*prop_area_by_scenario*)

fill_bucket()

init_bucket()

run_simulation(*wind_speed*)

6.2.9 vaws.model.main module

vaws.model.main.**init_bucket**(*cfg*)

vaws.model.main.**main**()

vaws.model.main.**process_commandline**()

vaws.model.main.**save_results_to_files**(*cfg*, *bucket*)

Parameters

- **cfg** –
- **bucket** –

Returns:

vaws.model.main.**set_logger**(*path_cfg*, *logging_level=None*)

Parameters

- **path_cfg** – path of configuration file
- **logging_level** – Level of messages that will be logged

Returns:

vaws.model.main.**simulate_wind_damage_to_houses**(*cfg*, *call_back=None*)

Parameters

- **cfg** – instance of Config class
- **call_back** – used by gui

Returns:

vaws.model.main.**update_bucket**(*cfg*, *bucket*, *results_by_speed*, *ispeed*)

6.2.10 vaws.model.output module

output.py - output module, postprocess and plot display engine

vaws.model.output.**draw_influence**(*cfg*, *infl_dic*, *dic_ax*, *conn_name*)

vaws.model.output.**plot_heatmap**(*grouped*, *values*, *vmin*, *vmax*, *vstep*, *xlim_max*, *ylim_max*, *file_name=None*)

Parameters

- **grouped** – pd.DataFrame (x_coord, y_coord, width, height)
- **values** – np.array(N,1)
- **vmin** – min. of scale in color bar
- **vmax** – max. of scale in color bar
- **vstep** – no. of scales in color bar
- **file_name** – file to save

Returns:

vaws.model.output.**plot_influence**(*cfg*, *conn_name*, *file_name=None*)

vaws.model.output.**set_axis_etc**(*ax*, *xlim_max*, *ylim_max*)

6.2.11 vaws.model.stats module

vaws.model.stats.**calc_big_a_b_values**(*shape_k*)

Parameters

- **shape_k** – parameter k of GEV III (JHD)
- **CDF** –
- **= u + a*A, s = a * B**(*m*) –
- **m**(*where*) – mean, s: std, u: location factor, a:scale factor
- **=**(*B*) –
- **=** –

Returns: A, B

vaws.model.stats.**calc_parameters_gev**(*mean_est*, *cov_est*, *big_a*, *big_b*)

Parameters

- **mean_est** – estimated mean (can be negative)
- **cov_est** – cov
- **big_a** – A value
- **big_b** – B value
- **CDF** –
- **m = u+a*A, s = a*B, cov = s/m**(*where*) –
- **= m*cov/B, u = m-a*A**(*a*) –

Returns:

vaws.model.stats.**compute_arithmetic_mean_stddev**(*m*, *stddev*)
compute arithmetic mean and std of x

Parameters

- **m** – mean of log x
- **stddev** – std of log x

Returns: arithmetic mean, std of x

vaws.model.stats.**compute_logarithmic_mean_stddev**(*m*, *stddev*)
compute mean of log x with mean and std. of x :param m: arithmetic mean of x :param stddev: arithmetic standard deviation of x :param mu = 2*log: :type mu = 2*log: m) - 0.5*log(v + m**2 :param std = sqrt: :type std = sqrt: log(V/m**2 +1) :param if m is zero, then return -999, 0.0:

Returns: mean and std of log x

vaws.model.stats.**sample_gev**(*mean_est*, *cov_est*, *big_a*, *big_b*, *shape_k*, *rnd_state=None*)
JHD F(u) = exp{-[1-k(U-u)/a]**(1/k)} where a: scale factor, u: location factor k < 0: Type II (Frechet), k > 0: Type III (Weibull)

scipy.stats.genextreme.rvs(c, loc=0, scale=1, size=1, random_state=None) c: shape (or k)

Parameters

- **mean_est** –
- **big_a** –
- **big_b** –
- **cov_est** –
- **shape_k** –
- **rnd_state** –

Returns: random sample from the extreme value distribution Type III

vaws.model.stats.**sample_lognorm_given_mean_stddev**(*m*, *stddev*, *rnd_state*)
generate rv following lognorm dist :param m: mean of x :param stddev: std of x :param rnd_state: :param size: size of rv (default: 1)

Returns:

vaws.model.stats.**sample_lognormal**(*mu_lnx*, *std_lnx*, *rnd_state*)
draw a sample from a lognormal distribution with mu, std of logarithmic x If std is zero, just return exp(mu_lnx)

Parameters

- **mu_lnx** – mean of log x
- **std_lnx** – std of log x
- **rnd_state** – numpy.random.RandomState

Returns:

6.2.12 vaws.model.version module

Single point of truth for version information

6.2.13 vaws.model.zone module

Zone Module - reference storage

- loaded from database.
- imported from '../data/houses/subfolder/zones.csv'
- holds zone area and CPE means.
- holds runtime sampled CPE per zone.
- calculates Cpe pressure load from wind pressure.

class vaws.model.zone.**Zone**(*zone_name=None*, ***kwargs*)
Bases: object

calc_diff_shielding(*ms*, *building_spacing*, *flag_diff_shielding*, *wind_dir_index*)

calc_zone_pressure(*wind_dir_index*, *cpi*, *qz*, *ms*, *building_spacing*, *flag_diff_shielding=False*)
Determine wind pressure loads (Cpe) on each zone (to be distributed onto connections)

Parameters

- **wind_dir_index** –
- **cpi** – internal pressure coeff
- **qz** –
- **ms** –
- **building_spacing** –
- **flag_diff_shielding** – flag for differential shielding (default: False)

Returns zone pressure

Return type pressure

static get_grid_from_zone_location(*_zone_name*)
Extract 0 based grid refs from string location (eg 'A10' to 0, 9)

static get_zone_location_from_grid(*_zone_grid*)
Create a string location (eg 'A10') from zero based grid refs (col=0, row=9)

Parameters **_zone_grid** – tuple

Returns: string location

sample_cpe(*wind_dir_index*, *cpe_cov*, *cpe_k*, *big_a*, *big_b*, *cpe_str_cov*, *cpe_str_k*, *big_a_str*, *big_b_str*, *rnd_state*)
Sample external Zone Pressures for sheeting, structure and eaves Cpe, based on TypeIII General Extreme Value distribution. Prepare effective zone areas for load calculations.

Parameters

- **wind_dir_index** – 0 to 7
- **cpe_cov** – cov of dist. of CPE for sheeting and batten
- **cpe_k** – shape parameter of dist. of CPE
- **big_a** –

- **big_b** –
- **cpe_str_cov** – cov. of dist of CPE for rafter
- **cpe_str_k** –
- **big_a_str** –
- **big_b_str** –
- **rnd_state** – default None

Returns: cpe, cpe_str, cpe_eave

vaws.model.zone.**num2str**(*n*)
n: number return string

vaws.model.zone.**str2num**(*s*)
s: string return number

BIBLIOGRAPHY

[JDH2010a] JDH Consulting, Randomized wind gust profiles, Report to Geoscience Australia, 2010.

[GA2011] Geoscience Australia, 2011. Wind Vulnerability Model Development for Adaptation Studies on Specific Residential and Industrial Buildings. Report to the Department of Climate Change and Energy Efficiency, Geoscience Australia.

[JDH2010b] JDH Consulting, 2010. Contributions to Development of a Wind Vulnerability Model - Final Report 2009-10. Report to Geoscience Australia.

[SA2011] Standards Australia, 2011. Australian/New Zealand Standard AS/NZS 1170.2:2011 Structural design actions Part 2: Wind actions.

[TuTo2006] Turner & Townsend Rawlinsons (T&TR), 2006. Wind Damage Cost Module Development for North Queensland Building Types. Report to Geoscience Australia in four volumes.

[JDH2010c] Holmes, J.; Wehner, M.; Sandland, C. & Edwards, M. Modelling damage to residential buildings from wind-borne debris – Part 1. Methodology 14th Australasian Wind Engineering Society Workshop, 2010

[JDH2010d] Holmes, J.; Wehner, M.; Sandland, C. & Edwards, M. Modelling damage to residential buildings from wind-borne debris – Part 2. Implementation 14th Australasian Wind Engineering Society Workshop, 2010

PYTHON MODULE INDEX

V

INDEX

D

F

G

H

I

L

M

N

O

P

Q

R

U

V

W

Z